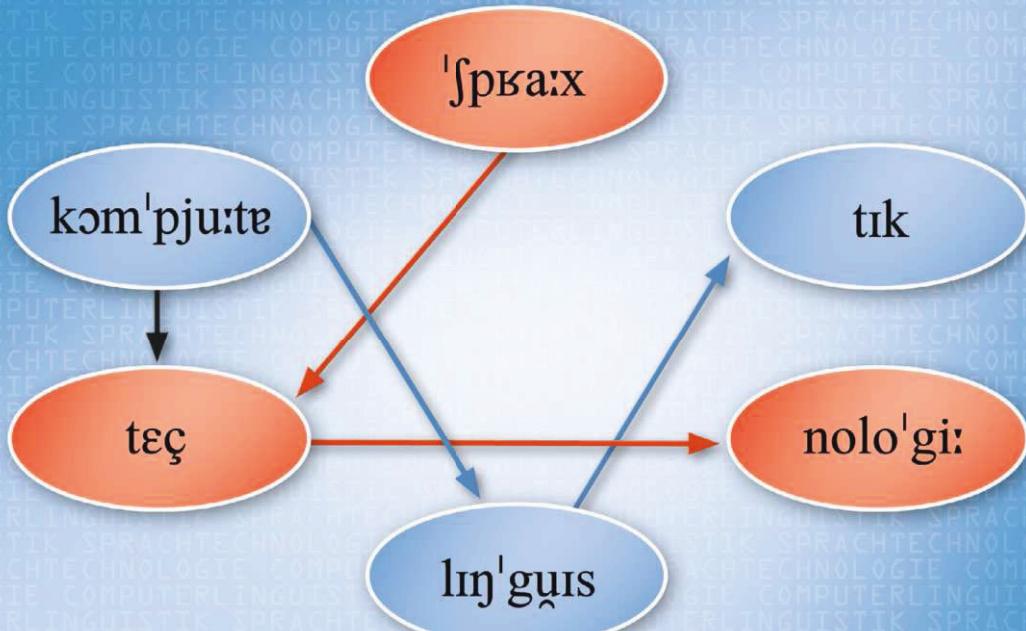


K.-U. Carstensen Ch. Ebert C. Ebert
S. Jekat R. Klabunde H. Langer (Hrsg.)

Computerlinguistik und Sprachtechnologie

Eine Einführung

3. Auflage



Computerlinguistik und Sprachtechnologie

Computerlinguistik und Sprachtechnologie

Eine Einführung

3., überarbeitete und erweiterte Auflage

Herausgegeben von
Kai-Uwe Carstensen, Christian Ebert,
Cornelia Ebert, Susanne Jekat,
Ralf Klabunde und Hagen Langer

Spektrum
AKADEMISCHER VERLAG

Herausgeber

Dr. Kai-Uwe Carstensen
Ruhr-Universität Bochum

Prof. Dr. Susanne J. Jekat
Zürcher Hochschule Winterthur

Dr. Christian Ebert
Universität Tübingen

Prof. Dr. Ralf Klabunde
Ruhr-Universität Bochum

Dr. Cornelia Ebert
Universität Osnabrück

Dr. habil. Hagen Langer
Universität Bremen

Für weitere Informationen zum Buch siehe:
www.linguistics.rub.de/CLBuch

Wichtiger Hinweis für den Benutzer

Der Verlag, die Herausgeber und die Autoren haben alle Sorgfalt walten lassen, um vollständige und akkurate Informationen in diesem Buch zu publizieren. Der Verlag übernimmt weder Garantie noch die juristische Verantwortung oder irgendeine Haftung für die Nutzung dieser Informationen, für deren Wirtschaftlichkeit oder fehlerfreie Funktion für einen bestimmten Zweck. Der Verlag übernimmt keine Gewähr dafür, dass die beschriebenen Verfahren, Programme usw. frei von Schutzrechten Dritter sind. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Buch berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürfen. Der Verlag hat sich bemüht, sämtliche Rechteinhaber von Abbildungen zu ermitteln. Sollte dem Verlag gegenüber dennoch der Nachweis der Rechtsinhaberschaft geführt werden, wird das branchenübliche Honorar gezahlt.

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Springer ist ein Unternehmen von Springer Science+Business Media
springer.de

3. Auflage 2010

© Spektrum Akademischer Verlag Heidelberg 2010
Spektrum Akademischer Verlag ist ein Imprint von Springer

10 11 12 13 14 5 4 3 2 1

Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Planung und Lektorat: Dr. Andreas Rüdinger, Bianca Alton

Satz: Autorensatz

Herstellung: Crest Premedia Solutions (P) Ltd, Pune, Maharashtra, India

Umschlaggestaltung: SpieszDesign, Neu-Ulm

ISBN 978-3-8274-2023-7

Geleitwort

Das Erscheinen dieses Handbuchs ist ein relevantes Ereignis in der Geschichte der Computerlinguistik im deutschsprachigen Raum. Diese Geschichte ist kurz, aber äußerst dynamisch. In den achtziger Jahren kämpfte eine recht kleine Truppe um ein eigenständiges Profil im Rahmen der Mutterdisziplinen Sprachwissenschaft und Informatik und suchte den Anschluss an die faszinierenden theoretischen und methodischen Fortschritte der amerikanischen Computerlinguistik. Inzwischen hat sich das Fach methodisch und institutionell etabliert – und hat gleichzeitig durch Brückenschläge in die Kognitions-, Neuro- und Ingenieurswissenschaften noch an reizvoller Interdisziplinarität gewonnen. Die Computerlinguistik in Europa hat sich eine ausgesprochen starke Position im weltweiten Vergleich erobert. Eine geradezu explosionsartige Entwicklung nimmt die Sprachtechnologie als kommerziell orientierte Anwendungsdisziplin; sie verspricht, zu einer Schlüsseltechnologie im beginnenden 21. Jahrhundert zu werden.

Die rasante Entwicklung, besonders im Anwendungsbereich, sorgt für steigende öffentliche Aufmerksamkeit und zunehmende Forschungsmittel, und sie liefert neue hoch interessante Fragestellungen und Forschungsthemen. Sie birgt aber auch Probleme. Damit meine ich nicht nur die Schwerpunktverschiebung von der Grundlagenforschung zur kurzfristigen Anwendung. Zeitdruck und Entwicklungstempo machen es schwer, den grundlegenden Aufgaben Aufmerksamkeit zu widmen, die für langfristiges Wachstum und Fortbestand des Faches als Grundlagen- und als Anwendungsdisziplin unabdingbar sind: Dazu gehören die sorgfältige Standortbestimmung des Faches, die Aufbereitung und Systematisierung von fachlichem Wissen und die Bereitstellung des Wissens für Fachleute und ein breiteres Publikum.

Durch dieses Handbuch wird für den deutschsprachigen Raum eine empfindliche Lücke geschlossen. Es umfasst die relevanten Aspekte computerlinguistischer Grundlagenwissenschaft und sprachtechnologischer Anwendung in eindrucksvoller Breite und auf aktuellem Stand. Ich rechne damit, dass das Handbuch seinen Platz als Standardwerkzeug für Studierende, für interessierte Wissenschaftler der Nachbardisziplinen und aus der industriellen Anwendung einnehmen wird. Dass sich zu diesem Unternehmen Herausgeber und Autoren – zahlreiche, überwiegend junge Wissenschaftlerinnen und Wissenschaftler – gefunden haben, ist sehr dankenswert und nebenbei ein deutliches Zeichen für die Lebendigkeit und Zukunftsfähigkeit unseres Faches als wissenschaftlicher Disziplin.

Manfred Pinkal

Geleitwort zur zweiten Auflage

Im Sommer 1998 saß ich während der European Summer School on Logic, Language and Linguistics (ESSLLI) in Saarbrücken mit einem Kollegen beim Frühstück – wir diskutierten künftige Lehrveranstaltungen. Er erzählte mir, dass er im kommenden Semester eine Einführung in die Computerlinguistik anbieten würde und ich fragte, auf welchem Lehrwerk der Kurs aufsetzen würde. Seine Antwort – er plante eine Einführung aus dem Jahr 1989 zu verwenden – verblüffte und erstaunte mich in mehrfacher Hinsicht. Da war zum einen die dort etwas eigentümlich umgesetzte Praxisorientierung, integrale Teile des Werks in einer Programmiersprache darzustellen, die man dann allerdings schon beherrschten musste. Dann hatte die Computerlinguistik gerade in den frühen 1990er Jahren gänzlich andere Wege beschritten als in den späten 1980ern. Das vielleicht überraschendste Moment war jedoch, dass ich trotz aller Bedenken zu diesem Lehrwerk insbesondere im deutschsprachigen Raum, aber auch international keine Alternative sah, *weil es tatsächlich kein anderes aktuelles Lehrwerk gab.*

Als damaliger Sprecher der Sektion Computerlinguistik der Deutschen Gesellschaft für Sprachwissenschaft habe ich dieses Forum genutzt, um auf diese Lücke aufmerksam zu machen. Wir kamen darin überein, dass es sinnvoll, ja sogar notwendig wäre, ein solches Lehrwerk zu haben. Ich bin nun außerordentlich erfreut, dass die resultierende Einführung nicht nur begeisterte Aufnahme fand, sondern nach wenigen Jahren bereits die zweite, substanzial überarbeitete Auflage erscheinen kann. Dies führt einerseits zu einer erneuten Aktualisierung, zeigt andererseits darüber hinaus auch, dass das vorliegende Buch die mittlerweile vorhandene Konkurrenz keineswegs zu fürchten hat.

Die letzten 20 Jahre haben innerhalb der Computerlinguistik überraschende Paradigmenwechsel und das Kommen und Gehen unterschiedlichster Trends gesehen. Sollte ich den gegenwärtigen Entwicklungsstand der Computerlinguistik charakterisieren, so würde ich sagen, dass wir zur Zeit vor einer reflektierten Synthese der deduktiven, theorie- und logiklastigen Methoden der 1980er und der eher induktiven, datenbasierten Verfahren der 1990er stehen. Das Zusammenführen scheinbar komplementärer Positionen gestattet unter Umständen überhaupt erst ein Weiterkommen, bringt aber auch erhöhte Anforderungen mit sich. Sinnvolle computerlinguistische Anwendungen bauen auf computerlinguistischen Grundlagen auf. Es ist daher umso bemerkenswerter, dass die Autoren dieses Buchs in Absehung des notorisch entropischen „Tagesgeschäfts“ die Zeit gefunden haben, die Grundlagen der Computerlinguistik in eine Form zu bringen, die die Vermittlung nicht nur ermöglicht sondern auch beschleunigt.

Tibor Kiss

Geleitwort zur dritten Auflage

Im Geleitwort zur ersten Auflage sprach Manfred Pinkal von den Herausgebern als „überwiegend junge[n] Wissenschaftlerinnen und Wissenschaftlern“ und wir hatten es mit etwas Neuem zu tun, einem Unternehmen, dem man viel Erfolg wünschen wollte, ohne jedoch sicher zu sein, ob sich dieser Erfolg auch einstellen würde. Es war der erste Versuch, eine deutschsprachige Einführung in die Computerlinguistik und Sprachtechnologie zu schreiben. Computerlinguistik und Sprachtechnologie hatten sich in Deutschland mittlerweile, unter anderem mit großen und ambitionierten Projekten wie LiLog und VerbMobil, mit Erfolg etabliert, aber ein deutschsprachiges Lehrbuch oder Handbuch gab es anno 2000 noch nicht.

Die Lage hat sich in den inzwischen vergangenen zehn Jahren deutlich geändert. Das Buch ist seit langem eine Institution der deutschsprachigen Computerlinguistik, wird an Universitäten und Fachhochschulen als Standardwerk in der Lehre eingesetzt, und die vorliegende dritte Auflage braucht keinen Geleitschutz mehr.

Im Untertitel stellt sich das vorliegende Buch als eine Einführung vor. Gemeint ist damit wohl in erster Linie, dass es systematisch aufgebaut ist, alle wichtigen Bereiche der Computerlinguistik und Sprachtechnologie abdeckt und dass es verständlich geschrieben ist. Aber es handelt sich nicht im eigentlichen Sinn um ein Lehr- und Unterrichtswerk, das primär didaktische Ziele hätte und die neuesten Entwicklungen des Faches dann doch lieber anderen Publikationen überließe. Das Buch ist klar auf dem neuesten Stand der Forschung, die einzelnen Abschnitte stammen von Autorinnen und Autoren, die auf den jeweiligen Spezialgebieten in der Forschung aktiv sind, und die aktuell breit diversifizierten Entwicklungen von Sprachtechnologie und Computerlinguistik sind hervorragend repräsentiert, in der vorliegenden Ausgabe noch einmal besser als in der zweiten Ausgabe. Das Buch ist nicht nur hervorragend für die Lehre, auch in fortgeschrittenen Lehrveranstaltungen, geeignet. Es hat auch alles, was man von einem guten Handbuch erwarten würde.

Schade nur, dass es noch nichts vergleichbares auf Englisch gibt.

Peter Bosch

Vorwort

Diese vorliegende Einführung in die maschinelle Sprachverarbeitung resultiert aus unserem Bemühen, ein deutschsprachiges Einführungsbuch zu konzipieren, das Studenten der Computerlinguistik und verwandter Fächer nicht nur das Wissen über die Grundlagen und Methoden darbietet, sondern auch den Bereich der Sprachtechnologie vorstellt, in dem die verschiedenen Grundlagen und Methoden Verwendung finden. Wir hoffen, dass dadurch deutlich wird, wie weit die Computerlinguistik mit ihren Ergebnissen und deren Umsetzung in diversen Anwendungen das alltägliche Leben in unserer modernen Informationsgesellschaft schon durchdrungen hat und wünschen uns natürlich, dass das Buch hierdurch noch viele Leser motivieren wird, sich intensiver mit der maschinellen Verarbeitung natürlicher Sprache zu beschäftigen.

Die Strukturierung des Buchs in die fünf Kapitel Grundlagen, Methoden, Ressourcen, Anwendungen und Evaluation soll dazu dienen, die wesentlichen Wissensbereiche in der maschinellen Sprachverarbeitung abzudecken. Gleichzeitig haben wir versucht, die Transfers zwischen diesen Wissensbereichen durch Querverweise auf die jeweilig relevanten Beiträge transparent zu machen, so dass deutlich wird, welche Grundlagen für welche Methoden und Anwendungen einschlägig sind, welche Methoden in welchen Systemen Anwendung finden, welche Ressourcen hierfür verwendet werden, und wie die Qualität eines sprachverarbeitenden Systems angemessen bestimmt werden kann.

Die Zusammenarbeit der Herausgeber für die Konzeption dieses Buchs hat einige Reiseaktivitäten nötig gemacht. Die Sektion Computerlinguistik der Deutschen Gesellschaft für Sprachwissenschaft (DGfS/CL) hat uns hierbei dankenswerterweise finanziell unterstützt. Danken möchten wir auch den folgenden Personen, die die Herausgeber bzw. die Beiträger in vielfältiger Weise hilfreich unterstützt haben: Sven Behnke, Peter Bosch, Stefanie Dipper, Gerald Friedland, Martin Glockemann, Alexander Gloye, Christopher Habel, Walther von Hahn, Michael Hess, Gerhard Jäger, Wolfgang Kehrein, Anke Lüdeling, Sabine Reinhard, Ingrid Renz, Raul Rojas, Michael Schiehlen, Lorenzo Tessiore, Andreas Wagner und Richard Wiese.

Schließlich möchten wir uns ganz herzlich bei allen Autorinnen und Autoren bedanken, ohne deren besonderes Engagement dieses Buch nicht möglich gewesen wäre.

Die Herausgeber

Vorwort zur zweiten Auflage

Die überaus positive Resonanz auf die Erstauflage dieses Buchs resultiert in der vorliegenden zweiten Auflage, die sich jedoch von der ersten Auflage nicht nur im Umfang, sondern zum Teil auch in der Gliederung unterscheidet. Diese Unterschiede sind hauptsächlich darin begründet, dass wir Bereiche, die in der Erstauflage unterrepräsentiert waren, jetzt umfangreicher vorstellen. Dies sind zum einen die maschinelle Auflösung anaphorischer Ausdrücke, die Verwendung von fokussierter Information und die Verwendung sogenannter flacher Verfahren zur Satzverarbeitung wie z.B. die automatische Wortartenbestimmung. Zum anderen wird in separaten Unterkapiteln auf texttechnologische Grundlagen und Ressourcen sowie auf die Repräsentation und Verarbeitung ontologischen Wissens eingegangen. Ein weiterer Unterschied zur Erstauflage besteht darin, dass in den meisten Beiträgen neuere Entwicklungen erläutert und relevante Literatur angegeben wird.

Gleichwohl hat jedes Buch nur eine endliche Anzahl von Seiten, und auch wir mussten uns bei der Auswahl der Themen und dem Seitenumfang für ihre Darstellung beschränken. So liegt der Schwerpunkt des Buchs auf den vielfältigen symbolischen Verfahren, die in der Sprachtechnologie verwendet werden, obwohl jedes Kapitel natürlich auch entsprechende Abschnitte über probabilistische Grundlagen und Verfahren enthält. Generell schien es uns für ein Einführungsbuch wichtiger zu sein, das Basiswissen verständlich darzustellen, als viel – wenn auch sehr interessantes – Detailwissen zu vermitteln.

Wie bereits bei der Erstauflage hat die Sektion Computerlinguistik der Deutschen Gesellschaft für Sprachwissenschaft (DGfS/CL) die Reisekosten der Herausgeber übernommen. Hierfür bedanken wir uns sehr herzlich. Dank gilt ebenfalls den Studierenden, die uns auf Fehler und Unzulänglichkeiten in der ersten Auflage hingewiesen haben. Außerdem danken wir den folgenden Personen, deren Hinweise und Kommentare zur Konzeption der zweiten Auflage beigetragen haben: Inge Endriss, Markus Greif, Daniela Hagenbruch, Emina Kurtic, Anna-Katharina Pantli, Stephanie Polubinski, David Reitter und Jan Strunk.

Unser ganz besonderer Dank geht wieder an die Autorinnen und Autoren für ihre Bereitschaft, an diesem Buch mitzuarbeiten.

Die Herausgeber

Vorwort zur dritten Auflage

Die Existenz dieser dritten Auflage spricht dafür, dass der Bedarf an einem deutschsprachigen umfassenden Einführungswerk in die Computerlinguistik und Sprachtechnologie immer noch hoch ist. Wir haben mit dieser neuen Auflage versucht, den stetigen Änderungen, denen die Forschung und Entwicklung zur maschinellen Sprachverarbeitung unterworfen ist, durch eine Restrukturierung der vorherigen Auflage Rechnung zu tragen. So wird stärker als bisher die Relevanz der Wahrscheinlichkeitstheorie für die Theoriebildung und die Entwicklung sprachverarbeitender Systeme berücksichtigt. Zudem haben wir im Anwendungskapitel einzelnen vorgestellten Anwendungen zwar mehr Umfang eingeräumt, ihre Zahl aber zugunsten einer stärkeren Anbindung an die Theorie und Methodik reduziert. Wir hoffen, dass dadurch noch deutlicher wird, wie die Sprachtechnologie mit den zugehörigen wissenschaftlichen Theorien verzahnt ist.

Für Verbesserungshinweise, Vorschläge sowie Hilfestellungen bei der Anfertigung des Manuskripts für die dritte Auflage danken wir Stefan Freund, Zeno Gantner, Janine Kerbei, Oliver Lomp, Anke Lüdeling und Eva Struebin.

Letztlich danken wir wieder ganz besonders den Autorinnen und Autoren, die immer wieder – und oftmals mit Mühen – in ihren vollen Terminkalendern Platz geschaffen haben, um ihre Beiträge für dieses Buch rechtzeitig fertig zu stellen.

Die Herausgeber

Inhaltsverzeichnis

1 Computerlinguistik – Was ist das?	1
1.1 Aspekte der Computerlinguistik	1
1.1.1 Computerlinguistik: Die Wissenschaft	2
1.1.2 Computerlinguistik und ihre Nachbardisziplinen	3
1.1.3 Teilbereiche der Computerlinguistik	6
1.1.4 Theoretische Computerlinguistik	8
1.1.5 Wissensbereiche	11
1.1.6 Industrielle Anwendungen	14
1.1.7 Berufsfelder für Computerlinguisten	16
1.1.8 Literaturhinweise	17
1.2 Zur Geschichte der Computerlinguistik	18
1.2.1 Die Ursprünge	18
1.2.2 Symbolische Sprachverarbeitung	19
1.2.3 Korpusstatistische Verfahren	21
1.2.4 Anwendungen der Computerlinguistik	23
2 Formale Grundlagen	27
2.1 Mengenlehre und Logik	28
2.1.1 Mengenlehre	28
2.1.2 Aussagenlogik	33
2.1.3 Prädikatenlogik	45
2.1.4 Typenlogik	53
2.1.5 Der Lambda-Kalkül	60
2.1.6 Literaturhinweise	65
2.2 Automatentheorie und Formale Sprachen	66
2.2.1 Grundlegende Definitionen	66
2.2.2 Grammatiken	67
2.2.3 Endliche Automaten, einseitig-lineare Grammatiken und reguläre Sprachen	70
2.2.4 Kontextfreie Sprachen und Grammatiken	79
2.2.5 Nicht-kontextfreie Sprachen und Grammatiken	84
2.2.6 Komplexitäts- und Entscheidbarkeitseigenschaften	90
2.2.7 Zusammenfassung	92
2.2.8 Literaturhinweise	93
2.3 Graphentheorie und Merkmalsstrukturen	94
2.3.1 Graphen und Bäume	94
2.3.2 Merkmalsstrukturen	97
2.3.3 Unifikation	103

2.3.4 Generalisierung	106
2.3.5 Typisierte Merkmalsstrukturen	108
2.3.6 Literaturhinweise	113
2.4 Statistische Grundlagen	114
2.4.1 Wahrscheinlichkeitstheoretische Grundlagen	114
2.4.2 Hidden-Markov-Modelle	130
2.4.3 Evaluation und Optimierung statistischer Modelle	147
2.4.4 Literaturhinweise	157
2.5 Texttechnologische Grundlagen	159
2.5.1 HTML – Hypertext Markup Language	160
2.5.2 XML – Extensible Markup Language	161
2.5.3 Verarbeitung XML-annotierter Daten	163
2.5.4 Texttechnologie und Computerlinguistik	167
2.5.5 Literaturhinweise	168
3 Methoden	169
3.1 Phonetik und Phonologie	170
3.1.1 Grundlagen der Computerphonologie	172
3.1.2 Empirische Methoden	190
3.1.3 Formale Methoden	197
3.1.4 Zusammenfassung und weitergehende Lektüre	211
3.2 Verarbeitung gesprochener Sprache	214
3.2.1 Spracherkennung	215
3.2.2 Sprachsynthese	223
3.2.3 Gemeinsamkeiten und Unterschiede	229
3.2.4 Literaturhinweise	235
3.3 Morphologie	236
3.3.1 Überblick	236
3.3.2 Grundbegriffe und -probleme	236
3.3.3 Modelle aus der Generativen Linguistik	240
3.3.4 Morphologie mit endlichen Automaten	244
3.3.5 Default-Vererbungsnetze: DATR	251
3.3.6 Erweiterte Finite-State-Ansätze	257
3.3.7 Morphologie und generative Kapazität	262
3.3.8 Zusammenfassung und Ausblick	263
3.3.9 Literaturhinweise	263
3.4 Flache Satzverarbeitung	264
3.4.1 Tokenisierung	264
3.4.2 Wortart-Tagging	271
3.4.3 Chunk-Parsing	275
3.4.4 Literaturhinweise	278
3.5 Syntax und Parsing	280
3.5.1 Syntax	281
3.5.2 Parsing	303
3.5.3 Literaturhinweise	328

3.6	Semantik	330
3.6.1	Grundlagen der natürlichsprachlichen Semantik	332
3.6.2	Formale Semantik	339
3.6.3	Diskursrepräsentationstheorie	359
3.6.4	Ansätze zur Unterspezifikation	371
3.6.5	Lexikalische Semantik	377
3.6.6	Literaturhinweise	393
3.7	Pragmatik	394
3.7.1	Text, Diskurs und Dialog	395
3.7.2	Anaphernresolution	399
3.7.3	Implikaturen und Präspositionen	410
3.7.4	Benutzermodellierung	422
3.8	Textgenerierung	436
3.8.1	Aufgaben der Planung und Umsetzung	437
3.8.2	Funktionalität des Planungsprozesses	439
3.8.3	Methoden zur Diskursplanung	446
3.8.4	Satzplanungsverfahren	453
3.8.5	Verfahren zur Oberflächenrealisierung	458
3.8.6	Linguistische Theorien zur Generierung	461
3.8.7	Ausblick	464
3.8.8	Literaturhinweise	465
3.9	Programmiersprachen in der Computerlinguistik	466
3.9.1	Die Anfänge: Hochsprachen und symbolische Sprachverarbeitung	466
3.9.2	C/C++	469
3.9.3	Programmierarchitekturen: Java und .Net	472
3.9.4	Dynamische Sprachen: Perl und Python	475
3.9.5	Von der Desktop- zur Web-Applikation	479
4	Ressourcen	481
4.1	Korpora	482
4.1.1	Aufbau eines Korpus	483
4.1.2	Typologie	486
4.1.3	Anwendungen	489
4.1.4	Weiterführende Informationen	491
4.2	Baumbanken	492
4.2.1	Zentrale Eigenschaften	492
4.2.2	Die wichtigsten Baumbanken	496
4.2.3	Suche in Baumbanken	502
4.2.4	Literaturhinweise	503
4.3	Lexikalisch-semantische Ressourcen	504
4.3.1	Lexikalisch-semantische Wortnetze	504
4.3.2	FrameNet	511
4.3.3	Literaturhinweise	514
4.4	Lexika für multimodale Systeme	515
4.4.1	Grundlagen	515

4.4.2	Die Lexikographie	517
4.4.3	Lexikalische Struktur- und Informationstypen	520
4.4.4	Literaturhinweise	523
4.5	Sprachdatenbanken	524
4.5.1	Definition	524
4.5.2	Primärdaten	524
4.5.3	Sekundärdaten	526
4.5.4	Tertiärdaten	528
4.5.5	Software	529
4.5.6	Anwendungsgebiete	530
4.5.7	Literaturhinweise	531
4.6	Nicht-sprachliches Wissen	532
4.6.1	Die Relevanz nicht-sprachlichen Wissens für die CL	532
4.6.2	Was ist „Wissen“ (nicht)?	533
4.6.3	Wissen und Wissensrepräsentation	533
4.6.4	Aspekte der Wissensrepräsentation	534
4.6.5	Wissensrepräsentation für die CL	541
4.6.6	Literaturhinweise	543
4.7	Das World Wide Web als computerlinguistische Ressource	544
4.7.1	Einleitung	544
4.7.2	Aspekte des Web als Korpus	544
4.7.3	Sozio-Semantisches Web	546
4.7.4	Sprachverarbeitungsanwendungen mit Nutzung des World Wide Web als Ressource	550
4.7.5	Computerlinguistik und Sprachtechnologie für das Web	550
4.7.6	Literaturhinweise	551
5	Anwendungen	553
5.1	Korrektursysteme	555
5.1.1	Korrektur von Nichtwörtern	556
5.1.2	Kontextabhängige Korrektur	559
5.1.3	Rechtschreibkorrektur für Suchmaschinen	561
5.1.4	Grammatikkorrektur	562
5.1.5	Perspektiven	563
5.1.6	Literaturhinweise	564
5.2	Computergestützte Lexikographie und Terminologie	566
5.2.1	Lexikographie und Terminologie	566
5.2.2	Die Teilbereiche im Überblick	567
5.2.3	Akquisition von lexikalischem Wissen	568
5.2.4	Verwaltung und Repräsentation lexikalischen Wissens	571
5.2.5	Nutzung von lexikalischem Wissen	573
5.2.6	Computerlinguistische Unterstützung lexikographischer Arbeit	574
5.2.7	Literaturhinweise	575
5.3	Text-basiertes Informationsmanagement	576
5.3.1	Überblick	576

5.3.2	Information Retrieval	587
5.3.3	Informationsextraktion	594
5.3.4	Domänenoffene Fragebeantwortung	606
5.3.5	Textzusammenfassung	611
5.3.6	Multilinguale und sprachübergreifendes TIM	613
5.3.7	Perspektiven	614
5.3.8	Literaturhinweise	615
5.4	Sprachein- und -ausgabe	616
5.4.1	Spracheingabe	616
5.4.2	Sprachausgabe	621
5.4.3	Literaturhinweise	623
5.5	(Multimodale) Dialogsysteme	624
5.5.1	Multimodale Kommunikation	624
5.5.2	Sprachdialogsysteme	626
5.5.3	Struktur eines multimodalen Dialogsystems	628
5.5.4	Modellierung und Repräsentation	631
5.5.5	Literaturhinweise	632
5.6	Angewandte natürlichsprachliche Generierungs- und Auskunftsysteme	633
5.6.1	Was ist angewandte NLG?	634
5.6.2	Beispiele für angewandte NLG-Systeme	635
5.6.3	Mechanismen und Methoden	637
5.6.4	Perspektiven	640
5.6.5	Literaturhinweise	641
5.7	Maschinelle und computergestützte Übersetzung	642
5.7.1	Einleitung	642
5.7.2	MÜ-Ansätze	644
5.7.3	Regel-basierte Systeme	645
5.7.4	Statistische Maschinelle Übersetzung	647
5.7.5	Evaluation von MÜ-Systemen	653
5.7.6	Computergestützte Übersetzung – CAT	654
5.7.7	Aktueller Stand und Perspektiven	656
5.7.8	Literaturhinweise	657
6	Evaluation von sprachverstehenden und -generierenden Systemen	659
6.1	Einführung	659
6.1.1	Warum wird evaluiert?	659
6.1.2	Wann und wie wird evaluiert?	660
6.1.3	Was wird evaluiert?	662
6.2	Evaluationskriterien für sprachverarbeitende Systeme	664
6.2.1	Spracherkennungssysteme	664
6.2.2	Evaluation von Dialogsystemen	669
6.2.3	Informationssuchsysteme	674
6.2.4	Sprachsynthesessysteme	674
6.2.5	Maschinelle Übersetzung	675
6.2.6	Fazit	677

6.2.7 Literaturhinweise	678
Literaturverzeichnis	679
Index	717
Die Autorinnen und Autoren	734

1 Computerlinguistik – Was ist das?

Kapitelherausgeber: Kai-Uwe Carstensen, Susanne Jekat und Ralf Klabunde

Die Computerlinguistik ist das Fachgebiet, das sich mit der maschinellen Verarbeitung natürlicher Sprache beschäftigt. Sie ist im Überschneidungsbereich von Informatik und Linguistik angesiedelt, aber die Wurzeln der Computerlinguistik reichen bis in die fünfziger Jahre zurück. In diesem halben Jahrhundert seit ihrem Entstehen hat sie sich mittlerweile national und international erfolgreich etabliert, so dass auf dem Wissen aus der Informatik und der Linguistik aufbauend neue und eigenständige Methoden für die maschinelle Verarbeitung gesprochener und geschriebener Sprache entwickelt wurden.

Unterkapitel 1.1 bringt die in diesem Buch dargestellten Grundlagen, Methoden und Anwendungen in einen umfassenden Bezug zu den verschiedenen Aufgaben der Computerlinguistik.

Anschließend werden in Unterkapitel 1.2 die zwei Verarbeitungsparadigmen der Computerlinguistik, die symbolische und die stochastische Verarbeitung, aus historischer Sicht vorgestellt.

1.1 Aspekte der Computerlinguistik

Jan W. Amtrup

Der Einfluss der Computerlinguistik (CL) auf das tägliche Leben in unserer „Informationsgesellschaft“ wächst. Es ist fast unvermeidlich, dass man mit den aus dieser relativ neuen Wissenschaft entstandenen Produkten in Kontakt kommt, sei es beim Surfen im Internet oder beim normalen Gebrauch des Computers. Ein Achtklässler, der am Computer einen Hausaufsatz schreibt, benutzt morphologische Prozesse (Rechtschreibkorrektur), grammatische Analyse (Grammatiküberprüfung), eventuell auch statistische Informationen über den geschriebenen Text (Häufigkeitsanalysen) oder Lexikographie (Thesaurus). Kommt eine Internet-Recherche dazu, erweitert sich der Kreis der Methoden um Informationserschließung und möglicherweise vollautomatische maschinelle Übersetzung.

Aber selbst wenn man keinen Computer benutzt, wird man mit Anwendungen der Computerlinguistik konfrontiert, etwa beim Lesen der halbautomatisch übersetzten Bedienungsanleitung für den neuen Toaster oder beim Telefonat mit der Bank, an dessen Beginn eine freundliche Maschine nach der Kontonummer fragt.

Diese wenigen Beispiele machen deutlich, welche Bedeutung die Computerlinguistik in den letzten Jahren erfahren hat: Sie erschließt Informationsquellen, erleichtert den Umgang mit Maschinen und hilft, Grenzen zwischen verschiedenen Sprachen zu überwinden.

1.1.1 Computerlinguistik: Die Wissenschaft

Gegenstand der Computerlinguistik ist die Verarbeitung natürlicher Sprache (als Abgrenzung zu z. B. Programmiersprachen) auf dem Computer, was sowohl geschriebene Sprache (Text) als auch gesprochene Sprache (engl: *speech*) umfasst. Computerlinguistik ist im Kern und von ihrer Historie her (siehe Unterkapitel 1.2) eine Synthese informatischer und linguistischer Methoden und Kenntnisse.

Diese Charakterisierung ist bewusst sehr allgemein gehalten, um die verschiedenen Auffassungen von „Computerlinguistik“ zu umfassen, die in diesem Buch vereint werden sollen:

- Computerlinguistik als Teildisziplin der Linguistik (wie Psycholinguistik, Soziolinguistik usw.), die sich, in der Regel theoriegeleitet, mit berechnungsrelevanten Aspekten von Sprache und Sprachverarbeitung beschäftigt (vgl. auch den englischen Terminus für Computerlinguistik, *computational linguistics*), unabhängig von ihrer tatsächlichen Realisierung auf dem Computer. Die Entwicklung von Grammatikformalismen ist ein Beispiel für diese Auffassung von Computerlinguistik.
- Computerlinguistik als Disziplin für die Entwicklung linguistik-relevanter Programme und die Verarbeitung linguistischer Daten („Linguistische Datenverarbeitung“). Diese Auffassung hat ihre Wurzeln in den Anfängen der Informatik und hat insbesondere durch die zunehmende Wichtigkeit empirischer Untersuchungen anhand umfangreicher Sprachdatenkorpora (s. Kapitel 4) eine Renaissance erfahren.
- Computerlinguistik als Realisierung natürlichsprachlicher Phänomene auf dem Computer („maschinelle Sprachverarbeitung“, engl: *natural language processing*). Die Untersuchung vieler dieser Phänomene hat eine lange Tradition innerhalb der Sprachphilosophie bzw. der sprachorientierten formalen Logik. Da Sprache als Teil eines kognitiven Systems aufgefasst werden kann, in dem sprachliche Kenntnis und nicht-sprachliches Wissen, Denkprozesse und Handlungsplanung eng miteinander verknüpft sind, sind insbesondere die Künstliche Intelligenz und die Kognitionswissenschaft an der Untersuchung bzw. der Modellierung dieser Phänomene interessiert. Die Computerlinguistik ist daher untrennbar mit den formalen und/oder kognitiven Disziplinen verknüpft.
- Computerlinguistik als praxisorientierte, ingenieursmäßig konzipierte Entwicklung von Sprachsoftware („Sprachtechnologie“).

Diese Liste verschiedener Auffassungen veranschaulicht *prinzipielle* Unterschiede in der Auffassung von Computerlinguistik. Die Computerlinguistik, die in diesem

Buch vorgestellt werden soll, ist als Summe und Synthese ihrer verschiedenen Ausprägungen zu verstehen.

Hierbei bilden vier Bereiche die Eckpfeiler der Computerlinguistik: Die Entwicklung von Methoden, durch die natürlichsprachliche Phänomene operationalisiert werden; der Aufbau und die Verwaltung großer wiederverwendbarer Korpora sprachlicher Daten, die für empirische, Entwicklungs- und Evaluationszwecke genutzt werden können; die Entwicklung realistischer Anwendungen, die die Relevanz der Computerlinguistik für die moderne Informationstechnologie aufzeigen und die gleichzeitig ihren technologischen Fortschritt widerspiegeln; und die Konzeption effektiver Evaluationsmechanismen, durch die der angesprochene Fortschritt objektiviert wird. Zudem ist die Computerlinguistik in fachlichen Grundlagen verankert, die sie zum Teil aus ihren Mutterdisziplinen erbt und zum Teil von weiteren Nachbardisziplinen übernimmt.

1.1.2 Computerlinguistik und ihre Nachbardisziplinen

Von der *Linguistik* übernimmt die Computerlinguistik den Untersuchungsgegenstand und gleichzeitig das Grundinventar linguistischer Termini und Differenzierungen. Die Strukturierung der Methodenbereiche in der Computerlinguistik orientiert sich daher weitestgehend an den etablierten Teilgebieten der Linguistik: Phonologie, Morphologie, Syntax, Semantik und Pragmatik, welche die Schwerpunktebenen der strukturellen Beschreibung natürlichsprachlicher Äußerungen bilden (vgl. etwa Grewendorf, Hamm und Sternefeld 1987).

Die Computerlinguistik ist aber nicht nur ein Abnehmer linguistischer Theorien und Sachverhalte, sondern sie kann auch ein Stimulus für Erkenntnisgewinn und die Erarbeitung neuer Ansätze innerhalb der Linguistik sein.

Ein erfolgreiches Beispiel für die interdisziplinäre Arbeit zwischen Linguistik und Computerlinguistik stellt die Entwicklung der Optimalitätstheorie dar (vgl. Prince und Smolensky 1993). Ursprünglich hervorgegangen aus der Verbindung von Ansätzen neuronaler Netze und Prinzipien der Universalgrammatik, um eine bessere Beschreibung der Phonologie zu ermöglichen, ist die Optimalitätstheorie neben regelorientierten Ansätzen inzwischen zu einem konkurrenzfähigen Modell für die Beschreibung phonologischer Sachverhalte geworden. Darüber hinaus wird sie zunehmend zur Beschreibung von Phänomenen auf anderen Ebenen, z. B. der Morphologie und der Syntax, benutzt.

Die Anwendung und Evaluation linguistischer Theorien ist eine weitere Aufgabe für die Computerlinguistik. Erst die Applikation von Theorien auf real vorkommende Daten liefert einen Aufschluss über deren Korrektheit und Vollständigkeit und kann teilweise sogar für deren Verwendung außerhalb streng theoretisch orientierter Kreise sorgen. Als ein Vertreter sei hier die Implementierung eines Systems zur Strukturanalyse erwähnt, das auf dem Prinzipien- und Parameter-Ansatz beruht (Fong 1991).

Und schließlich sind einige Zweige der Linguistik stärker als andere auf die Bearbeitung von Material durch Computer angewiesen. Die Korpuslinguistik etwa, die sich mit der Erforschung linguistischer Zusammenhänge durch die Betrachtung von Korpora befasst, ist erst durch die Verwendung von Computern in den

letzten Jahren dazu in die Lage versetzt worden, realistisch große Datenmengen mit einer hohen Abdeckung (oft im Größenbereich von Milliarden von Wörtern) zu untersuchen.

Die *Informatik* steuert zur Computerlinguistik im Wesentlichen das Wissen über Datenstrukturen sowie die Verwendung effizienter Verfahren bei. Neben dem offensichtlichen Zusammenhang zwischen der Untersuchung und Realisierung natürlichsprachlicher Systeme und der Informatik (Systemanalyse, Modellierung, Algorithmik, Implementation) spielen aber auch Aspekte der theoretischen Informatik (Berechenbarkeit, Komplexitätstheorie und der Bereich der formalen Sprachen) eine wichtige Rolle.

Aus der *Philosophie* (insbesondere der Sprachphilosophie und Logik) stammen vor allem Aspekte der Frage, wie sich Sprache, Denken und Handeln zueinander in Verbindung setzen lassen; Sprache an sich kann nicht nur als losgelöstes Phänomen betrachtet werden, sondern steht in enger Relation zu außersprachlichen Gegebenheiten, sowohl der Welt als solches und (in einem engeren Sinn von Welt) der Gemeinschaft der Sprecher einer Sprache (Schmidt 1968). Die formale Logik ist eines der zentralen Mittel in der Computerlinguistik zur präzisen Darstellung natürlichsprachlicher Phänomene.

Eine Reihe wichtiger Verfahren (z. B. Such- und Planungsverfahren) verdankt die Computerlinguistik der *Künstlichen Intelligenz*. Sie werden beispielsweise bei der Spracherkennung (Unterkapitel 5.4), der grammatischen Analyse (Unterkapitel 3.5) und der Generierung (Unterkapitel 5.6) eingesetzt. Vor allem für die Semantik (Unterkapitel 3.6) sind die Formalismen zur Darstellung von sprachlichem und nicht-sprachlichem Wissen (Wissensrepräsentation) relevant, die in der Künstlichen Intelligenz entwickelt worden sind (s. auch Unterkapitel 4.6) – ebenso wie Verfahren und Mechanismen, mit denen aus gegebenen Wissensstrukturen weitere Schlüsse (Inferenzen) gezogen werden. Mit der klassischen, symbolischen Künstlichen Intelligenz hat die Computerlinguistik zudem die verbreitete Verwendung zweier höherer Programmiersprachen, LISP und PROLOG, gemeinsam (vgl. auch das Unterkapitel 3.9).

Die Computerlinguistik steht zudem in enger Beziehung zur *Kognitionswissenschaft*. Das lässt sich dadurch erklären, dass die Sprachbeherrschung ein hochspezialisierter Teilbereich der generellen kognitiven Fähigkeiten des Menschen ist und dass sprachliches und nicht-sprachliches Wissen untrennbar miteinander verknüpft sind. Vor diesem Hintergrund erscheint es sinnvoll, bei der Konzeption von Verfahren zur maschinellen Sprachverarbeitung die Eigenschaften menschlicher Sprachverarbeitung und ihrer Beziehung zu allgemeinen Denkprozessen zu betrachten. Bis heute stellt die Fähigkeit zur adäquaten sprachlichen Kommunikation (Turing 1950, siehe auch Unterkapitel 1.2) einen wichtigen Test für die „Intelligenz“ einer Maschine dar, auch wenn der eigentliche Wert solcher Tests umstritten ist (vgl. z. B. Searle 1990).

Zahlreiche theorie- und anwendungsrelevante Facetten der Computerlinguistik fußen stark auf der Grundlage mathematischer bzw. mathematisch-logischer Theorien (Unterkapitel 2.1). Diese werden gegebenenfalls erweitert oder modifiziert, um die Eigenarten natürlicher Sprache adäquat beschreiben zu können. Beispielsweise basiert ein Großteil der semantischen Beschreibung sprachlicher Äußerungen auf der klassischen Prädikatenlogik. Diese zeigt sich jedoch schon

bei der Darstellung des Unterschieds der beiden folgenden einfachen Ausdrücke als unzulänglich.

- (1.1) a) *Ein großer Berg*
b) *Eine große Ameise*

Menschen haben keine Schwierigkeit, eine korrekte Skala für diese beiden Instanzen von *groß* zu finden, während das für eine maschinelle Bearbeitung mit einem Aufwand, etwa mit dem Einsatz von *Fuzzy-Logik* (Zadeh 1965) für die Behandlung der Vagheit des Adjektivs, verbunden ist. Ein weiteres Beispiel zeigt, dass selbst scheinbar widersprüchliche Aussagen manchmal mit Leichtigkeit verstanden werden können:

- (1.2) *Vögel können fliegen.*
Pinguine sind Vögel.
Pinguine können nicht fliegen.

Die alltägliche Annahme hier ist die, dass Vögel *normalerweise* fliegen können, Pinguine hingegen nicht. Um diesen Mechanismus in den Griff zu bekommen, werden oft *Default-Mechanismen* der Künstlichen Intelligenz eingesetzt, die es erlauben, Standardannahmen bei Vorliegen von gegensätzlicher Evidenz zurückzunehmen.

Die formale Beschreibung natürlicher Sprachen steht in einem engen Zusammenhang zum Gebiet der Automatentheorie und formalen Sprachen. Hier werden Repräsentationsmechanismen und Berechnungsmodelle für verschiedene Klassen von Sprachen entwickelt. Die Komplexität einer Sprache determiniert hierbei die Ausdrucksmächtigkeit der zu ihrer Beschreibung notwendigen Repräsentationen. Gleichzeitig wird dadurch auch die Klasse von Maschinen festgelegt, die zur Erkennung und Analyse von Ausdrücken in einer Sprache notwendig sind. Unterkapitel 2.2 führt genauer in diesen Problembereich ein.

Ein weiteres prominentes Teilgebiet der Mathematik, das für Computerlinguisten sehr wichtig ist, ist die Graphentheorie. Dieser Zweig der Mathematik beschäftigt sich mit der Beschreibung von Eigenschaften von Graphen, d.h. von Mengen von Knoten, die durch Kanten verbunden sein können. Graphenartige Repräsentationen sind auch im täglichen Leben oft anzutreffen (z.B. stellt das Liniennetz eines öffentlichen Nahverkehrssystems einen Graphen dar, bei dem die Haltestellen durch Knoten repräsentiert werden können, und die Streckenabschnitte zwischen den Haltestellen Kanten sind). Für die Computerlinguistik ist die Graphentheorie auf zwei Ebenen relevant. Zum einen sind die Objekte für eine ganze Reihe von Beschreibungsmechanismen Graphen, etwa die Merkmalsstrukturen in Unterkapitel 2.3, die in Unterkapitel 4.3 beschriebenen semantischen Hierarchien sowie die in Unterkapitel 4.6 vorgestellten Ontologien und semantischen Netze. Zum anderen spielt die Graphentheorie auch bei der Realisierung von anspruchsvollen Anwendungen für geschriebene und gesprochene Sprache eine herausragende Rolle. Die Einsatzgebiete reichen hier von der Darstellung gesprochener Äußerungen in Form von Wort- oder Phonemgraphen über die Modellierung syntaktischer Analyse als ein Suchproblem in Graphen

bis hin zur Architektur großer Systeme als gerichtete Graphen, die Komponenten und Datenströme beschreiben. Unterkapitel 2.3 befasst sich u.a. mit dieser Problematik.

Neben Logik, Automatentheorie und Graphentheorie spielt die Statistik eine immer größer werdende Rolle für die Computerlinguistik. Diese ist imminent für das Gebiet der automatischen Erkennung gesprochener Sprache, die heutzutage fast ausschließlich mittels stochastischer Automaten betrieben wird (Unterkapitel 5.4). Zusätzlich ist in den letzten Jahren die korpusorientierte Computerlinguistik stark gewachsen, die statistische Aussagen über die tatsächliche Verwendung von Sprache anhand großer Datenmengen extrahiert und Verarbeitungsverfahren zugänglich zu machen versucht (Unterkapitel 4.1, 4.2, 4.5, 5.3). Unterkapitel 2.4 führt genauer in dieses Gebiet ein.

1.1.3 Teilbereiche der Computerlinguistik

Wie viele Disziplinen, hat auch die Computerlinguistik eine theoretisch und eine praktisch ausgerichtete Seite. Die praktische Computerlinguistik ist der im Wesentlichen nach außen sichtbare Anteil: Hier werden neue Anwendungen erforscht und entwickelt, die sich möglicherweise auf dem lokalen Computer anfinden. Die theoretische Computerlinguistik hingegen untersucht die einer maschinellen Verarbeitung zugrundeliegenden Strukturen im Hinblick auf prinzipielle Fragestellungen wie deren Berechenbarkeit, Adäquatheit und Erlernbarkeit. Die Relevanz beider Aspekte wird in den folgenden Abschnitten erläutert.

Praktische Computerlinguistik

Entscheidende Fragen im Bereich der praktischen Computerlinguistik sind die folgenden:

1. Wie konstruiert man ein Softwaresystem zur Verarbeitung natürlicher Sprache?
2. Welche Formalismen scheinen relevant?
3. Welcher Gegenstandsbereich wird modelliert?
4. Welche interessanten einzelsprachlichen oder anwendungsbezogenen Eigenheiten sollen modelliert werden?
5. Was ist das globale Ziel der Entwicklung?

Das Hauptziel besteht somit darin, (sprachliches) Wissen erfolgreich auf einer Maschine zu modellieren und relevante praktische Probleme zu lösen, z.B. die Übersetzung eines Satzes vom Koreanischen ins Englische oder die Erkennung und Analyse einer telefonischen Pizza-Bestellung. Auf dem Weg zu diesem Ziel sind zahlreiche Aufgaben zu erfüllen, von denen einige den Kern der praktischen Computerlinguistik bilden:

- Die Entwicklung von Formalismen, die dazu genutzt werden können, bestimmte Aspekte natürlicher Sprache zu modellieren. Derartige Formalismen finden sich auf allen Ebenen der Beschreibung natürlicher Sprache, mit unterschiedlicher Ausdrucksmächtigkeit und Zielsetzung. Der Einsatz eines Formalismus, der unabhängig von einer bestimmten Sprache deklarativ die Modellierung sprachlicher Gegebenheiten erlaubt, ist von unschätzbarem Vorteil und hat konsequenterweise die direkte Implementierung von sprachverarbeitenden Algorithmen für die Behandlung bestimmter Phänomene in einer bestimmten Sprache weitgehend verdrängt.
- Die Bereitstellung von Wissen über individuelle Sprachen bzw. bestimmte Aspekte einer Sprache. Dazu gehört neben der Lexikographie (Unterkapitel 5.2) vor allem die grammatische Beschreibung einzelner Sprachen (normalerweise noch weiter eingeschränkt auf bestimmte Anwendungszusammenhänge oder Verwendungsformen). Ein wichtiges Teilgebiet ist die Beschäftigung mit realen Sprachdaten (d.h. die Sammlung, Aufbereitung und Verwaltung von Texten und Sprachaufnahmen, Unterkapitel 4.1–4.5). Die Menge und Verfügbarkeit solcher computerlinguistischer Ressourcen nimmt ständig zu, insbesondere deswegen, da sich die statistischen Eigenschaften bestimmter Phänomene anhand großer Datenmengen besser untersuchen lassen.
- Die Entwicklung von Algorithmen und Methoden zur Bearbeitung natürlichsprachlicher Äußerungen. Die Aufgabenfelder reichen hier von der Erkennung gesprochener Sprache über den Parserbau bis hin zum Design von Dialogsystemen für spezielle Anwendungen (vgl. die Unterkapitel 3.5, 5.4, und 5.5).
- Die Evaluation natürlichsprachlicher Systeme. Um die Performanz und Bandbreite eines Algorithmus oder Systems zu bewerten, reicht es normalerweise nicht aus, einige wenige Beispiele zu verarbeiten. Vielmehr ist es das Ziel, real vorkommende Daten in hinreichender Menge zu untersuchen. Dies gilt uneingeschränkt für Systeme, die auf einer statistischen Modellierung beruhen; aber auch für rein symbolische Verfahren werden Evaluierungen immer wichtiger. Kapitel 6 führt genauer in die Verfahrensweisen ein.

Ein Beispiel für ein Anwendungssystem, das hier prototypisch für den Einsatz praktischer Computerlinguistik genannt werden soll, ist **SmartWeb** (Reithinger, Herzog und Blocher 2007). Dies ist ein multimodales Zugangssystem zum *semantic web*, einem Ausschnitt des Internets, dessen Inhalte durch Metainformationen so angereichert sind, dass Korrelationen einfach hergestellt werden können. Für den Benutzer stellt sich SmartWeb schlicht als eine Applikation auf dem Mobiltelefon dar, die bei einigen täglichen Verrichtungen helfen kann, etwa bei der Auswahl eines Restaurants für den Abend und der Planung einer Autoroute dorthin mit einem kurzen Zwischenstopp an einer Tankstelle. Die

zugrundeliegenden Informationen sind sämtlich im Internet vorhanden; das Auffinden und Verknüpfen der Daten zu einem kohärenten Plan jedoch ist manuell mit einiger Mühe verbunden.

SmartWeb benutzt bereits vorhandene semantisch annotierte Informationsquellen direkt. Um den Zugang zu konventionellen Web-Seiten zu ermöglichen, wurden Verfahren entwickelt, deren Inhalt zumindest in Grenzen automatisch zu verstehen und maschinell zu annotieren.

Zur Realisierung eines solch umfangreichen Projekts sind nicht nur theoretische Einsichten der Computerlinguistik erforderlich; daneben müssen nahezu alle Teilgebiete der praktischen Computerlinguistik herangezogen werden.

Zunächst gilt es, gesprochene Sprache zu erkennen; für die hier angesprochene Anwendung wird das noch kompliziert durch die Vielzahl an Namen (Straßen, Orte, Restaurants usw.), für die das Spracherkennungssystem nicht vorher explizit vorbereitet werden kann. Außerdem kann die sprachliche Eingabe durch andere Modalitäten unterstützt werden, etwa durch Gesten oder über die Tastatur. Diese multimodalen Eingabeäußerungen müssen auf multiplen Ebenen analysiert werden: Syntaktisch, semantisch, und im Hinblick auf ihre Funktion innerhalb des Dialogkontextes. Das Ziel des Benutzers muss erschlossen werden, um die adäquaten Daten aus dem Semantic Web abzurufen. Und schließlich ist es erforderlich, die Resultate multimodal passend aufzubereiten, sei es als Text, in Form einer Landkarte, als Bild, Video oder Ausgabe über einen Lautsprecher.

Über die Entwicklung der Formalismen und Verarbeitungsmechanismen für einzelne Teilbereiche einer Gesamtanalyse hinaus muss allerdings auch dafür gesorgt werden, dass alle Einzelbausteine korrekt und effizient zusammenarbeiten können. Hier werden dann Fragen der Architektur von großen natürlichsprachlichen Systemen und softwaretechnische Aspekte der Integration von Komponenten sowie deren Kommunikation untereinander relevant.

1.1.4 Theoretische Computerlinguistik

Innerhalb der theoretischen Computerlinguistik geht es um die Frage, wie natürliche Sprache formalisiert und maschinell verarbeitet werden kann, ohne dass der Blickwinkel durch die Notwendigkeit, ein tatsächlich funktionierendes System bauen zu müssen, eingeschränkt wird. Abhängig vom tatsächlichen Fachgebiet sind Logik, formale Linguistik und Compilerbau wichtige Grundlagen für erfolgreiche Forschung, während Detailwissen um anwendungsrelevante Aspekte nicht zentral erscheint.

Formalismen spielen auch hier eine große Rolle, allerdings weniger unter dem Blickwinkel, Grammatiken mit einer hohen Abdeckung für eine konkrete Sprache anzufertigen. Vielmehr stehen prinzipielle Fragen wie die Eignung eines Formalismus zur Beschreibung verschiedener Phänomene oder die Komplexität einer Berechnung mittels eines Formalismus im Mittelpunkt. Wichtige Fragestellungen sind etwa:

- Welche Komplexität weist natürliche Sprache an sich auf, und inwieweit kann diese Komplexität durch heutzutage verfügbare Maschinen effektiv bewältigt werden? (vgl. Unterkapitel 2.2)

- Welche Eigenschaften muss ein Formalismus aufweisen, um relevante Aspekte natürlicher Sprache angemessen repräsentieren zu können? Diese Frage stellt sich ebenenübergreifend, so dass zum Teil unterschiedliche Formalismen zur Darstellung von Phonetik, Phonologie, Morphologie, Syntax, Semantik und Pragmatik entwickelt werden. Dies wirft wiederum die Frage auf, bis zu welchem Grade die Repräsentation ebenenübergreifend stattfinden kann, und welche Vor- und ggfs. Nachteile dies mit sich bringt.

Als ein Beispiel für die Forschung in der theoretischen Computerlinguistik sei hier die adäquate Modellierung syntaktischer Strukturen für natürlichsprachliche Äußerungen genannt. Beginnend mit Chomsky (1959) werden verschiedene Komplexitätsklassen formaler Sprachen unterschieden (siehe Unterkapitel 2.2). Diese Klassen entsprechen unterschiedlich komplexen Methoden zur Erkennung und Strukturanalyse. Gemeinhin wird angenommen, natürliche Sprachen seien zwischen den kontextfreien und kontextsensitiven Sprachen angesiedelt; sie sind „schwach kontextsensitiv“. Allerdings sind die Phänomene, die es notwendig machen, über den kontextfreien Rahmen hinauszugehen, eher selten (vgl. Sampson 1983, Shieber 1985). Ein wesentliches Motiv für die Entwicklung komplexer, merkmalsbasierter Formalismen ist denn auch weniger deren prinzipielle theoretische Notwendigkeit, sondern vielmehr ein stärkeres Bestreben nach der adäquaten Beschreibung natürlichsprachlicher Phänomene. Wichtige linguistische Merkmale (wie Kongruenz, Koreferenz oder Spuren) lassen sich kontextfrei analysieren, allerdings verliert die Modellierung an Allgemeingültigkeit dadurch, dass nicht über die Werte bestimmter Merkmale (Kasus etc.) abstrahiert werden kann. Auf der anderen Seite besteht die Gefahr, durch einen zu mächtigen Formalismus Effizienz (und manchmal sogar Berechenbarkeit) einzubüßen. Daraus wird innerhalb der theoretischen Computerlinguistik nach Wegen gesucht, komplexe Beschreibungsformalismen zu entwickeln, die gleichzeitig handhabbar und eingängig sind. Im Laufe der Zeit sind zahlreiche Vertreter solcher Modelle entstanden, die in der Folge auch innerhalb der praktischen Computerlinguistik (und zuweilen in kommerziellen Anwendungen) populär geworden sind (Lexical Functional Grammar (Bresnan 1982), Head Driven Phrase Structure Grammar (Pollard und Sag 1987), und Tree Adjoining Grammar (Joshi 1985), um nur einige Beispiele zu nennen).

Ein immer wichtiger werdender Anteil der theoretischen CL beschäftigt sich mit der Frage, ob und wie eine signifikante Untermenge sprachlicher Konstrukte und Konzepte automatisch erlernt werden kann¹. Dies hängt neben der Verfügbarkeit hochgradig leistungsfähiger Computer vor allem mit der ständig wachsenden Menge an Text zusammen, die leicht zugänglich ist.

Das initiale Problem ist das der Umwandlung von natürlichsprachlichen Eingaben in interne Repräsentationen oder direkt in andere natürlichsprachliche Ausgaben. Dies kann sich auf mehreren Ebenen abspielen: z. B. kann eine morphologische Analyse oder die Zuweisung von Wortarten (*Part-of-Speech Tagging*) als ein Klassifikationsproblem verstanden werden, bei dem jedes Wort der Ein-

¹Die Erlernbarkeit durch Maschinen steht hier im Vordergrund, nicht die Untersuchung der Mechanismen, die es Menschen erlauben, eine Sprache zu lernen (Spracherwerb).

gabe zu einer von mehreren Dutzend unterschiedlichen Kategorien zugewiesen wird. Im Rahmen der syntaktischen Analyse kann es als eine Transformation von einer linearen Struktur (der Eingabeäußerung) in eine Baum- oder Graphenförmige Struktur (der Analyse) behandelt werden. Und schließlich kann man es in der Maschinellen Übersetzung als eine Transformation und Umdeutung von einer linearen Eingabe in eine (anderssprachige) lineare Ausgabe ansehen.

Gängige Methoden zum Erlernen solcher Umwandlungen sind normalerweise sehr stark an statistische Prozesse gebunden (z. B. an stochastische Automaten für Morphologie, Markov-Modelle für Wortartenzuweisung, stochastische Grammatiken für Syntaxanalyse, oder *noisy channel models* für Übersetzung). Diese beruhen darauf, eine Menge von manuell mit dem gewünschten Resultat annotierten prototypischen Eingaben als Trainingsmaterial zu benutzen. Statistische Lernalgorithmen konsumieren das Material und produzieren Modelle, die von den einzelnen Eingaben abstrahieren und Generalisierungen über die vorkommenden Phänomene darstellen. Laufzeitkomponenten benutzen diese Modelle dann, um bisher ungesehene Eingaben zu analysieren und die gewünschten Resultate herzustellen. Kritische Fragestellungen im Umgang mit Lernalgorithmen sind u.a.:

- Wie gut ist der Lernmechanismus? Im Vordergrund steht hierbei natürlich, welchen Erfolg ein System bei der Analyse von unbekannten Eingaben hat: Wieviele Eingaben können überhaupt verarbeitet werden, wieviele Antworten werden erzeugt, und wieviele davon sind richtig (vgl. Kapitel 6)?
- Wie schnell ist der Mechanismus? Für diese Frage sind zunächst Aspekte der Komplexitätstheorie relevant, um festzustellen, ob ein Lernalgorithmus oder die Anwendung der generierten Modelle prinzipiell möglich scheint. Darüber hinaus ist es interessant abzuschätzen, welche Menge an Trainingseingaben notwendig ist, um ein akzeptables Modell zu erstellen (z. B., wenn man sich Gedanken über sog. *low density languages* macht, Sprachen, für die nur ein kleines Korpus verfügbar ist). Dies ist die Frage nach der Generalisierungsfähigkeit des Algorithmus, nach der Balance zwischen sturem Auswendiglernen von Trainingseingaben und der Extraktion von abstrakten Eigenschaften aller Trainingseingaben. Und schließlich ist wichtig zu untersuchen, wie schnell potentielle neue Eingaben in das Wissen des Mechanismus integriert werden können. Kann z. B. eine gerade analysierte und verifizierte Äußerung dazu benutzt werden, die Qualität des benutzten Modells inkrementell zu verbessern?
- Wie adäquat ist der Mechanismus? Hier sind (normalerweise zu einem kleineren Anteil) philosophische Aspekte zu betrachten, etwa der Art, ob der automatische Lernalgorithmus ein ähnliches Fehlerprofil wie Menschen aufweist. Wichtiger erscheint eine Abschätzung darüber, ob die untersuchte Methode relativ einfach auf eine neue Domäne, eine andere Sprache, oder ein anderes Teilgebiet linguistischer Phänomene angewendet werden kann.

Die angedeuteten Fragestellungen deuten darauf hin, dass das (theoretische) Feld der Lernalgorithmen eng mit dem Vorhandensein von Trainings- und Testkorpo-

ra zusammenhängt. So ist es kein Zufall, dass in den letzten Jahren zahlreiche regierungsfinanzierte Projekte zur Sammlung und Annotierung von Sprachdaten initiiert wurden. Diese umfassen zahlreiche Sprachen, Anwendungsdomänen und Modalitäten. Der diesen Anstrengungen innewohnende Aufwand hat zudem zu einem stärkeren Fokus auf unüberwachte Lernalgorithmen geführt, Algorithmen, die kein annotiertes Trainingskorpus benötigen, sondern Regularitäten ausschließlich basierend auf Eingabeäußerungen ableiten. Manchmal ist dies schon ausreichend, etwa im Bereich der Disambiguierung von Wortbedeutungen; meist werden die gefundenen Regularitäten allerdings einem weiteren, manuellen Analyseschritt unterworfen, um deren Korrektheit sicherzustellen und ihnen eine symbolische Bedeutung zuzuordnen.

Ein relativ neuer Bereich der Forschung ist der der hybriden Systeme. In der vorangegangenen Diskussion war davon ausgegangen, dass ausschließlich extensional gearbeitet wird: Paare von Eingabeäußerungen und den mit ihnen assoziierten korrekten Antworten wurden dazu benutzt, Regularitäten zu finden. Im Gegensatz dazu sind konventionelle Grammatiken stark intensional orientiert, in dem man direkt Abstraktionen formuliert, basierend auf der Intuition der Grammatikschreiber oder einer subjektiven Analyse eines Beispielkorpus. Die Proponenten beider Ansätze haben gewichtige Argumente für die Überlegenheit der eigenen Sichtweise. Intensionale Grammatikschreiber argumentieren, dass mit einer Regel eine ganze Klasse von Äußerungen abgedeckt werden kann, und dass sich feine Unterschiede in Strukturen einfach handhaben lassen, während extensionale Statistiker hervorheben, dass stochastische Methoden stärker an der realen Benutzung von Sprache orientiert sind, und dass die Verfügbarkeit von Sprachmaterial die Anwendung auf unterschiedliche Domänen und Sprachen enorm erleichtert. In den letzten Jahren haben sich diese beiden Schulen aneinander angenähert, insbesondere im Bereich der Maschinellen Übersetzung (s. z. B. Charniak, Knight und Yamada 2003). Statistische Methoden werden benutzt, um Übersetzungsmuster im Trainingstext zu finden, während linguistisch orientierte Strukturregeln die Validität von bestimmten Satzmustern hervorheben.

1.1.5 Wissensbereiche

Die Wissensbereiche innerhalb der Computerlinguistik sind weitgehend an den von der Linguistik angenommenen Beschreibungsebenen natürlicher Sprache orientiert. Dies erscheint aus methodischer Sicht zunächst unvermeidlich und sinnvoll, auch wenn aus theoretischen oder praktischen Erwägungen heraus diese Einteilung häufig aufgehoben wird.²

Generelles Paradigma der Computerlinguistik sollte das Streben nach Erkenntnissen über bedeutungsdefinierende und bedeutungsunterscheidende Merkmale sein. Insofern sind die Resultate der theoretischen Linguistik von weit stärkerer

²Etwa bei der Entwicklung von Übersetzungssystemen, die ausschließlich statistische Information nutzen (Brown et al. 1990). Hier wird versucht, ein zusammenhängendes Modell für alle relevanten Verarbeitungsschritte zu berechnen, so dass auf den Einfluss einzelner Ebenen nicht mehr geachtet werden muss.

Bedeutung für die Computerlinguistik als der Bereich der rein deskriptiven Linguistik, von dem überwiegend nur die Bereitstellung von initialen Daten über Sprachen von Interesse ist.

Eine vertikale Einteilung der Computerlinguistik umfasst zumindest die folgenden fünf Bereiche:

- **Phonetik und Phonologie** (Unterkapitel 3.1): Sie untersuchen die artikulatorischen Merkmale sowie die Lautstruktur natürlicher Sprachen und kommen in der Computerlinguistik vor allem im Bereich der Erkennung und Produktion *gesprochener* Sprache vor. Ziel ist u.a. zu modellieren, welche Segmente ein Wort enthält und wie sich deren Struktur auf die Aussprache auswirkt, z. B. wenn ein im Prinzip stimmhafter Konsonant am Wortende stummlos wird (Auslautverhärtung):

(1.3) *Dieb* vs. *Diebe*
 /Diep/ /Diebe/

- Die **Morphologie** (Unterkapitel 3.3) beschreibt die Bildung und Struktur von Wörtern. Untersucht wird hier, welche lexikalische Wurzel einzelne Wörter haben, welche Prozesse für die unterschiedlichen Erscheinungsformen an der Oberfläche verantwortlich sind, und wie diese Oberflächenmodifikationen die Verwendung und Bedeutung des Wortes verändern. Die Morphologie ist durch eine vorwiegend anglozentrische Forschung innerhalb der Computerlinguistik lange Zeit unterrepräsentiert gewesen; erst mit der Untersuchung stärker flektierender Sprachen gewann sie an Gewicht. Eine morphologische Analyse des Deutschen muss etwa erkennen können, dass das Suffix *-e* im folgenden Beispiel eine Pluralmarkierung darstellt:

(1.4) *Dieb-e*
Dieb-pl
 „Mehr als ein Dieb“

- In den Bereich der **Syntax** (Unterkapitel 3.5) fällt alles, was mit der Strukturbildung von Sätzen zu tun hat. Sie ist die traditionell am stärksten vertretene Teildisziplin der Computerlinguistik. Eine strukturelle Analyse von Äußerungen ist unverzichtbar für die erfolgreiche Erkennung von Grammatikalität und eine darauf folgende Bedeutungserschließung. So muss im folgenden Gegensatz nicht nur erkannt werden, dass (1.5b) ungrammatisch ist, auch der Zusammenhang zwischen den einzelnen Wörtern und die daraus gebildete Struktur sind relevant (ungrammatische Sequenzen werden mit einem Stern „**“ eingeleitet):

(1.5) a. *Der gewitzte Dieb stahl das Geld.*
 b. **Der Dieb gewitzte stahl das Geld.*

- Die **Semantik** (Unterkapitel 3.6) befasst sich mit der Bedeutung sprachlicher Einheiten. Dabei wird sowohl versucht, die Aspekte der Bedeutung

von lexikalischen Einheiten zu beschreiben (in der lexikalischen Semantik), als auch die Bedeutungszusammenhänge von größeren strukturellen Einheiten zu repräsentieren. Z. B. kann beiden Sätzen in Beispiel (1.6) dieselbe prinzipielle Bedeutungsstruktur zugewiesen werden, obwohl die Wortstellung unterschiedlich ist:

- (1.6) a. *Die Polizei beschlagnahmte das Diebesgut.*
- b. *Das Diebesgut beschlagnahmte die Polizei.*

- Die **Pragmatik** (Unterkapitel 3.7) untersucht sprachliche Ereignisse daraufhin, welchen Zweck eine Äußerung in der Welt hat. Die Frage

- (1.7) *Ist das Fenster auf?*

mag schlicht eine einfache Informationsfrage sein. Weitauß wahrscheinlicher ist jedoch, dass der fragenden Person kalt ist, oder dass es zieht. In diesem Zusammenhang muss die Frage dann als Aufforderung verstanden werden, das betreffende Fenster doch bitte zu schließen. Die Abschnitte in Unterkapitel 3.7 befassen sich unter anderem mit der automatischen Bestimmung des Antezedens einer Anapher wie in *Die Katze₁ schnurrt. Sie₁ hat Hunger.* (Abschnitt 3.7.2), die Äußerungen innenwohnenden impliziten Annahmen (Präsuppositionen, Abschnitt 3.7.3) und der Frage, welche Annahmen eine Maschine über einen Benutzer machen kann und sollte (Benutzermodellierung, Abschnitt 3.7.4). Auch der Bereich der Konstruktion sprachlicher Oberflächenrepräsentationen durch eine Maschine (Generierung, Unterkapitel 5.6) ist pragmatisch motiviert.

Zusätzlich lassen sich einige Bereiche erfassen, die ebenenübergreifend von Relevanz sind: Ein Beispiel hierfür ist die Prosodie, deren Einfluss auf praktisch alle oben genannten Gebiete nachgewiesen werden kann.

Neben dieser vertikalen Einteilung der hier aufgeführten Wissensbereiche lassen sich zwei weitere, mehr horizontale Unterscheidungskriterien herausarbeiten:

- Es muss zwischen der Repräsentation von Wissen und der Modellierung der Prozesse, die dieses Wissen benutzen, um ein bestimmtes Phänomen zu untersuchen, unterschieden werden. Beide sind gleichermaßen notwendig und wichtig, um erfolgreich natürliche Sprache zu erforschen und funktionierende Systeme zu deren Verarbeitung zu konstruieren.
- Alle hier genannten Wissensebenen spielen sowohl bei der Analyse als auch der Produktion natürlicher Sprache eine Rolle. So ist beispielsweise die *Analyse* der syntaktischen Struktur einer Äußerung der Kernbereich des Parsing (vgl. Unterkapitel 3.5), während die *Erzeugung* einer Oberflächenstruktur ausgehend von einer syntaktischen Beschreibung als Generierung im engeren Sinne bezeichnet wird (vgl. Unterkapitel 5.6).

1.1.6 Industrielle Anwendungen

Ergebnisse aus der Computerlinguistik-Forschung haben bereits Einzug gehalten in einen weiten Bereich industrieller Anwendungen. Das Paradebeispiel hier ist *Google*: Die Suchanfragen nach Webseiten werden z. B. normalerweise einer morphologischen Analyse unterzogen, um die Menge an potentiell relevanten Seiten zu erhöhen. Findet man eine Seite in einer Sprache, die man nicht versteht, kann Google diese übersetzen. Eine andere Anwendung, *Google News*, benutzt unüberwachte Clustering-Methoden und Textzusammenfassung, um einen Überblick über die augenblickliche Nachrichtenlage zu ermöglichen.

Das Internet enthält eine sehr große Menge an Information (vgl. Unterkapitel 4.7). Das bedeutet aber nicht, dass diese Information immer leicht zugänglich ist. Im Gegenteil, sie ist hochgradig unstrukturiert, so dass ein direkter Zugang zu relevanten Daten unwahrscheinlich ist. Um einen Zugriff auf Information für einen weiten Kreis von Benutzern verfügbar zu machen, oder bestimmten Aufgaben in einer einfacheren, natürlicheren Art und Weise gerecht zu werden, scheinen natürlichsprachliche Schnittstellen sinnvoll. Eine Anfrage wie „*Wie kann ich am billigsten nach Amerika telefonieren*“ ist in vielen Fällen einfacher zu stellen als die ungefähr äquivalente Form „+telefon +amerika +preis +vergleich“. Folglich arbeitet eine beachtliche Anzahl von Firmen an der Frage, wie natürlichsprachliche Anfragen dazu benutzt werden können, Information aus einer Menge von Dokumenten zu extrahieren. Ein solches Verfahren ist insbesondere dann extrem anspruchsvoll, wenn die Eingabe nicht mehr oder weniger direkt auf eine syntaktisch äquivalente Datenbankanfrage abgebildet werden kann, sondern versucht werden muss, Teile der Bedeutung von Dokumenten zu modellieren, so dass auch eine Frage, die nicht aus relevanten Kennwörtern besteht, Aussicht auf erfolgreiche Beantwortung haben kann (vgl. Unterkapitel 5.3).

Als zweites Beispiel für den immer wichtiger werdenden Einfluss der natürlichsprachlichen Verarbeitung sei die Einführung von Dialoganwendungen genannt (vgl. Unterkapitel 5.5). Diese können einen relativ einfachen Zugang zu komplexen Systemen realisieren, bei denen eine Reihe von Informationen vom Benutzer zum System geleitet werden müssen. Als Paradebeispiel hierfür gilt normalerweise die Bestellung eines Bahn- oder Flugtickets, aber auch die Interaktion mit der eigenen Bank. Während hier Telefonanwendungen, die auf dem Eingeben numerischer oder alphabetischer Daten mit Hilfe der Tastatur des Telefons beruhen, inzwischen weit verbreitet gefunden haben, sind natürlichsprachliche Anwendungen, innerhalb derer der Benutzer verbal mit einer Maschine kommuniziert, noch selten. Allerdings existieren bereits seit einigen Jahren beachtenswerte prototypische Systeme hierzu (vgl. Unterkapitel 5.5).

Übersetzungssysteme erlangen stärkere Marktdurchdringung. Dies ist nicht nur motiviert durch den Wunsch von Endbenutzern, Web-Seiten in anderen Sprachen lesen zu können. Der Trend zur Globalisierung zwingt Anbieter von Produkten und Maschinen, Information in mehreren Sprachen anzubieten (z. B. in der Form von Gebrauchsanweisungen) oder dazu in der Lage zu sein, solche zu konsumieren (in der Form von Anfragen, Serviceanforderungen usw.). Geopolitische Realitäten zwingen insbesondere Regierungen dazu, in Übersetzungs-

systeme zu investieren, um Personal dazu in die Lage zu versetzen, erfolgreich mit Personen und Gruppen in anderen Ländern zu kommunizieren. Dies hat in den letzten Jahren zur verstärkten Forschung und Produktentwicklung von Übersetzungssystemen vor allem für nichteuropäische Sprachen geführt.

Schließlich sei auch angemerkt, dass eine Reihe von Geschäftsprozessen bereits durch die CL unterstützt sind. Z. B. ist es wahrscheinlich, dass ein Bewerbungsbrief und Lebenslauf, der an eine sehr große Firma geschickt wird, zunächst von einer Maschine untersucht wird, um relevante Qualifikationen zu extrahieren und möglicherweise die am besten passende Stelle zu ermitteln. Auch werden die in einem Konzern eingehenden Briefe vielfach gemäß ihres Inhaltes klassifiziert, um die richtige Abteilung in einer großen Organisation zu identifizieren.

Die hier zitierten Schwerpunkte der Anwendung computerlinguistischen Wissens in der Industrie bedeuten, dass vor allem drei Bereiche stark nachgefragt sind:

- Die Verbindung von Sprachkenntnissen mit Computerlinguistik-Wissen, insbesondere im Bereich der Lexikographie und Korpusbearbeitung. Die Erweiterung einer Anwendung auf eine neue Sprache verlangt zunächst nach einem Muttersprachler für diese Sprache. Aus praktischen Erwägungen heraus ist es von unschätzbarem Vorteil, wenn dieser darüberhinaus über die notwendigen Grundlagen zur effektiven Modellierung sprachlichen Wissens verfügt. Dazu gehören neben dem prinzipiellen Aufbau eines Lexikons und den Eigenschaften von Einträgen (Argumentstrukturen, lexikalische Semantik) auch Fertigkeiten im Bereich des Grammatikentwurfs (Linguistik und Formalismen) und die Fähigkeit, Korpora aufzubauen oder zusammenzustellen und daraus relevante linguistische Fakten abzuleiten.
- Dialogsystembau. Zum gegenwärtigen Zeitpunkt sind kommerzielle Dialogsysteme noch meist einfach strukturiert. Der Ablauf eines Dialogs ist weitgehend vorher festgelegt, ohne dass der Benutzer die Möglichkeit hat, großen Einfluss auf dessen Inhalte und Strukturen zu nehmen. Es ist folglich umso wichtiger, dass das Design eines Dialogs umfassend und korrekt ist, und auf ungewöhnliche Phänomene vorbereitet ist. Zur Modellierung von Anwendungen werden eine Reihe von Designtools benutzt, deren prinzipielle Möglichkeiten und Begrenzungen bekannt sein müssen. Ein Computerlinguist bringt hier sein Wissen um Dialogstrukturierung und die genannten linguistischen Teilgebiete Syntax, Semantik und Pragmatik ein.
- Erfahrung in der Entwicklung natürlichsprachlicher Systeme. Die genaue Ausrichtung hängt selbstverständlich von dem jeweiligen Anwendungszweck ab, doch lässt sich feststellen, dass ein umfassendes Querschnittswissen für die Entwicklung der meisten Systeme unumgänglich ist. Um nur ein Beispiel zu nennen: Für die erfolgreiche Entwicklung eines Systems zur Informationsrecherche im Internet sind zumindest die Teilbereiche Morphologie und Syntax (um Anfragen zu analysieren), Semantik (vornehmlich zur Modellierung des Wissens in Dokumenten), und statistische

Computerlinguistik (erneut zur Inhaltsmodellierung und Abschätzung von Relevanzfragen) wichtig.

In der Zukunft wird sich die Interaktion von Konsumenten mit Produkten und die Handhabung von Information weiterhin stark verändern. Es ist abzusehen, dass immer mehr Funktionen unter Zuhilfenahme persönlicher Assistenten erlebt werden. Insbesondere die Möglichkeit zur Eingabe natürlich gesprochener Sprache sowie die immer besser werdenden Systeme zur Informationsextraktion, Plansynthese und dynamischer Textzusammenfassung bedeuten, dass das Internet immer weniger als eine passive Informationsquelle angesehen werden muss, sondern dass man quasi mit ihm kooperiert. Während man heute relativ einfach nach günstigen Flugpreisen nach Miami suchen kann, könnte die Reiseplanung in Zukunft beinhalten, dass der persönliche Assistent Alternativen vorschlägt (*„Du bist letztes Jahr schon nach Miami geflogen. Wie wäre es mit Jamaika? Ähnliches Klima, aber wesentlich exotischer.“*), Nachrichten zusammenfasst (*„Das Hotel ist in einer Gegend mit hoher Kriminalität. Ich weiß, es ist billig, aber vielleicht solltest Du doch besser dieses hier nehmen.“*), und komplexe Prozesse übernimmt (*„Ok, soll ich das jetzt buchen?“*).

Auch Haushaltsgeräte könnten mit Sprachtechnologie ausgerüstet werden (dann kann der Kühlschrank mitteilen, was er enthält, und einen Einkaufszettel vorschlagen). Das Hauptproblem hier könnte das Überangebot an sprachlicher Kommunikation sein, und folglich könnte die Aggregation und Priorisierung von Information im Vordergrund stehen. Natürlichsprachliche Zugangssysteme zu Fahrzeugen existieren bereits rudimentär, hauptsächlich in Form von Kommandosystemen und in niedriger Zahl als sogenannte Sprachdialogsysteme. Auch in diesem Bereich kann erwartet werden, dass die Bandbreite an relevanter Information, die mit Hilfe natürlicher Sprache abgefragt und kontrolliert werden kann, stetig wächst. Eine kluge Anwendung von Computerlinguistik kann hier dazu führen, dass die Ergonomie solch komplexer Systeme stark verbessert wird.

Auch in der Geschäftswelt wird sich der Einfluss der CL erhöhen. Während ein Teil der Kommunikation zwischen Unternehmen stark formalisiert ist (Rechnungen usw.) und mit relativ einfachen Mechanismen gehandhabt werden kann, so ist ein weiterer großer Teil natürlichsprachlich (Anfragen, Beschwerden, Notizen, Memos usw.) und erfordert computerlinguistische Methoden, um wenigstens partiell automatisch behandelt werden zu können.

1.1.7 Berufsfelder für Computerlinguisten

Die Computerlinguistik/Sprachtechnologie eröffnet vielfältige Anwendungsbereiche innerhalb einer modernen Informationsgesellschaft – das Kapitel 5 stellt die wichtigsten Anwendungen vor. Es ist abzusehen, dass die Verarbeitung gesprochener Sprache für die Interaktion mit Computern und für die Steuerung intelligenter Geräte an Bedeutung gewinnen wird, und dass die Verarbeitung von Texten als allgegenwärtigen Trägern von Information ohne texttechnologische Anteile (z. B. Klassifikation, Retrieval, Übersetzung, Zusammenfassung) kaum denkbar sein wird. Schon jetzt verfügen weltweit operierende Softwareanbieter

in der Regel über eigene Sprachtechnologie-Forschungslabore, während die Zahl eigenständiger Computerlinguistik-Firmen stetig zunimmt (allein für den Bereich der maschinellen und computergestützten Übersetzung listet Hutchins und Hartmann (2002) mehr als 160 Firmen auf).

Neben diesem Bereich der Computerlinguistiksoftware-*Entwicklung* finden Computerlinguisten und Computerlinguistinnen ihre Berufsfelder vor allem im Rahmen des *Einsatzes* bzw. der *Verwendung* sprachtechnologischer Software und Ressourcen (in Verlagen, Übersetzungsbüros, Verwaltungen etc.) und, insbesondere langfristig gesehen, auch in deren *Wartung/Support* und *Vertrieb* (zu detaillierteren Informationen siehe auch <http://berufenet.arbeitsamt.de> mit dem Suchwort „Computerlinguistik“).

1.1.8 Literaturhinweise

Es existieren mittlerweile eine Reihe von Einführungen und Handbüchern zur Computerlinguistik und Sprachtechnologie. Der „Klassiker“ ist in dieser Hinsicht Allen (1995), das 1987 zuerst erschienen ist. Neuere englischsprachige Alternativen hierzu sind insbesondere Jurafsky und Martin (2009) sowie Mitkov (2003). Die erste umfassende und gute Einführung in die statistische Computerlinguistik stellt Manning und Schütze (2003) dar. Weiterhin sind Cole et al. (1997), Dale et al. (2000) sowie Hausser (2001) (das auch in deutscher Sprache als Hausser 2000 vorliegt) zu nennen.

Eine sehr grundlegende deutschsprachige Einführung ist Schmitz (1992). Die für die (Computer)linguistik notwendigen Statistik-Kenntnisse vermittelt anschaulich und fundiert Gries (2008). Der Sammelband Batori und Lenders (1989) dokumentiert den Kenntnisstand in der Computerlinguistik aus den 80er Jahren, ist aber immer noch teilweise lesenswert. Heyer et al. (2006) führen in praxisorientierte Aspekte der Textverarbeitung ein, während Lobin und Lemnitzer (2004b) eine Mischung aus Grundlagen, Methoden und Anwendungen in der Texttechnologie präsentiert. Carstensen (2009b) bietet einen Überblick über die komplexen Anwendungen in der Computerlinguistik.

Görz et al. (2003) ist eine allgemeine Einführung in die Künstliche Intelligenz, die auch einen Teil über Sprachverarbeitung enthält. Für Darstellungen von aktuellen Entwicklungen sei auf die Zeitschrift *Computational Linguistics* verwiesen, das Organ der ACL (*Association for Computational Linguistics*). Es ist online unter <http://www.aclweb.org/anthology-new> verfügbar, zusammen mit elektronischen Versionen von Beitragsbänden zahlreicher CL-Konferenzen.

Die Referenzadresse zur Sprachtechnologie im (deutschsprachigen) Web ist <http://www.lt-world.org>. Hier finden sich Neuigkeiten und nach Sparten geordnete Informationen zur praxisorientierten Sprachverarbeitung.

1.2 Zur Geschichte der Computerlinguistik

Wolfgang Menzel

1.2.1 Die Ursprünge

Die frühen Entwicklungen zur Computertechnologie in den dreißiger und vierziger Jahren des 20. Jahrhunderts waren sehr stark durch die Hinwendung zu numerischen Problemstellungen geprägt. Dieser Umstand spiegelt sich recht deutlich in den ursprünglichen Namensgebungen wider: computational machinery, machine à calculer, ordinateur, ная машина, Elektronenrechner usw. Allerdings wurde auch damals schon das enorme Potential der neuen Technologie für die Behandlung rein symbolischer Verarbeitungsaufgaben erkannt. Ausschlaggebend hierfür war wohl nicht zuletzt der erfolgreiche Einsatz zur Dechiffrierung verschlüsselter Nachrichtentexte, der letztendlich auch die maschinelle Übersetzung der natürlichen Sprache als Spezialfall einer Dekodierungsaufgabe realisierbar erscheinen ließ (Weaver 1949). Zugleich wurden erste Überlegungen zu den prinzipiellen Möglichkeiten der maschinellen Informationsverarbeitung angestellt (Turing 1950). Auch wenn es sich dabei anfangs noch um reine Gedankenexperimente handelte, so bezogen sie sich doch ebenfalls auf ein Szenario, das dem Bereich der maschinellen Sprachverarbeitung zuzuordnen ist, und setzten damit die prinzipielle Realisierbarkeit eines natürlichsprachlichen Dialogs zwischen Mensch und Maschine indirekt schon einmal voraus.

In diesen frühen Überlegungen weisen die sich abzeichnenden Lösungsansätze zur maschinellen Sprachverarbeitung durchaus noch eine gemeinsame Wurzel auf, die stochastische Informationstheorie (Shannon und Weaver 1949). Aus deren Perspektive erscheint ein fremdsprachlicher Text als das Ergebnis der Übertragung einer Nachricht über einen gestörten Kanal. Die Aufgabe etwa der maschinellen Übersetzung besteht dann darin, den ursprünglichen Nachrichtentext unter Verwendung der sprachspezifischen Symbolwahrscheinlichkeiten und der Kanalcharakteristika beim Empfänger zu rekonstruieren.

War zu diesem Zeitpunkt die Einheit des methodischen Inventariums noch weitgehend gewahrt, so konnte man schon bald darauf eine stärkere Aufspaltung in stochastische Verfahren einerseits und symbolische Ansätze andererseits beobachten. Während erstere vor allem im Bereich der Informationswissenschaft, aber auch zur Verifizierung der Autorenschaft eines Textes zum Einsatz kamen, wurden letztere geradezu zum Synonym der späteren Computerlinguistik und dominierten die Entwicklung des Gebiets über einen erstaunlich langen Zeitraum.

Für diese recht einseitige Entwicklung lassen sich sicherlich mehrere Gründe identifizieren. Zum einen war da Chomsky's Diktum (Chomsky 1957), dass prinzipiell kein statistischer Ansatz in der Lage sein kann, den fundamentalen Unterschied zwischen den beiden Sätzen

(1.8) *Colorless green ideas sleep furiously.*

(1.9) *Furiously sleep ideas green colorless.*

zu erfassen, da man mit einiger Sicherheit davon ausgehen darf, dass keiner von beiden jemals in einem englischen Diskurs auftreten würde, und somit einer stochastischen Beobachtung *per se* nicht zugänglich ist. Es hat letztendlich mehr als vier Jahrzehnte intensiver Forschung benötigt, um erkennen zu können, dass diese Annahme grundfalsch war, und dass sich unter Zuhilfenahme versteckter Variablen durchaus stochastische Modelle auf ganz gewöhnlichen englischen Korpusdaten trainieren lassen, die tatsächlich einen Unterschied von mehr als fünf Größenordnungen zwischen den Wahrscheinlichkeiten für diese beiden Sätze vorhersagen (Pereira 2000).

Auf der anderen Seite hatte die einseitige Bevorzugung symbolischer Verfahren aber sicherlich auch ganz praktische Gründe, die vor allem in der mangelnden Leistungsfähigkeit der damals verfügbaren Hardware zu suchen sind. Derartige Beschränkungen bevorzugen in der Tat symbolische Ansätze in ganz entscheidender Weise: So lässt sich etwa die prinzipielle Idee eines symbolischen Verfahrens immer auch anhand eines extrem stark vereinfachten Modells (wenige Regeln, geringer Abdeckungsgrad usw.) demonstrieren, wobei sich die eigentlichen Schwierigkeiten dann natürlich bei der Verallgemeinerung auf größere Sprachausschnitte einstellen. Dagegen muss bei einem vergleichbaren stochastischen Ansatz bereits für das allererste Experiment ein ganz erheblicher Aufwand im Bereich der Datensammlung und der sehr ressourcenintensiven Schätzverfahren (Training) geleistet werden.

1.2.2 Symbolische Sprachverarbeitung

Die frühen Arbeiten zur symbolischen Sprachverarbeitung orientierten sich einerseits sehr stark an den vorhandenen linguistischen Beschreibungsebenen (Morphologie, Syntax, Semantik), zum anderen aber auch an den unmittelbaren Bedürfnissen praktischer Anwendungen, wie Maschinelle Übersetzung und Informationsrecherche. Im Mittelpunkt standen daher Untersuchungen zur lexikalischen Repräsentation und morphosyntaktischen Analyse von Wortformen, sowie zur syntaktischen Struktur von Sätzen.

Auf der Ebene der Morphotaktik lässt sich ein starker Trend hin zu elementaren Techniken aus dem Bereich der Endlichen Automaten bereits seit den frühesten Ansätzen nachweisen. Hinsichtlich der lexikalischen Beschreibungen konzentrierten sich die Bemühungen stark auf die syntaktischen Auswirkungen von Wortbildungs- und Flexionsprozessen, während die semantischen Aspekte lange Zeit eher ausgeklammert wurden. Seit den achtziger Jahren wurden verstärkt Anstrengungen unternommen, die Redundanz im Lexikon zu reduzieren. Einen ersten Schritt hierzu stellte die systematische Nutzung von Transducern zur Modellierung der phonologischen Variation (Koskenniemi 1983) dar. Durch geeignete Vererbungsmechanismen konnte auch auf der Seite der Lexikoninformation eine kompaktere Beschreibung erreicht werden. Um dabei dem Spannungsverhältnis zwischen Regel und Ausnahme angemessen Rechnung zu tragen, kamen dabei zunehmend auch Techniken der nichtmonotonen Vererbung zum Einsatz (Evans und Gazdar 1989).

Wichtigster Motor für die Aktivitäten zur syntaktischen Analyse waren sicherlich die Bedürfnisse der Maschinellen Übersetzung, wo man sich von dem Rückgriff auf syntaktische Repräsentationen einen deutlichen Fortschritt gegenüber den rein wortformbasierten Ansätzen versprach. Zum anderen lag hier ein enger Bezugspunkt mit parallelen Entwicklungen im Bereich der Programmiersprachen vor, wo beim Compilerbau durchaus vergleichbare Techniken zum Einsatz kamen. Dadurch gab es insbesondere in den sechziger und siebziger Jahren eine starke wechselseitige Befruchtung.

Kontrovers wurde vor allem die Frage nach dem jeweils geeigneten Grammatiktyp diskutiert, wobei im wesentlichen Ansätze zur Modellierung der Phrasenstruktur (Chomsky 1957) bzw. der Abhängigkeitsbeziehungen (Tesnière 1959), aber auch Kategorialgrammatiken (Bar-Hillel 1954) verwendet wurden. Besonders einflussreich war hierbei die Schule der Transformationsgrammatik (Chomsky 1957; Chomsky 1965), obwohl diese wegen der zugrundeliegenden generativen Sicht letztendlich keinerlei praktikable Sprachanalysesysteme hervorgebracht hat. Breiten Raum nahmen Untersuchungen zur effizienten Realisierung der syntaktischen Analyse (Parsing) ein. Wichtige Meilensteine stellen der Nachweis eines polynomiauen Algorithmus für beliebige kontextfreie Grammatiken (Earley 1970), sowie die Idee der Wiederverwendung partieller Analyseergebnisse beim Chart-Parsing (Kaplan 1973; Kay 1973) dar.

Waren die frühen Systeme zur Sprachverarbeitung im wesentlichen *ad hoc*-Implementierungen bestimmter algorithmischer Ideen, so ist seit den siebziger Jahren eine zunehmende Tendenz hin zu generischen Formalismen zu verzeichnen, die dank ihres hohen Abstraktionsgrades dann auch für ganz unterschiedliche Verarbeitungsaufgaben eingesetzt werden können. Diese Entwicklung vollzog sich über spezielle Programmiersprachen mit teilweise noch stark prozedural orientierter Semantik (z. B. der durch gezielte Erweiterung aus den Endlichen Automaten entstandene Formalismus der Augmented Transition Networks, ATN; Woods 1970), über stärker deklarativ angelegte Formalismen zur Darstellung linguistischen Wissens (z. B. die Baum- und Graphtransformationssprachen ROBRA; Boitet, Pierre und Quèzel-Ambrunaz (1978) bzw. Systèmes-Q; Colmerauer 1970), bis hin zu den rein deklarativen Formalismen auf der Basis der Unifikation (z. B. die unifikationsbasierten Grammatikformalismen mit kontextfreiem Grundgerüst, wie PATR-II; Shieber 1986). Mit den constraint-basierten Unifikationsformalismen (Shieber 1992) liegt nunmehr auch ein rein deklaratives und dennoch berechnungsuniverselles Modell vor, das einerseits hohen Ansprüchen im Hinblick auf eine prinzipienorientierte und damit erklärendadäquate Modellierung der Grammatik gerecht wird (Chomsky 1981; Pollard und Sag 1994), andererseits aber auch die Brücke zum Paradigma der Logikprogrammierung in der Informatik schlägt.

Generell sind durch die verstärkte Hinwendung zu universell verwendbaren Formalismen auch deren formale Eigenschaften verstärkt ins Blickfeld geraten. Ziel dieser Untersuchungen ist es vor allem, diejenigen Modellklassen zu identifizieren, die es gestatten, eine gegebene Problemstellung mit minimaler Mächtigkeit und größtmöglicher Effizienz zu lösen.

Universell verwendbare Formalismen eröffnen darüber hinaus auch die Möglichkeit zur Realisierung ebenenübergreifender Modelle, die sehr unterschiedliche Aspekte des sprachlichen Wissens integrieren können. Ein Beispiel hierfür ist die Konstruktion einer semantischen Repräsentation auf der Grundlage der Montague-Grammatik (Montague 1974), die dann mit den Mitteln der Unifikation in einem constraint-basierten Formalismus emuliert werden kann (Bouma et al. 1988). Vergleichbare Erweiterungen sind auch zur Einbeziehung satzübergreifender Phänomene auf der Grundlage der Diskursrepräsentationstheorie (DRT; Kamp und Reyle 1993) möglich.

1.2.3 Korpusstatistische Verfahren

Das Wiedererwachen des Interesses an stochastischen Verfahren steht in engem Zusammenhang mit den deutlichen Fortschritten bei der Erkennung gesprochener Sprache seit Anfang der achtziger Jahre. Gerade in diesem Gebiet hat sich gezeigt, dass die automatische Ermittlung von Modellparametern aus einem speziell aufbereiteten Korpus von Sprachdaten (oftmals als Training bezeichnet), einen entscheidenden Schritt zur Lösung des Wissensakquisitionsproblems darstellt. Letztendlich wurde erst durch den konsequenten Einsatz solcher Trainingsverfahren die Erkennung mit großen Wortschätzten und mehreren Sprechern überhaupt ermöglicht (Jelinek 1976).

Für die erfolgreiche Anwendung stochastischer Techniken müssen mehrere, teils widersprüchliche Forderungen erfüllt sein:

- Zum einen muss die Struktur des Modells so gewählt werden, dass die Zahl der zu schätzenden Modellparameter und die verfügbaren Trainingsdaten in einem ausgewogenen Verhältnis stehen.
- Zum anderen sollte das Modell über genügend Freiheitsgrade verfügen, um die Struktur der Daten angemessen widerspiegeln zu können, gleichzeitig aber beschränkt genug sein, um eine Generalisierung über den Trainingsdaten zu erzwingen und ein „Auswendiglernen“ der Einzelbeispiele zu vermeiden.

Ausgangspunkt des Modellentwurfs ist hierbei also nicht ein extern vorgegebener Adäquatheitsanspruch, wie dies für die symbolischen Verfahren charakteristisch ist, sondern vor allem die Frage der wirksamen Trainierbarkeit eines Modells auf einem vorgegebenen Datensatz.

Diese grundlegende Besonderheit teilen die generativ orientierten, stochastischen Verfahren mit anderen Klassen von trainierbaren Modellen, zu denen mit den konnektionistischen Ansätzen, den Support-Vektor-Maschinen, und den Entscheidungsbaum- bzw. Regelinduktionsverfahren auch Systeme zum diskriminativen, sowie zum rein symbolischen Lernen gehören. Wesentliches Charakteristikum ist also nicht so sehr die wahrscheinlichkeitstheoretische Fundierung des Ansatzes, sondern vielmehr die Tatsache, dass in der Trainingsphase die für die jeweilige Aufgabe relevanten statistischen Eigenschaften der Daten zur Modelladaption ausgenutzt werden.

Die wohl erste computerlinguistische Aufgabe, die Ende der achtziger Jahre mit korpusstatistischen Methoden erfolgreich bearbeitet wurde, war die Wortartendisambiguierung (Tagging; DeRose 1988). Angespornt von diesen Anfangserfolgen wurden dann zunehmend anspruchsvollere Zielstellungen verfolgt und Erfahrungen mit komplexeren Modellstrukturen gesammelt. Zu diesen Aufgaben gehören

- die syntaktische Analyse (Parsing) unter Verwendung unterschiedlich stark strukturierter Repräsentationen, z. B. (Briscoe und Waegner 1992),
- die strukturelle syntaktische Disambiguierung, z. B. PP-Attachment (Hindle und Rooth 1993),
- die semantische Lesartendisambiguierung,
- die automatische Ermittlung lexikalischer Information und
- die bilinguale Übersetzung (Brown et al. 1990).

Auch wenn bei den vielfältigen Experimenten zur Entwicklung korpusstatistischer Verfahren oftmals die klassischen Modellvorstellungen der strukturellen Linguistik Pate gestanden haben, so hat sich jedoch bald gezeigt, dass die elementaren Modellstrukturen der traditionellen Ansätze (z. B. kontextfreie Regeln) für eine direkte Übernahme in das neue Paradigma nur bedingt geeignet sind. Dies hat zu einer Reihe von Akzentverschiebungen geführt:

- In vielen Fällen kann eine stochastische bzw. konnektionistische Modellierung besser über die elementaren Operationen des zugrundeliegenden Entscheidungsprozesses (z. B. Transformation von Symbolsequenzen, Paraseraktionen, ...) erfolgen, als auf der Ebene der Modellstrukturen selbst (Magerman 1995, Nivre et al. 2006). Somit rückt die Perspektive der Performance wieder stärker in den Mittelpunkt.
- Das klassische Ideal einer redundanzarmen Beschreibung bringt gleichzeitig eine massive Verletzung der stochastischen Unabhängigkeitsannahme mit sich, so dass sich für eine erfolgreiche Modellierung vielfach sehr komplexe und hochgradig redundante Modellstrukturen besser eignen (Bod 1995).
- Es hat sich herausgestellt, dass sich die verschiedenen Arten von Strukturbeschreibungen unterschiedlich gut mit bestimmten Lernparadigmen (generativ vs. diskriminativ, struktur- vs. operationsbasiert) behandeln lassen. Dies hat u.a. zu einem so völlig unerwarteten Wiedererwachen des Interesses an Dependenzmodellen geführt (McDonald et al. 2005).

Zunehmende Aufmerksamkeit wird nunmehr auch der Frage nach möglichen Synergieeffekten durch die Integration symbolischer, stochastischer und konnektionistischer Verfahren in hybriden Systemlösungen gewidmet. Dies betrifft sowohl die Kopplung von Modellen auf der Basis unterschiedlicher Lernparadigmen (z. B. Nivre und McDonald 2008), als auch die Kombination trainierbarer Verfahren mit klassischen Ansätzen zur manuellen Grammatikentwicklung (z. B.

Foth und Menzel 2006). Eine besondere Herausforderung stellt dabei die optimale Zusammenführung von tiefen und flachen Analyseverfahren dar. Hierdurch kann erreicht werden, dass Verarbeitungskomponenten, die auf den im vorangegangenen Abschnitt behandelten ausdrucksmächtigen Repräsentationsformalismen beruhen, von der Effizienz und breiten sprachlichen Abdeckung flacher Analysetechniken (vgl. Unterkapitel 3.4) profitieren können, auch wenn diese Informationsbeiträge nicht immer sehr zuverlässig sind.

1.2.4 Anwendungen der Computerlinguistik

Obwohl das anwendungsbezogene Problem der Maschinellen Übersetzung bereits am Anfang der Arbeiten zur Computerlinguistik stand, zieht es auch ein halbes Jahrhundert später noch ein unvermindert starkes Forschungsinteresse auf sich, das nur gegen Ende der sechziger Jahre durch die recht pessimistischen Prognosen des ALPAC-Reports (siehe Hutchins 1986) für kurze Zeit abgeschwächt worden war.

Dass trotz einer jahrzehntelangen und intensiven Forschungsarbeit auf diesem Gebiet noch immer wesentliche Fragen der Übersetzungsqualität, sowie der Portierbarkeit auf neue Anwendungsbereiche und Sprachpaare offen sind, zeigt zum einen, dass es sich bei der Maschinellen Übersetzung um ein überaus schwieriges Sprachverarbeitungsproblem handelt. Zum anderen wird aber auch deutlich, dass wir es hier mit einer typischen technologischen Fragestellung zu tun haben, die immer durch einen Kompromiss zwischen Anspruch und Wirklichkeit gekennzeichnet ist, und dass damit so etwas wie eine *endgültige* Lösung des gegebenen Problems auch gar nicht erwartet werden darf. In diesem Sinne steht die Maschinelle Übersetzung gleichberechtigt in einer Reihe mit anderen technologischen Aufgabenbereichen, die sich in einer ganz ähnlichen Situation befinden: Zwar existieren nach nunmehr schon mehreren Jahrhunderten intensiver Entwicklungsarbeiten zahlreiche brauchbare Lösungsansätze für das Problem des Transports von Personen und Gütern, dennoch sind auch hier keinerlei Aussichten auf eine abschließende Behandlung dieser Aufgabenstellung zu erkennen.

Analog hierzu haben seit den achtziger Jahren einige Übersetzungssysteme durchaus auch die Reife zum Einsatz in speziellen Anwendungsszenarien erlangt. Ein Weg hierzu führte über die Beschränkung auf sehr spezielle Textsorten (z. B. Wetterberichte; Thouin 1982). Alternative Ansätze setzen stärker auf eine manuelle Nachbereitung der Übersetzungsresultate. Andere Entwicklungen wiederum zielen vor allem auf eine optimale Unterstützung des Humanübersetzers, dem eine Reihe von Werkzeugen zur Sicherung der terminologischen Konsistenz, zur Wiederverwendung bisheriger Übersetzungsresultate, sowie zur partiellen (Roh-)Übersetzung bei Routineaufgaben an die Hand gegeben werden soll.

Parallel zu den Arbeiten an der Maschinellen Übersetzung ist in den letzten drei Jahrzehnten eine erstaunliche Vielfalt von Anwendungssystemen auf der Grundlage computerlinguistischer Verfahren entwickelt und teilweise auch schon zur Einsatzreife gebracht worden. In vielen Fällen sind diese Arbeiten erst durch die bedeutenden Fortschritte auf anderen Gebieten der Informationstechnologie initiiert bzw. vorangetrieben worden. So wurde die wohl erste erfolgreiche An-

wendung morphologischer Analysetechniken zur automatischen Silbentrennung ganz wesentlich durch den umfassenden Übergang zum Photosatz im Druckereigewerbe Anfang der sechziger Jahre forciert. Erst mit der flächendeckenden Verbreitung der Mikrorechner seit den achtziger Jahren steht diese Technologie als standardmäßiger Bestandteil aller Textverarbeitungssysteme auch einem Massenpublikum zur Verfügung. Vergleichbare Entwicklungen waren auch im Bereich der Hilfsmittel zur Rechtschreibprüfung und -korrektur zu verzeichnen (Peterson 1980).

Recht deutlich lässt sich der Einfluss externer Faktoren auch auf dem Gebiet der Informationssuche nachvollziehen, wo durch die zunehmende Verbreitung des WWW eine deutliche Belebung der diesbezüglichen Forschungsaktivitäten zu verzeichnen ist (Baeza-Yates und Ribeiro-Neto 1999). Durch die explosionsartig anwachsende Menge der digital verfügbaren Information sind in diesem Zusammenhang eine Reihe von Anwendungsszenarien mit zum Teil ganz neuartigen Anforderungen entstanden:

- die Online-Recherche, die sich insbesondere durch extreme Effizienzerwartungen auszeichnet und durch das kontinuierliche Wachstum der online verfügbaren Textinformation mit ständig steigenden Qualitätsanforderungen konfrontiert ist,
- die Informationsfilterung und -klassifikation zur Zuordnung relevanter Dokumente z. B. bei der E-Mail-Sortierung bzw. als Grundlage hochgradig individualisierter Informationsangebote (vgl. das Unterkapitel 5.3),
- die Informationsextraktion zur inhaltlichen Erschließung von Textdokumenten im Hinblick auf stark spezialisierte Informationsbedürfnisse (vgl. ebenfalls das Unterkapitel 5.3) oder aber
- die Beantwortung von beliebigen Fragen aufgrund der in großen Textkorpora enthaltenen Information.

Ein Bereich, der vor allem von der gewaltigen Steigerung der Hardwareleistungsfähigkeit seit Beginn der neunziger Jahre profitiert hat, ist die automatische Spracherkennung, die insbesondere in Form von Diktieranwendungen zunehmende Verbreitung findet. Ein wesentlicher Berührungs punkt mit computerlinguistischen Forschungen ergibt sich hierbei durch die Notwendigkeit, Prädiktionen über Wortformsequenzen (Sprachmodellierung) in die Ermittlung des Erkennungsergebnisses einfließen zu lassen. Benötigt werden hierzu vor allem Verfahren zur leichteren Modelladaption an neue Nutzer und unbekannte Textsorten, sowie Techniken zur besseren Einbeziehung nichtlokaler Abhängigkeiten auf den verschiedenen sprachlichen Ebenen.

Dass sich die fundamentalen Trends der Informationstechnologie durchaus nicht immer förderlich auf die Entwicklung computerlinguistischer Anwendungen auswirken müssen, lässt sich etwa am Beispiel des natürlichsprachlichen Zugriffs zu Datenbanken beobachten, an den Mitte der achtziger Jahre erhebliche kommerzielle Hoffnungen geknüpft waren. Hier wurde die Entwicklung jedoch durch

2010	Dokumentenretrieval für gesprochene Sprache diskriminativ trainierbare Modelle	
		Multimodale Nutzungsschnittstellen
2000	Integration von flacher und tiefer Verarbeitung	
		Fragebeantwortung für offene Textkorpora
		MÜ für gesprochene Sprache
1990	stochastisches Parsing	Informationsextraktion
	stochastisches Tagging	stochastische MÜ, Diktiersysteme
	Constraint-basierte Grammatiken	
	Vererbung im Lexikon	
	Unifikationsgrammatiken, Zweisebenenmorphologie	
1980	Diskursrepräsentationstheorie	
	Semantikkonstruktion	MÜ im Routineeinsatz
	Chart-Parsing	Rechtschreibfehlerkorrektur
1970	ATN-Grammatiken	natürlichsprachliche Datenbankabfrage
		Automatische Silbentrennung
	Morphologische Analyse	
1960	syntaktisches Parsing mit CFG	
		experimentelle MÜ
	Sprachverarbeitung als Zeichenkettenmanipulation	
1950	Erste Gedankenexperimente	

Abbildung 1.1: Zeittafel

das Aufkommen graphischer Nutzerschnittstellen vollständig überholt. Für spezielle, aber typische Anwendungskontexte, wie Fahrplan- und Produktauskünfte, konnte alternativ zur geschriebenen Sprache ein Kommunikationskanal bereitgestellt werden, der eine bequemere und zugleich robustere Mensch-Maschine-Interaktion ermöglicht. Wichtige Aspekte dieser Technologie erfahren allerdings bereits heute eine Neuauflage in Dialogsystemen zur automatischen Telefonauskunft bzw. durch aktuelle Entwicklungsarbeiten zur automatischen Beantwortung von E-Mail im Servicebereich.

2 Formale Grundlagen

Kapitelherausgeber: Ralf Klabunde

Jede computerlinguistische Methode basiert auf speziellen mathematischen und informatik-orientierten Grundlagen. Diese Methoden wiederum finden bei der Entwicklung diverser Werkzeuge und Systeme Anwendung. In diesem Kapitel werden daher die Grundlagen für die im Kapitel 3 vorgestellten computerlinguistischen Methoden eingeführt sowie Grundlagen, die direkt für bestimmte Anwendungen einschlägig sind.

Das Unterkapitel 2.1 stellt die mengentheoretischen und logischen Grundlagen bereit. Während die Mengentheorie für praktisch sämtliche Bereiche der Computerlinguistik unentbehrlich ist, sind die logischen Grundlagen insbesondere für Methoden der Semantik relevant. Die Computersemantik setzt nämlich Konzepte der linguistisch-logischen Semantik in Programme um. In dieser linguistischen Semantikkonzeption werden diverse Logiken herangezogen, um Folgerungsbeziehungen zwischen natürlichsprachlichen Sätzen bzw. Texteinheiten zu erklären. Der Beitrag 3.6 des Methoden-Kapitels stellt die verwendeten semantischen Methoden vor.

Das Unterkapitel 2.2 über formale Sprachen und Automaten führt wichtige Eigenschaften formaler Sprachen ein, um auf dieser Basis Aussagen zur Effizienz der Verarbeitung natürlicher Sprachen zu machen. Zudem finden die vorgestellten Automaten ganz konkrete Anwendung in der Computerphonologie, die im Unterkapitel 3.1 vorgestellt werden, sowie in der Morphologie (Unterkapitel 3.3). Aber auch Anwendungen wie die in Unterkapitel 5.4 vorgestellten Sprachein- und -ausgabesysteme verwenden automatentheoretische Konzepte.

Die Unifikation von Merkmalsstrukturen ist für viele Bereiche der Computerlinguistik die Standardoperation zur Beschreibung sprachlicher Strukturen. Das Unterkapitel 2.3 über Graphentheorie, Merkmalsstrukturen und Unifikation stellt diese wichtige Operation vor, die unter anderem in der Phonologie, der Morphologie und der Syntax (vgl. Beitrag 3.5) eingesetzt wird.

Neben dem symbolischen Verfahren der Unifikation spielen stochastische Verfahren in der Computerlinguistik eine immer größer werdende Rolle. Das Unterkapitel 2.4 stellt diese Verfahren vor, die z. B. in der Syntax Anwendung finden, wenn dort Regeln mit einem Wahrscheinlichkeitswert für ihre Anwendung versehen werden sollen. Aber auch viele Anwendungen wie beispielsweise die in Unterkapitel 5.4 beschriebene Spracherkennung oder das textbasierte Informationsmanagement (siehe Unterkapitel 5.3) verwenden diverse stochastische Verfahren.

Das Unterkapitel 2.5 schließlich stellt texttechnologische Grundlagen vor. Dies sind diejenigen XML-basierten Grundlagen, die bei der computerlinguistischen Verarbeitung relevant werden und insbesondere für die Verwendung des World Wide Webs als Ressource (Unterkapitel 4.7) einschlägig sind.

2.1 Mengenlehre und Logik

Christian Ebert und Cornelia Ebert

Für ein formales Vorgehen im Rahmen der Computerlinguistik sind Grundlagen der Mengenlehre und der Logik unverzichtbar. In der **naiven Mengenlehre**, die wir in diesem Kapitel zunächst betrachten werden, wird unter einer Menge eine beliebige Ansammlung von Objekten verstanden. Diese sehr allgemeine Definition lässt erahnen, dass die entsprechenden mengentheoretischen Konzepte in nahezu allen Bereichen der Computerlinguistik Verwendung finden, sei es beispielsweise bei der Bestimmung von Wahrscheinlichkeiten für Mengen von Ereignissen in der Statistik (Unterkapitel 2.4) oder bei der Definition einer formalen Sprache als Menge von Zeichenketten (Unterkapitel 2.2).

Die **Logik** als Lehre des Schlussfolgerns tritt als Teilgebiet in verschiedenen Disziplinen, wie z. B. der Philosophie und der Informatik, auf. Im Rahmen der Sprachwissenschaft und der Computerlinguistik findet sie hauptsächlich bei der Formalisierung des Bedeutungsbeitrags natürlichsprachlicher Ausdrücke und deren Folgerungspotentials Verwendung, was ausführlich in Unterkapitel 3.6 diskutiert wird. Mit dem Anspruch einer formalen Darstellung befindet man sich dabei auf dem Gebiet der formalen oder **symbolischen Logik**.

Im Rahmen dieses Unterkapitels werden wir logische **Formeln** definieren und daraufhin untersuchen, ob sie bezüglich eines formalen **Modells** eine wahre oder falsche Aussage machen – wir analysieren Formeln als **modelltheoretisch**. Die Frage nach der Bedeutung einer Formel reduziert sich dabei auf die Frage, unter welchen Bedingungen die Formel wahr ist. Man untersucht also die **Wahrheitsbedingungen** der Formel. Wir werden dabei verschiedene Logiksysteme aufeinander aufbauend einführen, sodass wir am Ende ein System zur Hand haben, das mächtig genug ist, um die Berechnung des Bedeutungsbeitrags natürlichsprachlicher Ausdrücke beschreiben zu können.

2.1.1 Mengenlehre

Unter einer **Menge** versteht man eine beliebige Ansammlung von Objekten. Die einfachste Möglichkeit, eine Menge anzugeben, ist, diese Objekte aufzuzählen. Soll z. B. das Symbol F für die Menge der Farben der französischen Flagge – also für blau, weiß und rot – stehen, so schreibt man dafür

$$F = \{ \text{blau}, \text{wei\ss}, \text{rot} \}.$$

Die Menge F wurde somit durch eine Auflistung ihrer **Elemente** innerhalb von geschweiften Klammern angegeben. Dabei ist die Reihenfolge, in der die Ele-

mente angegeben werden, nicht ausschlaggebend. Die Elemente einer Menge sind **ungeordnet**, weshalb man auch $F = \{rot, blau, weiß\}$ hätte schreiben können. Außerdem kommt es bei Mengen nur auf das Vorkommen eines Elements an – mehrfache Vorkommen werden ignoriert. Damit sind z. B. die Mengen $\{a, b, c\}$ und $\{a, a, b, c, c, c\}$ identisch. Möchte man mehrfache Vorkommen unterscheiden, so spricht man von einer **Multimenge**.

Eine Menge lässt sich auch durch Angabe einer charakteristischen Eigenschaft ihrer Elemente beschreiben. Durch

$$F' = \{x \mid x \text{ ist eine Farbe der französischen Flagge}\}$$

wird eine Menge F' beschrieben, die die Farben der französischen Flagge enthält. Obwohl die Mengen F und F' unterschiedlich beschrieben worden sind, enthalten sie dennoch die gleichen Elemente. Man legt fest, dass Mengen gleich sind, wenn sie – unabhängig von der Art der Beschreibung – die gleichen Elemente enthalten, und benutzt als formale Schreibweise dafür das Gleichheitszeichen. Damit gilt also $F = F'$.

Um auszudrücken, dass ein Objekt zu einer Menge gehört, bedient man sich folgender formaler Schreibweise:

Definition 2.1.1

Gehört das Objekt x zur Menge A , so nennt man x ein **Element** der Menge A und schreibt $x \in A$. Kommt x nicht in A vor, so wird dies durch $x \notin A$ ausgedrückt. \square

Durch die Möglichkeit, Mengen durch eine charakteristische Eigenschaft zu beschreiben, ergibt sich scheinbar ein Problem, wie folgende Beschreibung zeigt.

$$L = \{x \mid x \text{ ist eine Primzahl und } x \text{ ist teilbar durch } 6\}$$

Offensichtlich gibt es kein Element, das die charakteristische Eigenschaft erfüllen kann, denn keine Primzahl ist durch sechs teilbar. Damit enthält die Menge L keine Elemente – L ist die **leere Menge**, die mit \emptyset bezeichnet wird. Es gilt also $L = \emptyset$. Allgemein spricht man bei Betrachtung der Anzahl der Elemente einer Menge von der **Kardinalität** der Menge.

Definition 2.1.2

Die Anzahl der Elemente, die eine Menge A enthält, wird als ihre **Kardinalität** bezeichnet und mit $|A|$ bezeichnet. \square

Mengen müssen nicht unbedingt nur endlich viele Elemente enthalten. Die Menge der natürlichen Zahlen $\mathbb{N} = \{1, 2, 3, \dots\}$ enthält z. B. unendlich viele Elemente. In einem solchen Fall ist es natürlich nicht möglich, alle Elemente aufzuzählen, weshalb man sich der abkürzenden Schreibweise mit den drei Punkten bedient und annimmt, dass die zugrunde liegende charakteristische Eigenschaft der Elemente klar ist.

Beispiel 2.1.1

Weitere Beispiele für Mengen sind:

1. $P = \{x \mid x \text{ ist eine Primzahl}\}$ ist die Menge der Primzahlen.
2. $G = \{\text{rot, blau, gelb}\}$ ist die Menge der Grundfarben.
3. $U = \{1, 3, 5, 7, \dots\}$ ist die Menge der ungeraden Zahlen.
4. $K = \{\alpha, \beta, \gamma, \delta, \dots, \omega\}$ ist die Menge der Kleinbuchstaben des griechischen Alphabets.

Es ist z. B. $25 \in U$, aber $25 \notin P$. K und G sind endliche Mengen und es gilt $|K| = 24$ und $|G| = 3$. Die beiden Mengen P und U sind unendlich. Außerdem gilt $|\emptyset| = 0$. \square

Mengen können in verschiedenen Beziehungen zueinander stehen.

Definition 2.1.3

Eine Menge A ist **Teilmenge** von B (geschrieben $A \subseteq B$), wenn jedes Element von A auch in B enthalten ist. Damit ist B eine **Obermenge** von A und man schreibt dafür $B \supseteq A$. Die **Potenzmenge** $\wp(A)$ einer Menge A ist diejenige Menge, die alle Teilmengen von A umfasst, also

$$\wp(A) = \{X \mid X \subseteq A\}$$

\square

Die Definition der Teilmenge schließt nicht aus, dass A und B gleich sind. So ist z. B. nach obigen Definitionen der Farben der französischen Flagge F eine Teilmenge von F' und umgekehrt F' auch eine Teilmenge von F . Möchte man ausdrücken, dass eine Menge A eine Teilmenge von B , aber nicht gleich B ist, so benutzt man den Begriff der **echten Teilmenge** und schreibt dafür $A \subset B$. Entsprechend spricht man auch von einer **echten Obermenge** und schreibt $B \supset A$.

Beispiel 2.1.2

Mit den Bezeichnungen aus Beispiel 2.1.1 gilt $U \subset \mathbb{N}$ und $\{\alpha, \omega\} \subset K$ (und damit $\mathbb{N} \supset U$ und $K \supset \{\alpha, \omega\}$). Außerdem gilt z. B. $F \subseteq F'$, aber nicht $F \subset F'$. Die leere Menge ist Teilmenge jeder Menge. Weiterhin gilt

$$\begin{aligned} \wp(G) &= \{ \emptyset, \{\text{rot}\}, \{\text{blau}\}, \{\text{gelb}\}, \{\text{rot, blau}\}, \{\text{rot, gelb}\}, \\ &\quad \{\text{blau, gelb}\}, \{\text{rot, blau, gelb}\} \}. \end{aligned}$$

Allgemein gilt, dass für jede Menge A die Potenzmenge $\wp(A)$ die leere Menge und die Menge A selbst enthält. Weiterhin ist $\wp(\emptyset) = \{\emptyset\}$. \square

Mittels **Mengenoperationen** lassen sich aus gegebenen Mengen neue Mengen konstruieren.

Definition 2.1.4

Für zwei Mengen A und B sind folgende Mengenoperationen definiert.

1. Die **Vereinigung** $A \cup B$ ist definiert durch

$$A \cup B = \{x \mid x \in A \text{ oder } x \in B\}$$

und enthält also alle Elemente, die in A oder in B vorkommen.

2. Der **Schnitt** $A \cap B$ ist definiert durch

$$A \cap B = \{x \mid x \in A \text{ und } x \in B\}$$

und enthält also alle Elemente, die sowohl in A als auch in B vorkommen.

Gilt $A \cap B = \emptyset$ so haben A und B keine gemeinsamen Elemente und man nennt die beiden Mengen **disjunkt**.

3. Die **Differenz** $A \setminus B$ (A ohne B) ist definiert durch

$$A \setminus B = \{x \mid x \in A \text{ und } x \notin B\}$$

und enthält also alle Elemente, die zwar in A aber nicht in B vorkommen.

Gibt man sich eine Grundmenge X von Objekten vor, so ist das **Komplement** \overline{A} einer Teilmenge $A \subseteq X$ definiert durch

$$\overline{A} = X \setminus A$$

□

Beispiel 2.1.3

Mit den Bezeichnungen aus Beispiel 2.1.1 gilt $P \cap U = P \setminus \{2\}$ (denn alle Primzahlen außer der 2 sind ungerade). Weiterhin gilt z.B. $K \cap G = \emptyset$ (da beide Mengen keine gemeinsamen Elemente haben), $U \cup \mathbb{N} = \mathbb{N}$ (da U Teilmenge von \mathbb{N} ist) und $G \cup F = \{\text{rot, blau, gelb, weiß}\}$. Bezuglich der Grundmenge \mathbb{N} ist $\overline{U} = \{2, 4, 6, \dots\}$, d.h. das Komplement der ungeraden Zahlen (bzgl. den natürlichen Zahlen) sind die geraden Zahlen. □

Im Zusammenhang mit Mengenoperationen gelten einige einfache Gesetze, die in Tabelle 2.1 angegeben sind.

Zu einer Menge lässt sich ihre **charakteristische Funktion** definieren. Sie wird auf ein Objekt der Grundmenge angewandt und liefert den Wert 1, wenn das Objekt ein Element der Menge ist, sonst 0.

Definition 2.1.5

Sei X eine Grundmenge und A eine Teilmenge von X . Die **charakteristische Funktion** $C_A: X \rightarrow \{0, 1\}$ der Menge A ist definiert durch:

$$C_A(x) = \begin{cases} 0 & x \notin A \\ 1 & x \in A \end{cases}$$

1. Kommutativgesetz:

$$\begin{aligned} A \cap B &= B \cap A \\ A \cup B &= B \cup A \end{aligned}$$

2. Assoziativgesetz:

$$\begin{aligned} A \cup (B \cup C) &= (A \cup B) \cup C \\ A \cap (B \cap C) &= (A \cap B) \cap C \end{aligned}$$

3. Distributivgesetz:

$$\begin{aligned} A \cup (B \cap C) &= (A \cup B) \cap (A \cup C) \\ A \cap (B \cup C) &= (A \cap B) \cup (A \cap C) \end{aligned}$$

4. Gesetze von DeMorgan:

$$\begin{aligned} \overline{(A \cup B)} &= (\overline{A} \cap \overline{B}) \\ \overline{(A \cap B)} &= (\overline{A} \cup \overline{B}) \end{aligned}$$

Tabelle 2.1: Einige Gesetze der Mengenlehre

□

Bislang wurden Mengen als beliebige Ansammlung von Objekten aufgefasst. Dieser naive und informelle Zugang führt allerdings zu einem Problem, das von dem britischen Philosophen und Mathematiker Bertrand Russell (1872–1970) entdeckt wurde und das deshalb als **Russells Mengenparadoxon** bekannt ist. Das Problem ist folgendes:

Wie am Beispiel der Potenzmenge zu sehen war, können Mengen selbst wieder Mengen als Elemente enthalten. Damit lässt sich auch folgende Menge definieren:

$$R = \{X \mid X \notin X\}$$

R ist die Menge aller derjenigen Mengen, die sich nicht selbst als Element enthalten (z. B. wären alle Potenzmengen solche und deshalb in R). Da R selbst wieder eine Menge ist, stellt sich die Frage, ob R sich selbst enthält. Zwei Annahmen sind möglich:

1. Nehmen wir an, R enthielte sich selbst, d.h. $R \in R$. Dann müsste R ihre eigene charakteristische Eigenschaft erfüllen, d.h. es müsste $R \notin R$ gelten. Damit dürfte sich R aber eben gerade nicht selbst enthalten, was zu einem Widerspruch zur Annahme führt.
2. Nehmen wir nun also an, R enthielte sich nicht selbst, d.h. $R \notin R$. Dann erfüllt R aber gerade ihre eigene charakteristische Eigenschaft und müsste sich deswegen enthalten, d.h. es müsste $R \in R$ gelten. Dies führt wiederum zu einem Widerspruch.

Dieser widersprüchliche Zustand ist natürlich unbefriedigend. Das Problem wurde von Russell selbst dadurch gelöst, dass Mengen bestimmte Typen zugeordnet wurden. Eine Menge hat in dieser **Typentheorie** immer einen höheren Typ als

ihre Elemente. Damit ist ausgeschlossen, dass eine Menge sich selbst enthalten kann. Die Typentheorie wird im Abschnitt 2.1.4 detailliert behandelt.

Die Konzepte des vorangegangenen Abschnittes werden vor allem im Rahmen der Automatentheorie und Theorie der formalen Sprachen in Unterkapitel 2.2 und im Abschnitt über statistische Grundlagen in Unterkapitel 2.4 Verwendung finden.

2.1.2 Aussagenlogik

In der Aussagenlogik betrachtet man – wie der Name schon sagt – Aussagen und deren Verknüpfung. Natürlichsprachlich betrachtet ist eine Aussage eine Behauptung, die wahr oder falsch sein kann. Beispielsweise ist die Aussage

(2.1) *Der Mond ist aus grünem Käse.*

nach dem heutigen Stand der Wissenschaft eine falsche Behauptung, hingegen würde man

(2.2) *Alle Menschen sind sterblich.*

wohl als wahr ansehen. Aus solchen einfachen atomaren Aussagen lassen sich neue gewinnen, indem man sie verknüpft. Diese Verknüpfung geschieht in der natürlichen Sprache mittels einer Satzkonjunktion wie z. B. *und*. Benutzt man diese Konjunktion mit den beiden obigen Aussagen (2.1) und (2.2), so ergibt sich eine neue Aussage:

(2.3) *Der Mond ist aus grünem Käse und alle Menschen sind sterblich.*

Auch diese Aussage ist offensichtlich falsch, da sie ja die falsche Aussage (2.1) enthält. Hätte man statt der falschen Aussage eine wahre genommen, wäre die zusammengesetzte Aussage wahr; die Aussage wäre auch wahr, hätte man statt der Konjunktion *und* die Konjunktion *oder* benutzt. Die Wahrheit einer zusammengesetzten Aussage hängt also direkt von der Wahrheit ihrer Bestandteile und von den verwendeten Verknüpfungen ab. Die Wahrheit oder Falschheit einer Aussage wird als ihr **Wahrheitswert** bezeichnet. Der Wahrheitswert kann also entweder *wahr* oder *falsch* sein, was noch weiter abgekürzt werden kann, indem man die Zahl 1 für *wahr* und die Zahl 0 für *falsch* verwendet.

Die Aussagenlogik beschäftigt sich nun ausschließlich mit der formalen Beschreibung und Berechnung von Wahrheitswerten, ohne auf andere Zusammenhänge zu achten. So kann z. B. die Aussage

(2.4) *Wenn der Mond aus Käse ist, dann regnet es.*

im Rahmen der Aussagenlogik ohne Probleme interpretiert werden, auch wenn z. B. kein kausaler Zusammenhang zwischen den beiden Teilaussagen erkennbar ist und sie als natürlichsprachliche Äußerung vielleicht wenig Sinn ergibt. Am folgenden Beispiel sieht man auch, dass die Verknüpfungen wirklich nur zur Ermittlung der Wahrheitswerte beitragen. Die *und*-Verknüpfung hat in der natürlichen Sprache beispielsweise auch eine temporale Bedeutung, wie z. B. in

(2.5) *Ich verlasse das Haus und ich steige in mein Auto.*

Eine Umkehrung der Teilaussagen, wie bei

(2.6) *Ich steige in mein Auto und ich verlasse das Haus.*

kann natürlich einen Bedeutungsunterschied ausmachen. Für die Ermittlung des Wahrheitswerts in der Aussagenlogik ist sie allerdings ohne Belang. Um das formale System der Aussagenlogik zu definieren, muss man sich zunächst überlegen,

1. welche Verknüpfungen es geben soll,
2. was als Aussage aufgefasst werden soll und wie sich neue Aussagen aus anderen mittels dieser Verknüpfungen konstruieren lassen und
3. wie die Wahrheitswerte dieser Aussagen zu berechnen sind.

Zunächst wollen wir uns mit den ersten beiden Punkten – der **Syntax der Aussagenlogik** – beschäftigen.

Syntax der Aussagenlogik

Als Grundbausteine der Aussagenlogik dienen **atomare Aussagen** wie z. B. die Aussagen in (2.1) und (2.2), also Aussagen, die nicht zusammengesetzt sind und auf deren Wahrheitswert man sich irgendwie einigen muss. Man beginnt also zunächst mit einer Menge atomarer Aussagen \mathcal{A} , z. B. $\mathcal{A} = \{A_1, A_2, A_3, \dots\}$ oder $\mathcal{A} = \{p, q, r, \dots\}$.

Formal gesehen sind atomare Aussagen also nur noch abstrakte Symbole, was es möglich macht, sich von der natürlichen Sprache zu lösen. Man wird so z. B. nicht in Versuchung kommen wird, diese Symbole weiter zu zerlegen. Aus diesen atomaren Aussagen lassen sich mittels folgender Regeln komplexere Formeln bilden.

Definition 2.1.6

Die Menge der **aussagenlogischen Formeln** über \mathcal{A} ist wie folgt definiert:

1. Jede atomare Aussage in \mathcal{A} ist eine Formel.
2. Ist φ eine Formel, so ist auch $\neg\varphi$ eine Formel.
3. Sind φ und ψ Formeln, so sind auch $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$ und $(\varphi \leftrightarrow \psi)$ Formeln.

□

Der Begriff der Formel umfasst also atomare (Punkt 1.) und zusammengesetzte Aussagen (Punkte 2. und 3.). In dieser formalen Notation sind anstelle der natürlichsprachlichen Satzkonjunktionen *und*, *oder*, *wenn–dann*, *genau dann–wenn* und *nicht* die **Junktoren** \wedge , \vee , \rightarrow , \leftrightarrow bzw. \neg getreten. Man nennt $\neg\varphi$ die

Negation von φ , $(\varphi \wedge \psi)$ die **Konjunktion** von φ und ψ , $(\varphi \vee \psi)$ die **Disjunktion** von φ und ψ , $(\varphi \rightarrow \psi)$ die **Implikation** von φ und ψ und $(\varphi \leftrightarrow \psi)$ die **Äquijunktion** bzw. **Biimplikation** von φ und ψ .

Beispiel 2.1.4

Bei dem folgenden Ausdruck handelt es sich um eine Formel im Sinne der Definition:

$$((\neg A_2 \wedge A_5) \rightarrow (A_1 \vee A_8)) \quad (2.7)$$

Dies wird ersichtlich, wenn man nach und nach alle Teilformeln herleitet. Im folgenden ist diese Herleitung für die Formel (2.7) zusammengefasst, wobei immer eine Rechtfertigung des Schrittes auf der rechten Seite angegeben wurde.

(1)	A_2	ist eine Formel wegen Definition 2.1.6, 1.
(2)	A_5	dto.
(3)	A_1	dto.
(4)	A_8	dto.
(5)	$\neg A_2$	wegen Def. 2.1.6, 2. mit (1)
(6)	$(\neg A_2 \wedge A_5)$	wegen Def. 2.1.6, 3. mit (5) und (2)
(7)	$(A_1 \vee A_8)$	wegen Def. 2.1.6, 3. mit (3) und (4)
(8)	$((\neg A_2 \wedge A_5) \rightarrow (A_1 \vee A_8))$	wegen Def. 2.1.6, 3. mit (6) und (7)

So konnte man also den gewünschten Ausdruck herleiten und damit bestätigen, dass er tatsächlich eine Formel darstellt. \square

Damit ist geklärt, welche Verknüpfungen es geben soll und wie sich komplexe Aussagen mittels dieser Verknüpfungen konstruieren lassen. Die **Semantik der Aussagenlogik** beschäftigt sich nun damit, wie sich deren Wahrheitswert berechnen lässt.

Semantik der Aussagenlogik

Um den Wahrheitswert von Formeln allgemein berechnen zu können, muss man sich zunächst über den Wahrheitswert der atomaren Aussagen klar werden. Im vorigen Abschnitt wurde dargelegt, dass man sich irgendwie darauf einigen muss, ob eine atomare Aussage falsch oder wahr sein soll. Formal geschieht dies mittels einer **Belegungsfunktion**, die jeder atomaren Aussage den Wahrheitswert 1 (wahr) oder 0 (falsch) zuweist.

Definition 2.1.7

Eine **aussagenlogische Belegung** ist eine Funktion $g: \mathcal{A} \rightarrow \{0, 1\}$, die jeder atomaren Aussage den Wert 0 oder 1 zuweist. \square

Von einer aussagenlogischen Belegung ausgehend lässt sich nun der Wahrheitswert einer komplexen Formel berechnen, wenn man festlegt, wie jeder darin vorkommende Junktor die Wahrheitswerte der durch ihn verknüpften Teilformeln miteinander kombiniert. Wie in den vorigen Abschnitten schon angedeutet, soll sich der Junktor \wedge beispielsweise ähnlich der natürlichsprachlichen Konjunktion *und* verhalten. Genauer gesagt soll eine komplexe Formel $(\varphi \wedge \psi)$ genau dann

wahr sein, wenn die beiden Teilformeln φ und ψ wahr sind. Formal fasst man dies durch Definition einer entsprechenden **Interpretationsfunktion** \mathcal{I} , die basierend auf einer Belegungsfunktion jeder komplexen Formel einen Wahrheitswert abhängig vom Wahrheitswert ihrer Teilformeln und dem verknüpfenden Junktor zuweist.

Definition 2.1.8

Die **Interpretation** \mathcal{I} einer Formel der Aussagenlogik bzgl. einer Belegung g ist wie folgt definiert:

1. $\mathcal{I}(A_i) = g(A_i)$ für atomare Aussagen A_i .
2. $\mathcal{I}(\neg\varphi) = 1$, falls $\mathcal{I}(\varphi) = 0$, sonst $\mathcal{I}(\neg\varphi) = 0$.
3. $\mathcal{I}((\varphi \wedge \psi)) = 1$, falls $\mathcal{I}(\varphi) = 1$ und $\mathcal{I}(\psi) = 1$, sonst $\mathcal{I}((\varphi \wedge \psi)) = 0$.
4. $\mathcal{I}((\varphi \vee \psi)) = 1$, falls $\mathcal{I}(\varphi) = 1$ oder $\mathcal{I}(\psi) = 1$, sonst $\mathcal{I}((\varphi \vee \psi)) = 0$.
5. $\mathcal{I}((\varphi \rightarrow \psi)) = 1$, falls $\mathcal{I}(\varphi) = 0$ oder $\mathcal{I}(\psi) = 1$, sonst $\mathcal{I}((\varphi \rightarrow \psi)) = 0$.
6. $\mathcal{I}((\varphi \leftrightarrow \psi)) = 1$, falls $\mathcal{I}(\varphi) = \mathcal{I}(\psi)$, sonst $\mathcal{I}((\varphi \leftrightarrow \psi)) = 0$.

□

Beispiel 2.1.5

Steht A für *Der Mond ist aus grünem Käse* und B für *Alle Menschen sind sterblich*, so wäre nach dem heutigen Stand der Wissenschaft eine aussagenlogische Belegung g mit $g(A) = 0$ und $g(B) = 1$ vernünftig. Nach Punkt 3. der obigen Definition gilt nun beispielsweise $\mathcal{I}((A \wedge B)) = 0$, denn nach Punkt 1. ist $\mathcal{I}(A) = g(A) = 0$ und $\mathcal{I}(B) = g(B) = 1$. Die zusammengesetzte Formel $(A \wedge B)$ steht also im Prinzip für die *und*-verknüpfte natürlichsprachliche Aussage *Der Mond ist aus grünem Käse und alle Menschen sind sterblich* und wird bzgl. g entsprechend als falsch interpretiert. □

Die Semantik der Verknüpfungen lässt sich sehr einfach anhand von **Verknüpfungstafeln** darstellen, die auch **Wahrheitswertetafeln** genannt werden. Tabelle 2.2 stellt die Wahrheitswertetafel der aussagenlogischen Verknüpfungen dar. Die aussagenlogische Belegung ist dabei links von dem Doppelstrich \parallel zu sehen. Jede Zeile in der Tabelle entspricht einer Belegung und den daraus resultierenden Ergebnissen für die Interpretation.

$g(A)$	$g(B)$	$\mathcal{I}(\neg A)$	$\mathcal{I}((A \wedge B))$	$\mathcal{I}((A \vee B))$	$\mathcal{I}((A \rightarrow B))$	$\mathcal{I}((A \leftrightarrow B))$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Tabelle 2.2: Wahrheitswertetafeln der aussagenlogischen Verknüpfungen

Mittels Wahrheitswertetafeln lässt sich der Wahrheitswert von Formeln sehr einfach bestimmen, indem man nach und nach die Wahrheitswerte für alle Teilformeln berechnet.

Beispiel 2.1.6

Die unten stehende Tabelle zeigt eine solche Berechnung anhand von Formel (2.7) aus Beispiel 2.1.4. Abkürzend lassen wir dabei die Anwendung der Interpretationsfunktion \mathcal{I} weg. Allerdings wollen wir nicht Wahrheitswerte für alle möglichen Belegungen (das wären $2^4 = 16$ Stück) berechnen, sondern uns beispielhaft auf drei Belegungen g_1, g_2, g_3 , also drei Tabellenzeilen festlegen:

	A_2	A_5	A_1	A_8	$\neg A_2$	$(\neg A_2 \wedge A_5)$	$(A_1 \vee A_8)$	Formel (2.7)
$g_1 :$	0	0	0	0	1	0	0	1
$g_2 :$	0	1	0	0	1	1	0	0
$g_3 :$	1	1	1	1	0	0	1	1

□

In Beispiel 2.1.6 sieht man also, dass Formel (2.7) je nach Belegung wahr oder falsch sein kann. Bei Belegungen, die die Formel wahr machen, sagt man, dass sie die Formel **erfüllen** und benutzt das Symbol \models um dies zu notieren. Es gilt beispielsweise $g_1 \models ((\neg A_2 \wedge A_5) \rightarrow (A_1 \vee A_8))$. Auch g_3 , nicht jedoch g_2 , erfüllt die Formel.

Es gibt jedoch auch Formeln, die unabhängig von der Belegung immer wahr sind. Beispiele für solche Formeln sind $(A \vee \neg A)$ oder auch $(\neg A \rightarrow (A \rightarrow B))$, was sich leicht mithilfe von Wahrheitswertetafeln wie der obigen nachprüfen lässt: in jeder Zeile – also für jede Belegung – ergibt sich das Resultat 1. Solche Formeln sind **allgemeingültig**, und man nennt sie **Tautologien**.

Auch der umgekehrte Fall kann eintreten. Es gibt Formeln, die von keiner Belegung wahr gemacht werden, in gewissem Sinne also widersprüchlich sind. Solche Formeln sind **unerfüllbar**. Ein einfaches Beispiel ist die Formel $(\neg A \wedge A)$, die besagt, dass neben A auch die Negation von A gelten soll – ein offensichtlicher Widerspruch.

Definition 2.1.9

Für eine Formel φ und eine Belegung g schreibt man $g \models \varphi$, falls $\mathcal{I}(\varphi) = 1$. Man sagt, g **erfüllt** φ . Eine Belegung g erfüllt eine Menge von Formeln Φ , wenn g jede Formel in Φ erfüllt.

Gilt φ unter allen möglichen Belegungen, so nennt man φ **allgemeingültig** oder auch eine **Tautologie** und schreibt $\models \varphi$. Gibt es keine Belegung, die φ wahr macht, so nennt man φ **unerfüllbar**, sonst **erfüllbar**. □

Die oben erwähnte Formel $(A \vee \neg A)$ illustriert das logische Prinzip des **tertium non datur** (dt. *ein Drittes gibt es nicht*) bzw. das Prinzip des **ausgeschlossenen Dritten**. Dies besagt für jede beliebige Aussage, dass entweder die Aussage selbst oder ihr Gegenteil gelten muss – eine dritte Möglichkeit gibt es nicht. In anderen Worten muss bei jeder Disjunktion der Form $(\varphi \vee \neg \varphi)$ entweder die erste

oder die zweite Teilformel wahr und die Disjunktion somit insgesamt allgemeingültig sein.

Bezüglich des Zusammenhangs zwischen Allgemeingültigkeit und Erfüllbarkeit kann man nun folgende Überlegungen anstellen. Nehmen wir an, φ ist eine allgemeingültige Formel. Dann ist per Definition φ unter jeder Belegung wahr, d.h. es gibt keine Belegung, die φ falsch macht. Damit gibt es aber auch keine Belegung, die die Negation $\neg\varphi$ wahr macht: $\neg\varphi$ ist also unerfüllbar. Dieser Schluss funktioniert natürlich entsprechend auch in umgekehrter Richtung. Es gilt somit:

$$\varphi \text{ ist allgemeingültig gdw. } \neg\varphi \text{ ist unerfüllbar} \quad (2.8)$$

Folgerung und Äquivalenz

Einer der wichtigsten Begriffe im Rahmen logischer Betrachtungen ist der der **Folgerung**. Um ihn zu illustrieren, betrachten wir als Beispiel zwei simple Aussagen.

1. *Kuno trinkt Bier.*
2. *Wenn Kuno Bier trinkt, dann freut sich der Wirt.*

Angenommen, beide Aussagen wären wahr – was könnten wir daraus schließen? In diesem Fall ist das recht einfach: die Wahrheit der Aussage *der Wirt freut sich*. Formal gesprochen ist die erste Aussage A atomar, wogegen die zweite eine Implikation ($A \rightarrow B$) darstellt, bei der B für die Aussage *der Wirt freut sich* steht. Die folgende Definition präzisiert den Begriff der Folgerung, sodass sich der intuitiv plausible Schluss von A und ($A \rightarrow B$) zu B formal nachbilden lässt.

Definition 2.1.10

Eine Formel ψ ist eine **Folgerung** einer Menge von Formeln Φ , wenn alle Belegungen, die Φ erfüllen, auch ψ erfüllen. Man schreibt dafür $\Phi \models \psi$ und nennt die Elemente von Φ die **Prämissen** und ψ die **Konklusion**.

Zwei Formeln φ und ψ sind **äquivalent**, wenn sowohl $\varphi \models \psi$ als auch $\psi \models \varphi$ gilt. Die Schreibweise dafür ist $\varphi \equiv \psi$. \square

Beispiel 2.1.7

Wie oben illustriert, gilt $A, (A \rightarrow B) \models B$ (wie hier lässt man die Mengenklammern um die Prämissen der Einfachheit halber oft weg). Für jede Belegung g , die beide Prämissen wahr macht, muss wegen der ersten Prämissen zunächst $g(A) = 1$ gelten. Wenn man dies in Betracht zieht, muss wegen der zweiten Prämissen ($A \rightarrow B$) gelten, dass $g(B) = 1$ – in anderen Worten, dass die Konklusion B wahr ist. Diese Schlussfigur ist unter dem Namen **modus ponens** bekannt. In gewissem Sinne die gegenteilige Schlussfigur ist der **modus tollens** $\neg B, (A \rightarrow B) \models \neg A$. Hier gilt für jede Belegung g , die die Prämissen erfüllt, $g(B) = 0$ und damit $g(A) = 0$, womit also auch die Konklusion $\neg A$ erfüllt ist. Weiterhin nennt man $(\neg B \rightarrow \neg A)$ die **Kontraposition** von $(A \rightarrow B)$ und es gilt $(A \rightarrow B) \equiv (\neg B \rightarrow \neg A)$. \square

In Tabelle 2.3 sind einige Äquivalenzen angegeben, die leicht anhand der Wahrheitswertetafeln in Tabelle 2.2 nachgeprüft werden können. Man sieht in

1.	$\neg\neg\varphi \equiv \varphi$
2.	$(\varphi \wedge \psi) \equiv (\psi \wedge \varphi)$ $(\varphi \vee \psi) \equiv (\psi \vee \varphi)$
3.	$(\varphi \rightarrow \psi) \equiv (\neg\varphi \vee \psi)$ $(\varphi \leftrightarrow \psi) \equiv ((\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi))$
4.	$\neg(\varphi \wedge \psi) \equiv (\neg\varphi \vee \neg\psi)$ $\neg(\varphi \vee \psi) \equiv (\neg\varphi \wedge \neg\psi)$
5.	$(\varphi \wedge (\varphi \vee \psi)) \equiv \varphi$ $(\varphi \vee (\varphi \wedge \psi)) \equiv \varphi$

Tabelle 2.3: Einige aussagenlogische Äquivalenzen

Punkt 3., dass sich Implikation und Äquijunktion auch mittels Disjunktion, Konjunktion und Negation definieren lassen. Die Konjunktion kann ihrerseits mit Punkt 4. (den **De Morganschen Regeln**) und unter Verwendung von Punkt 1. auch auf die Disjunktion zurückgeführt werden.

Beispiel 2.1.8

Folgende Ableitung zeigt, wie sich die Äquijunktion mittels der Äquivalenzen in Tabelle 2.3 auf Negation und Disjunktion zurückführen lässt.

$$\begin{aligned}
 (A \leftrightarrow B) &\equiv ((A \rightarrow B) \wedge (B \rightarrow A)) && \text{mit Punkt 3.} \\
 &\equiv ((\neg A \vee B) \wedge (\neg B \vee A)) && \text{mit Punkt 3.} \\
 &\equiv \neg\neg((\neg A \vee B) \wedge (\neg B \vee A)) && \text{mit Punkt 1.} \\
 &\equiv \neg(\neg(\neg A \vee B) \vee \neg(\neg B \vee A)) && \text{mit Punkt 4.}
 \end{aligned}$$

□

Alle Junktoren lassen sich also mittels \neg und \vee ausdrücken. Die Symbole \wedge , \rightarrow und \leftrightarrow kann man daher einfach als abkürzende Schreibweisen entsprechend der Äquivalenzen in Tabelle 2.3 verstehen.

Es gibt einen wichtigen Zusammenhang zwischen dem semantischen Begriff der Folgerung und der Allgemeingültigkeit. Für eine endliche Menge von Formeln $\{\varphi_1, \dots, \varphi_n\}$ und eine Formel ψ gilt nämlich das **Deduktionstheorem**:

$$\{\varphi_1, \dots, \varphi_n\} \models \psi \text{ gdw. } \models (\varphi_1 \wedge \dots \wedge \varphi_n) \rightarrow \psi \quad (2.9)$$

Damit lässt sich also die Frage nach der Folgerung auf die Frage nach der Allgemeingültigkeit einer Implikationsformel zurückführen.

Beweisverfahren und die Tableaux-Methode

Mit der Methode der Wahrheitswertetafeln hat man ein einfaches systematisches Verfahren, mit dem sich die Erfüllbarkeit (und wegen (2.8) auch Allgemeingültigkeit) von aussagenlogischen Formeln überprüfen lässt, allerdings mit recht großem Aufwand: Enthält die zu prüfende Formel n atomare Aussagen, so gibt es 2^n verschiedene Belegungen und somit ebenso viele Zeilen in der Verknüpfungstafel. Weiterhin ist in mächtigeren Logiken (wie der im nächsten Abschnitt vorgestellten Prädikatenlogik) ein vergleichbares, auf der Semantik basierendes Verfahren nicht anwendbar, da an die Stelle von endlich vielen Belegungen unendlich viele sogenannte Modelle treten, so dass es unmöglich ist, *alle* Modelle zu betrachten.

Es wäre wünschenswert, ein systematisches Verfahren an der Hand zu haben, das diese Probleme umgeht und stattdessen anhand der Syntax – d.h. anhand des Aufbaus der Formeln – möglichst effizient eine Entscheidung über deren Erfüllbarkeit bzw. Allgemeingültigkeit trifft. Solche Verfahren heißen **Beweisverfahren**, und die hier vorgestellte **Tableaux-Methode** ist eine davon.

Wie könnte man beispielsweise beweisen, dass $(A \rightarrow (B \rightarrow A))$ eine Tautologie (d.h. allgemeingültig) ist? Zunächst benutzt man den Zusammenhang zwischen Allgemeingültigkeit und Erfüllbarkeit in (2.8), womit sich die Frage nach der Erfüllbarkeit (genau genommen Unerfüllbarkeit) von

$$\neg(A \rightarrow (B \rightarrow A))$$

stellt. Bei der Tableaux-Methode versucht man zunächst, die Erfüllbarkeit dieser Formel nachzuweisen. Wenn dies misslingt und die Erfüllbarkeit somit widerlegt ist, hat man die Unerfüllbarkeit dieser Formel und damit die Allgemeingültigkeit der ursprünglichen Formel nachgewiesen. Damit gehört die Tableaux-Methode zu den sogenannten **Widerlegungsverfahren**.

Mit dem Wissen über die Semantik der Aussagenlogik stellen wir folgende Überlegungen an: Um die negierte Formel $\neg(A \rightarrow (B \rightarrow A))$ wahr zu machen (d.h. zu erfüllen), muss die Implikation $(A \rightarrow (B \rightarrow A))$ falsch sein. Um dies zu erreichen, muss wiederum A wahr und $(B \rightarrow A)$ falsch (d.h. $\neg(B \rightarrow A)$ wahr) sein. Etwas übersichtlicher schreibt man dafür:

- | | | |
|----|---|--------------|
| 1. | $\neg(A \rightarrow (B \rightarrow A))$ | \checkmark |
| 2. | A | |
| 3. | $\neg(B \rightarrow A)$ | |

In dieser Tabelle – dem **Tableaux** – stehen untereinander die Formeln, die im Verlaufe des Beweises gleichzeitig erfüllt werden müssen. In der ersten Zeile steht die Ausgangsformel, die inzwischen abgearbeitet und deshalb abgehakt wurde. Damit stehen zwei weitere Formeln zur Abarbeitung an. Die atomare Formel A in Zeile 2. kann nicht weiter zerlegt werden – sie stellt nur die Forderung dar, dass A wahr sein muss. Damit bleibt noch $\neg(B \rightarrow A)$ in Zeile 3. zu erfüllen. Dazu muss B wahr und A falsch (d.h. $\neg A$ wahr) sein und das Tableaux kann wie folgt **expandiert** werden:

1.	$\neg(A \rightarrow (B \rightarrow A))$	✓
2.	A	
3.	$\neg(B \rightarrow A)$	✓
4.	B	
5.	$\neg A$	

Damit enthält das Tableaux aber widersprüchliche Information: In Zeile 2. wird gefordert, dass A wahr sein muss, in Zeile 5. hingegen, dass $\neg A$ wahr und damit A falsch sein muss! Ein solches Tableaux, das widersprüchliche Information enthält, nennt man **geschlossen**. Ein geschlossenes Tableaux zeigt, dass wir nicht in der Lage waren, die Ausgangsformel $\neg(A \rightarrow (B \rightarrow A))$ zu erfüllen. Diese Formel ist also unerfüllbar und $(A \rightarrow (B \rightarrow A))$ deshalb eine Tautologie.

Bei näherer Betrachtung des obigen Beweises fällt auf, dass wir sehr schematisch vorgegangen sind: Zeile 1. als auch Zeile 3. sind von der Form $\neg(\varphi \rightarrow \psi)$, und beide Male haben wir das Tableaux durch φ und $\neg\psi$ expandiert. Im Prinzip haben wir also die folgende **Expansionsregel** angewandt:

$$F_{\rightarrow} : \frac{\neg(\varphi \rightarrow \psi)}{\begin{array}{c} \varphi \\ \neg\psi \end{array}}$$

Über dem Strich steht ein Formelschema, auf das die Expansionsregel angewandt werden kann, indem die entsprechenden Formeln unter dem Strich dem Tableaux hinzugefügt werden. Zur Benennung der Regeln benutzen wir F und W mit tiefergestellten Junktoren, um anzudeuten, dass die Falschheit bzw. Wahrheit einer Formel mit dem entsprechenden Junktor bewiesen werden soll.

Hinter der Expansionsregel F_{\rightarrow} steht die semantische Überlegung, dass eine Implikation $(\varphi \rightarrow \psi)$ genau dann falsch ist, wenn φ wahr und ψ falsch ist. Trotzdem ist sie rein syntaktischer Natur: Man könnte ohne weiteres einen Computer programmieren, der diese Regel auf Eingabeformeln anwendet und z. B. obiges Tableaux erzeugt – natürlich ohne sich im Klaren darüber zu sein, was die semantischen Überlegungen dahinter sind.

Mit ähnlichen Überlegungen lassen sich nun Expansionsregeln für die Wahrheit bzw. Falschheit von Formeln mit anderen Junktoren definieren. Eine Konjunktion $(\varphi \wedge \psi)$ ist beispielsweise wahr, wenn φ und ψ wahr sind:

$$W_{\wedge} : \frac{(\varphi \wedge \psi)}{\begin{array}{c} \varphi \\ \psi \end{array}}$$

Allerdings gibt es eine Komplikation: Eine Disjunktion $(\varphi \vee \psi)$ ist beispielsweise wahr, wenn φ oder ψ wahr sind. Diese beiden Formeln untereinander in das Tableaux einzutragen würde nicht das gewünschte Ergebnis liefern, denn das entspräche ja gerade der Konjunktion. Die Lösung besteht darin, die beiden Fälle φ ist wahr und ψ ist wahr getrennt zu betrachten und das Tableaux **verzweigen** zu lassen. Am Beispiel der Formel $((A \vee B) \wedge \neg B) \rightarrow A$, die auf Allgemeingültigkeit hin untersucht werden soll, lässt sich das illustrieren. Das **initiale Tableaux** enthält wieder die Negation der zu überprüfenden Formel:

$$1. \quad \neg(((A \vee B) \wedge \neg B) \rightarrow A)$$

Zunächst kann das Tableaux mittels der Regeln F_{\rightarrow} und W_{\wedge} wie folgt expandiert werden, wobei die Anwendung der Regeln in einer zusätzlichen Spalte mit angegeben ist:

1.	$\neg(((A \vee B) \wedge \neg B) \rightarrow A)$	\checkmark
2.	$((A \vee B) \wedge \neg B)$	1. mit F_{\rightarrow} \checkmark
3.	$\neg A$	1. mit F_{\rightarrow}
4.	$(A \vee B)$	2. mit W_{\wedge}
5.	$\neg B$	2. mit W_{\wedge}

In dieser Situation bleibt als einzige abzuarbeitende Formel die Disjunktion in Zeile 4. übrig. Wie oben erwähnt, muss das Tableaux nun verzweigen, damit man die beiden Möglichkeiten, A oder B zu erfüllen, parallel verfolgen kann:

1.	$\neg(((A \vee B) \wedge \neg B) \rightarrow A)$	\checkmark
2.	$((A \vee B) \wedge \neg B)$	1. mit F_{\rightarrow} \checkmark
3.	$\neg A$	1. mit F_{\rightarrow}
4.	$(A \vee B)$	2. mit W_{\wedge} \checkmark
5.	$\neg B$	2. mit W_{\wedge}
6a.	A	4. mit W_{\vee}
6b.	B	4. mit W_{\vee}

Das Tableaux enthält nun zwei **Zweige**, die disjunktiv zu lesen sind, was die Erfüllbarkeit angeht: Um die Ausgangsformel zu erfüllen, muss man die (noch nicht abgehakten) Formeln in 1.–5. und 6a. *oder* 1.–5. und 6b. erfüllen. Bei genauer Betrachtung sieht man, dass dies in obigem Tableaux nicht möglich ist. Die Forderung in Zeile 6a. nach der Wahrheit von A widerspricht der in Zeile 3. und die Forderung in Zeile 6b. nach Wahrheit von B widerspricht der in Zeile 5. Damit enthalten diese Zweige widersprüchliche Information und man nennt sie **geschlossen**. Man nennt nun ein **Tableaux geschlossen**, wenn alle Zweige des Tableauxs geschlossen sind.

$$\begin{array}{ccccccc}
 N : & \frac{\neg\neg\varphi}{\varphi} & W_{\wedge} : & \frac{(\varphi \wedge \psi)}{\varphi} & F_{\vee} : & \frac{\neg(\varphi \vee \psi)}{\neg\varphi} & F_{\rightarrow} : \frac{\neg(\varphi \rightarrow \psi)}{\varphi} \\
 & & & \psi & & \neg\psi & \neg\psi \\
 & & & & & & \\
 F_{\wedge} : & \frac{\neg(\varphi \wedge \psi)}{\neg\varphi \mid \neg\psi} & W_{\vee} : & \frac{(\varphi \vee \psi)}{\varphi \mid \psi} & W_{\rightarrow} : & \frac{(\varphi \rightarrow \psi)}{\neg\varphi \mid \psi} &
 \end{array}$$

Tabelle 2.4: Expansionsregeln für aussagenlogisches Tableaux

In Tabelle 2.4 sind alle Expansionsregeln der aussagenlogischen Tableaux-Methode angegeben. Wendet man eine der Regeln der unteren Zeile an, so muss das Tableaux erneut verzweigen, was durch den Strich in der Mitte angedeutet

ist. In der folgenden Definition sind die obigen Begriffe nochmals zusammengefasst und der Begriff der **Ableitbarkeit** definiert:

Definition 2.1.11

Ein Zweig eines Tableauxs heißt **geschlossen**, wenn er eine Formel φ und ihre Negation $\neg\varphi$ enthält. Ein Tableaux heißt **geschlossen**, wenn alle seine Zweige geschlossen sind.

Eine Formel φ der Aussagenlogik ist **ableitbar/beweisbar**, wenn das initiale Tableaux, das nur $\neg\varphi$ enthält, nach Anwendung aller möglichen Expansionsregeln in ein geschlossenes Tableaux übergeht. Man schreibt dafür $\vdash \varphi$ und nennt φ ein **Theorem**. \square

Im Folgenden sollen kurz einige wichtige Eigenschaften und Regeln für das Arbeiten mit der Tableaux-Methode angeführt werden.

1. Die Reihenfolge der Expansionen ist unwichtig – verschiedene Reihenfolgen führen zu vielleicht verschiedenen Tableauxs aber zum selben Endergebnis bzgl. Ableitbarkeit. Eine zweimalige Anwendung einer Regel auf dieselbe Formel (d.h. dieselbe Tableauxzeile) hat keine Auswirkungen.
2. Expandiert man das Tableaux wegen einer Formel oberhalb einer oder mehrerer Verzweigungen, so muss jeder Zweig unterhalb dieser Formel entsprechend expandiert werden.
3. Die Tableaux-Methode terminiert immer, d.h. das Verfahren kommt garantiert zum Schluss und hat keine Schleife.

Der letzte Punkt ist insbesondere für die Implementierung des Verfahrens auf einem Computersystem wichtig. Das folgende, etwas komplexere Beispiel, soll obige Punkte nochmals illustrieren.

Beispiel 2.1.9

Durch Anwendung des Deduktionstheorems (2.9) lassen sich auch Folgerungen auf Gültigkeit prüfen. Als Beispiel soll die Folgerung

$$(A \rightarrow C), (A \vee B), \neg B \models (A \wedge C)$$

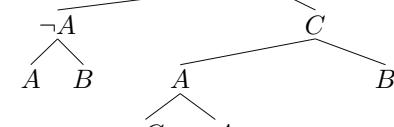
dienen. Durch Anwendung des Deduktionstheorems muss also überprüft werden, ob

$$\models (((A \rightarrow C) \wedge (A \vee B) \wedge \neg B) \rightarrow (A \wedge C))$$

gilt. Wie auch schon oben testet man hierzu die Negation

$$\neg(((A \rightarrow C) \wedge (A \vee B) \wedge \neg B) \rightarrow (A \wedge C))$$

auf Erfüllbarkeit. Das vollständige Tableaux nach Anwendung aller möglichen Expansionsregeln sieht damit wie folgt aus.

1.	$\neg(((A \rightarrow C) \wedge (A \vee B) \wedge \neg B) \rightarrow (A \wedge C))$	✓
2.	$((A \rightarrow C) \wedge (A \vee B) \wedge \neg B)$	1. mit F_{\rightarrow} ✓
3.	$\neg(A \wedge C)$	1. mit F_{\rightarrow} ✓
4.	$(A \rightarrow C)$	2. mit W_{\wedge} ✓
5.	$(A \vee B)$	2. mit W_{\wedge} ✓
6.	$\neg B$	2. mit W_{\wedge}
7.		4. mit W_{\rightarrow}
8.		5. mit W_{\vee}
9.	$\neg C \quad \neg A$	3. mit F_{\wedge}

Zeile 8. ist das Resultat der Anwendung von Regel W_{\vee} auf alle Zweige aus Zeile 7. (siehe Punkt 2. oben). Damit wurden schon drei Zweige geschlossen und müssen nicht mehr weiter betrachtet werden. Eine weitere Expansion mittels F_{\wedge} verzweigt das Tableaux erneut und liefert auch wiederum geschlossene Zweige, womit das ganze Tableaux geschlossen ist und somit die Gültigkeit der Folgerung bewiesen ist. Die Expansionsregeln wurden hier beispielshalber nicht in der Reihenfolge der Tableauxzeilen angewandt (Zeile 3. etwa wurde am Ende expandiert). Eine andere Reihenfolge hätte zwar mit weniger Verzweigungen zu einem geschlossenen Tableaux geführt, aber damit am Ergebnis nichts geändert. \square

Mit der Ableitbarkeit hat man einen syntaktischen Begriff definiert, der dem der Allgemeingültigkeit gegenübersteht. Was nun noch fehlt, ist, die Verbindung zwischen diesen Begriffen herzustellen. Bei der Entwicklung der Tableauxregeln (für beispielsweise F_{\rightarrow}) haben wir das Wissen über die Semantik (in diesem Fall: der Implikation) benutzt, um sicherzustellen, dass die Regeln tatsächlich die semantischen Gegebenheiten widerspiegeln. Deshalb wissen wir, dass es sich bei einer Formel φ tatsächlich dann um eine Tautologie handelt, wenn wir mittels der Tableauxmethode ein geschlossenes Tableaux aus $\neg\varphi$ erhalten. Etwas formaler:

$$\text{Wenn } \vdash \varphi \text{ dann } \models \varphi.$$

Erfüllt ein Beweisverfahren diesen Zusammenhang, so spricht man von der **Korrektheit** des Verfahrens. Sie besagt, dass alle ableitbaren Formeln tatsächlich allgemeingültig sind. Oder in anderen Worten: das Verfahren macht nichts falsch.

Den umgekehrten Zusammenhang bezeichnet man als **Vollständigkeit**:

$$\text{Wenn } \models \varphi \text{ dann } \vdash \varphi.$$

Ist ein Verfahren vollständig, so kann jede Tautologie auch tatsächlich abgeleitet werden. Mit anderen Worten: das Verfahren lässt nichts aus. Die oben angegebene Tableaux-Methode ist ein korrektes und vollständiges Beweisverfahren für die Aussagenlogik. Es gilt also:

$$\text{Allgemeingültigkeit} = \text{Ableitbarkeit} \quad \text{bzw.} \quad \text{Tautologien} = \text{Theoreme}.$$

2.1.3 Prädikatenlogik

Zu Beginn des Abschnitts über Aussagenlogik hatten wir den Satz *Alle Menschen sind sterblich* als atomare Aussage benutzt. In gewisser Hinsicht ist es aber unbefriedigend, einen solchen Satz als nicht weiter zerlegbar zu betrachten. Eigentlich macht der Satz eine Aussage über alle Objekte, die Menschen sind, indem er von diesen Objekten behauptet, dass sie sterblich sind. In der Aussagenlogik hat man keine Möglichkeit, über eine Eigenschaft – ein **Prädikat** – von Objekten zu sprechen oder eine quantitative Aussage zu machen, z. B. eine, die sich auf *alle* Objekte bezieht. Mit dieser Art von Beschreibung beschäftigt sich die **Prädikatenlogik**.

Syntax der Prädikatenlogik

Ein sehr einfaches natürlichsprachliches Beispiel für die Anwendung eines Prädikats stellt der Satz *Peter schläf*t dar, da er dem Objekt Peter die Eigenschaft zu schlafen zuschreibt. Um diesen Satz formal darzustellen, ist es nötig, eine formale Beschreibung für Peter und für die Eigenschaft des Schlafens zu finden. Hierzu dienen **Konstanten(symbole)** und **Prädikats(nsymbole)**. Jedes Prädikat hat eine gewisse **Stelligkeit**, die angibt, wieviele Argumente das Prädikat nehmen kann.

Beispiel 2.1.10

Benutzt man beispielsweise das Konstantensymbol p für das Objekt Peter und das Prädikatssymbol S für die Eigenschaft des Schlafens, so wird obiger Satz mittels $S(p)$ formalisiert. Die Anwendung eines Prädikates (d.h. die Zuschreibung einer Eigenschaft) auf ein Objekt wird also wie die Anwendung einer Funktion auf ihr Argument notiert. S hat damit die Stelligkeit 1, da es genau einem Objekt eine Eigenschaft zuschreibt, nämlich dass das Objekt schläf. Ein Prädikat L , das dem Verb *lieben* entsprechen soll, hätte dementsprechend die Stelligkeit 2, da es zwei Objekten (dem Liebenden und dem Geliebten) gleichzeitig die Eigenschaft zuschreibt. Nehmen wir als weiteres Konstantensymbol j für eine weitere erdachte Person Johanna an, so würde $L(p,j)$ über die beiden Objekte Peter und Johanna aussagen, dass Peter Johanna liebt. \square

Eine Menge von Konstantensymbolen und Prädikatssymbolen mit gegebener Stelligkeit bildet einen **Symbolvorrat**. Es hängt von der konkreten Anwendung ab, welche Symbole man sich vorgibt. Wir werden – wie schon im vorigen Abschnitt zur Aussagenlogik – die Prädikatenlogik zur Formalisierung des Bedeutungsbeitrags sprachlicher Ausdrücke verwenden und entsprechend Symbole benutzen, die mit bestimmten sprachlichen Einheiten korrespondieren. In der Linguistik hat es sich eingebürgert, dementsprechend aussagekräftige Symbole zu wählen, also etwa ein Prädikatssymbol *schlafen'* und ein Konstantensymbol *peter'*, um somit den Satz *Peter schläf*t als *schlafen'(peter')* zu formalisieren. Die unterschiedliche Schriftart und das Hochkomma dienen hierbei dazu, das Symbol, das Teil der Sprache der Prädikatenlogik ist, von einem entsprechenden natürlichsprachlichen Ausdruck (z. B. der Infinitivform *schlafen*) zu unterscheiden.

den. Wichtig ist außerdem, sich der Willkürlichkeit dieser Symbole klar zu werden – man könnte die Prädikation genauso als **apfel(birne)** ausdrücken. Welche genaue Bedeutung den Symbolen zukommt, ist nämlich nicht Teil der Definition des Symbolvorrats, sondern der Interpretation als Teil der Semantik, die im nächsten Abschnitt betrachtet wird.

Zunächst befassen wir uns aber wieder mit der Syntax, also der Definition wohlgeformter Ausdrücke der Prädikatenlogik. Neben dem Symbolvorrat wird noch eine unendliche Menge von **Variablen** \mathcal{V} , z. B. $\mathcal{V} = \{v_0, v_1, v_2, \dots\}$ oder $\mathcal{V} = \{x, y, \dots\}$ benötigt. Diese Variablen dienen wie auch Konstanten dazu, Objekte zu bezeichnen. Beide werden zu **Termen** zusammengefasst.

Definition 2.1.12

Die Menge der prädikatenlogischen **Terme** bei gegebenem Symbolvorrat ist wie folgt definiert:

1. Jede Variable ist ein Term.
2. Jede Konstante ist ein Term.

□

Wie auch in der Aussagenlogik werden nun wieder syntaktische Ausdrücke namens **Formeln** dazu dienen, Aussagen zu machen bzw. einen Wahrheitswert auszudrücken. Anders als in der Aussagenlogik sind diese nun allerdings Prädikationen wie oben illustriert oder Identitätsausdrücke der Form $x = \text{peter}'$. Die Junktoren, mit denen sich aus einfachen Formeln komplexere formen lassen, sind aber wiederum dieselben.

Definition 2.1.13

Die Menge der prädikatenlogischen **Formeln** ist wie folgt definiert:

1. Sind t und s Terme, so ist $t = s$ eine Formel.
2. Sind t_1, \dots, t_n Terme und ist P ein n -stelliges Prädikatssymbol, so ist $P(t_1, \dots, t_n)$ eine Formel.
3. Ist φ eine Formel, so ist auch $\neg\varphi$ eine Formel.
4. Sind φ und ψ Formeln, so sind auch $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$ und $(\varphi \leftrightarrow \psi)$ Formeln.
5. Ist φ eine Formel und x eine Variable, so sind auch $\forall x\varphi$ und $\exists x\varphi$ Formeln.

□

Die nach den Punkten 1. und 2. aufgebauten Formeln heißen **atomare Formeln** und finden ihre Entsprechung in den atomaren Aussagen in Punkt 1. der Definition 2.1.6 der Syntax der Aussagenlogik. Die **komplexen Formeln**, die durch die Punkte 3. und 4. definiert werden, entsprechen den komplexen Aussagen in den Punkten 2. und 3. der Definition 2.1.6.

Durch Punkt 5. der Definition werden nun die **Quantoren** – nämlich der **Allquantor** \forall und der **Existenzquantor** \exists – in die Logik eingeführt. In einer quantifizierten Formel wie $\forall x\varphi$ bezeichnet man φ als den **Skopus des Quantors**. Man sagt, dass die Variable x (durch den Quantor) **gebunden** wird, oder genauer, falls sie im Skopus des Quantors erscheint, dass sie innerhalb des Skopus **gebunden vorkommt**. Entsprechend kann eine Variable auch nicht gebunden vorkommen und heißt dann **frei**. Mit dem Begriff des freien Vorkommens von Variablen kann man nun **Sätze** wie folgt definieren:

Definition 2.1.14

Eine Formel φ ist ein **Satz**, wenn in φ keine Variable frei vorkommt. \square

Beispiel 2.1.11

Benutzen wir $peter'$ und $johanna'$ als Konstantensymbole und $schlafen'$ als einstelliges und $lieben'$ als zweistelliges Prädikatssymbol, dann sind folgende Ausdrücke Beispiele für Formeln der Prädikatenlogik.

1. $(schlafen'(peter') \vee schlafen'(johanna'))$
2. $\forall x lieben'(x, johanna')$
3. $(\forall x \exists y lieben'(z, y) \wedge \neg(y = peter'))$

In der Formel in Punkt 2. ist der Skopus des Allquantors $lieben'(x, johanna')$. Die Variable x ist somit gebunden und die Formel deshalb ein Satz. In der Formel in Punkt 3. ist der Skopus des Existenzquantors $lieben'(z, y)$ und der des Allquantors $\exists y lieben'(z, y)$. Es fällt zunächst auf, dass der Skopus des Allquantors kein Vorkommen der Variable x enthält, die er zu binden versucht. Damit hat der Quantor keinen semantischen Effekt, wie wir im nächsten Unterkapitel zur Semantik sehen werden, und ist im Prinzip überflüssig. Weiterhin kommt die Variable y innerhalb des Skopus des Existenzquantors gebunden vor, aber nicht im zweiten Konjunkt $\neg(y = peter')$, wo sie frei ist. Die Variable z schließlich kommt nur frei vor. \square

Im folgenden Abschnitt werden wir sehen, wie Terme und Formeln interpretiert werden. Dabei werden Terme durch Objekte und Prädikate durch Eigenschaften von bzw. Relationen zwischen Objekten interpretiert. Formeln werden wiederum Wahrheitswerte zugewiesen, wobei quantifizierte Formeln so interpretiert werden, dass sie All- bzw. Existenzaussagen über ihren Skopus machen.

Semantik der Prädikatenlogik

Da in der Prädikatenlogik Aussagen über Eigenschaften von Objekten gemacht werden, muss man sich zunächst auf eine Menge von Objekten einigen und angeben, wie die entsprechenden syntaktischen Einheiten darauf abgebildet werden. Diese Menge von Objekten samt der Abbildung nennt man ein **Modell**.

Definition 2.1.15

Ein **Modell der Prädikatenlogik** ist ein Paar $\mathcal{M} = (D, F)$, wobei

- D eine nicht-leere Menge, die so genannte **Domäne**, ist,
- F eine Abbildung ist, die
 1. jedem Konstantensymbol ein Element aus D und
 2. jedem n -stelligen Prädikatssymbol eine n -stellige Relation über D zuordnet.

□

D nennt man auch **Individuenbereich** (und die Elemente von D entsprechend **Individuen**) oder **Universum**. Man könnte sich beispielsweise als Domäne die Menge der natürlichen Zahlen, die Menge der Knoten in einem Graphen oder auch die Menge der Dinge in einer bestimmten Situation (Personen, Sachen, Abstrakta) vorstellen. Im Folgenden wollen wir als Beispiel für eine Domäne D eine fiktive Lerngruppe, bestehend aus Peter, Hans und Johanna samt zugehöriger Bücher, nämlich dem *Duden* und *Algebra II* annehmen, also

$$D = \{ \text{Peter, Hans, Johanna, Duden, Algebra II} \}. \quad (2.10)$$

Die zweite Komponente eines Modells ist eine Funktion F , die die Elemente aus dem Symbolvorrat – also die Konstanten- und Prädikatssymbole – auf Individuen bzw. Relationen über der Domäne D abbildet.

Wir erweitern den schon oben benutzten Symbolvorrat etwas, sodass wir als Konstantensymbole nun

peter, johanna, duden, algebra

zur Verfügung haben (wobei wir ab jetzt der Einfachheit halber keine Hochkommas mehr benutzen). Als Prädikatssymbole benutzen wir

schlafen, buch, lesen

Wie schon oben erwähnt, sind diese Symbole im Prinzip willkürlich gewählt und müssen in keiner Weise mit der Bezeichnung der Objekte der Domäne korrespondieren, wobei wir aber wieder entsprechend aussagekräftige Symbole gewählt haben. Die eigentliche Verbindung zwischen diesen Symbolen und den Objekten der Domäne kommt über die Interpretationsfunktion F zustande. Für die Konstantensymbole, die mittels F auf Objekte abgebildet werden, soll folgende Zuweisung gelten:

$$F(\text{peter}) = \text{Peter} \quad F(\text{johanna}) = \text{Johanna}$$

$$F(\text{duden}) = \text{Duden} \quad F(\text{algebra}) = \text{Algebra II}$$

Es mag an dieser Stelle auffallen, dass die Domäne D das Objekt Hans enthält, obwohl es nicht als Interpretation irgendeines Symbols dient. Das Objekt Hans hat in gewisser Weise also keinen Namen, den wir benutzen könnten, um innerhalb der Logik direkt darüber zu reden. Trotzdem kann man solche unbenannten

Objekte nicht einfach aus der Domäne entfernen, da sie z. B. für die Interpretation quantifizierender Formeln wichtig sind, die Aussagen über *alle* Objekte machen.

Die Interpretationsfunktion F soll nun noch die Prädikatssymbole wie folgt interpretieren:

$$\begin{aligned} F(\text{schlafen}) &= \{\text{Peter}\} \\ F(\text{buch}) &= \{\text{Duden, Algebra II}\} \\ F(\text{lesen}) &= \{\langle \text{Peter}, \text{Duden} \rangle, \langle \text{Johanna}, \text{Duden} \rangle, \langle \text{Hans}, \text{Algebra II} \rangle\} \end{aligned}$$

Wir haben F so definiert, dass **buch** als die Menge aller Bücher und **lesen** als eine Relation, also als eine Menge von Paaren $\langle x, y \rangle$, wobei x y liest, interpretiert wird. Das Modell $\mathcal{M} = (D, F)$ ist damit also vollständig festgelegt. Bevor allerdings die Interpretation von Formeln definiert werden kann, ist noch festzulegen, wie Variablen interpretiert werden sollen, die ja auch für Objekte der Domäne stehen sollen. Eine **Variablenbelegungsfunktion** wird dies übernehmen.

Definition 2.1.16

Eine **Variablenbelegung der Prädikatenlogik** ist eine Funktion $g: \mathcal{V} \rightarrow D$, die jeder Variablen einen Wert aus der Domäne zuweist. Man schreibt $h[x]g$ für zwei Belegungen, wenn h und g auf allen Variablen übereinstimmen, mit Ausnahme der Variablen x , der auch unterschiedliche Werte zugewiesen werden können. \square

Nun haben wir alles beisammen um die Interpretation von Formeln und Termen formal definieren zu können. Analog zum Fall der Aussagenlogik (Definition 2.1.8) wird hierzu eine **Interpretationsfunktion** $\llbracket \cdot \rrbracket^{\mathcal{M}, g}$ definiert (basierend auf einem Modell \mathcal{M} und einer Variablenbelegung g), die jedem Term und jeder Formel ein Objekt bzw. einen Wahrheitswert zuordnet. Für die Interpretation eines Ausdrucks φ schreibt man $\llbracket \varphi \rrbracket^{\mathcal{M}, g}$.

Definition 2.1.17

Die **Interpretation eines Terms der Prädikatenlogik** bezüglich eines Modells $\mathcal{M} = (D, F)$ und einer Variablenbelegung g ist wie folgt definiert:

1. $\llbracket x \rrbracket^{\mathcal{M}, g} = g(x)$ für alle Variablen x .
2. $\llbracket c \rrbracket^{\mathcal{M}, g} = F(c)$ für alle Konstanten c .

Die **Interpretation einer Formel der Prädikatenlogik** bezüglich eines Modells $\mathcal{M} = (D, F)$ und einer Variablenbelegung g ist wie folgt definiert:

3. $\llbracket t = s \rrbracket^{\mathcal{M}, g} = 1$ gdw. $\llbracket t \rrbracket^{\mathcal{M}, g} = \llbracket s \rrbracket^{\mathcal{M}, g}$
4. $\llbracket P(t_1, \dots, t_n) \rrbracket^{\mathcal{M}, g} = 1$ gdw. $\langle \llbracket t_1 \rrbracket^{\mathcal{M}, g}, \dots, \llbracket t_n \rrbracket^{\mathcal{M}, g} \rangle \in F(P)$
5. $\llbracket \neg \varphi \rrbracket^{\mathcal{M}, g} = 1$ gdw. $\llbracket \varphi \rrbracket^{\mathcal{M}, g} = 0$
6. $\llbracket (\varphi \wedge \psi) \rrbracket^{\mathcal{M}, g} = 1$ gdw. $\llbracket \varphi \rrbracket^{\mathcal{M}, g} = 1$ und $\llbracket \psi \rrbracket^{\mathcal{M}, g} = 1$

7. $\llbracket(\varphi \vee \psi)\rrbracket^{\mathcal{M},g} = 1$ gdw. $\llbracket\varphi\rrbracket^{\mathcal{M},g} = 1$ oder $\llbracket\psi\rrbracket^{\mathcal{M},g} = 1$
8. $\llbracket(\varphi \rightarrow \psi)\rrbracket^{\mathcal{M},g} = 1$ gdw. $\llbracket\varphi\rrbracket^{\mathcal{M},g} = 0$ oder $\llbracket\psi\rrbracket^{\mathcal{M},g} = 1$
9. $\llbracket(\varphi \leftrightarrow \psi)\rrbracket^{\mathcal{M},g} = 1$ gdw. $\llbracket\varphi\rrbracket^{\mathcal{M},g} = \llbracket\psi\rrbracket^{\mathcal{M},g}$
10. $\llbracket\forall x\varphi\rrbracket^{\mathcal{M},g} = 1$ gdw. für alle Belegungen $h[x]g$ gilt: $\llbracket\varphi\rrbracket^{\mathcal{M},h} = 1$
11. $\llbracket\exists x\varphi\rrbracket^{\mathcal{M},g} = 1$ gdw. für mindestens eine Belegung $h[x]g$ gilt: $\llbracket\varphi\rrbracket^{\mathcal{M},h} = 1$

□

Auch bei der Semantik finden sich Parallelen zu der Aussagenlogik, denn die Punkte 5.–9. entsprechen den Punkten 2.–6. der Semantik der Aussagenlogik (Definition 2.1.8). Somit kann man zur Berechnung des Wahrheitswertes komplexer Formeln wieder die Wahrheitswertetafeln in Tabelle 2.2 benutzen. Folgende Definition ist nur eine leichte Anpassung von Definition 2.1.9 der Aussagenlogik.

Definition 2.1.18

Für eine Formel φ , ein Modell \mathcal{M} und eine Variablenbelegung g schreibt man $\mathcal{M}, g \models \varphi$, falls $\llbracket\varphi\rrbracket^{\mathcal{M},g} = 1$. Man sagt, **die Formel φ ist erfüllt** (bezüglich \mathcal{M} und g). Eine Menge von Formeln Φ ist erfüllt bzgl. \mathcal{M} und g , wenn jede Formel aus Φ bzgl. \mathcal{M} und g erfüllt ist. Weiterhin schreibt man $\mathcal{M} \models \varphi$, falls für alle Variablenbelegungen g gilt, dass $\mathcal{M}, g \models \varphi$ (also falls die Interpretation unabhängig von der Variablenbelegung ist).

Gilt $\mathcal{M} \models \varphi$ für alle Modelle \mathcal{M} , so nennt man φ **allgemeingültig** bzw. eine **Tautologie** und schreibt $\models \varphi$. Gibt es kein Modell, das φ erfüllt, so nennt man φ **unerfüllbar**, sonst **erfüllbar**. □

Beispiel 2.1.12

Im Folgenden sollen die Interpretationen bzgl. des oben festgelegten Lerngruppenmodells \mathcal{M} und einer Variablenbelegung g mit $g(x) = \text{Hans}$ und $g(y) = \text{Duden}$, vorgenommen werden. Die Terme **johanna** und **y** erhalten damit beispielsweise folgende Interpretation:

$$\begin{aligned}\llbracket \text{johanna} \rrbracket^{\mathcal{M},g} &= F(\text{johanna}) = \text{Johanna} \\ \llbracket y \rrbracket^{\mathcal{M},g} &= g(y) = \text{Duden}\end{aligned}$$

Die Formel $\text{lesen}(x, y)$ ist nicht erfüllt bzgl. \mathcal{M} und g , denn gemäß Definition ist

$$\llbracket \text{lesen}(x, y) \rrbracket^{\mathcal{M},g} = 1 \quad \text{gdw.} \quad \langle g(x), g(y) \rangle \in F(\text{lesen})$$

Da $g(x) = \text{Hans}$ und $g(y) = \text{Duden}$ und das Paar $\langle \text{Hans}, \text{Duden} \rangle$ nicht in $F(\text{lesen})$ enthalten ist, ist die Formel bzgl. dieser Variablenbelegung g nicht erfüllt. Bezüglich einer anderen Belegung, die sich in x von g unterscheidet (für die also $h[x]g$ gilt), sodass $h(x) = \text{Peter}$ wäre sie erfüllt: es würde $\mathcal{M}, h \models \text{lesen}(x, y)$ gelten.

Folgendes Beispiel illustriert die Interpretation einer quantifizierten Formel.

$$\llbracket \exists x \text{lesen}(x, \text{duden}) \rrbracket^{\mathcal{M}, g} = 1$$

gdw. für mind. eine Belegung $h[x]g$ gilt: $\llbracket \text{lesen}(x, \text{duden}) \rrbracket^{\mathcal{M}, h} = 1$

gdw. für mind. eine Belegung $h[x]g$ gilt: $\langle \llbracket x \rrbracket^{\mathcal{M}, h}, \llbracket \text{duden} \rrbracket^{\mathcal{M}, h} \rangle \in F(\text{lesen})$

gdw. für mind. eine Belegung $h[x]g$ gilt: $\langle h(x), \text{Duden} \rangle \in F(\text{lesen})$

Diese Formel ist also genau dann wahr, wenn man (mindestens) eine Belegung h finden kann, die allen Variablen genau die gleichen Werte wie g zuweist – außer der Variable x – sodass $\langle h(x), \text{Duden} \rangle$ in $F(\text{lesen})$ enthalten ist. So eine Belegung gibt es tatsächlich, wie wir gerade zuvor festgestellt haben: das zuvor verwendete h unterscheidet sich von g nur in x , und es gilt $\mathcal{M}, h \models \text{lesen}(x, \text{duden})$. Somit hat man also eine entsprechende Belegung gefunden und die ursprüngliche Formel ist damit wahr. Wie man sieht, kommt es aber tatsächlich nicht auf die Existenz einer Belegung, sondern vielmehr eines entsprechenden Wertes (hier: Peter) an, auf den man die quantifizierte Variable (in diesem Falle: x) abbilden kann. Die Wirkung des Quantors \exists ist also existentiell: eine derart quantifizierte Formel ist genau dann wahr, wenn im Modell ein Objekt existiert, mithilfe dessen sich der Skopus erfüllen lässt. Die Formel macht also die Aussage: es gibt (mindestens) ein Individuum, das den Duden liest.

Dieses Beispiel illustriert auch einen anderen wichtigen Punkt: die Interpretation von Sätzen (also von Formeln ohne freie Variablen) ist unabhängig von der gegebenen Variablenbelegung. Es spielt keine Rolle, welche Werte g den Variablen zuweist, denn die einzige vorkommende Variable x ist gebunden. Somit darf man g an dieser Stelle sowieso abändern und die Formel bzgl. einer anderen Belegung $h[x]g$ betrachten. Der ursprüngliche Wert $g(x)$ ist irrelevant, genauso wie die Werte, die g anderen Variablen zuweist.

Betrachten wir eine noch etwas komplexere Formel mit zwei verschachtelten Quantifikationen:

$$\llbracket \forall y (\text{buch}(y) \rightarrow \exists x \text{lesen}(x, y)) \rrbracket^{\mathcal{M}, g} = 1$$

gdw. für alle Belegungen $h[y]g$ gilt: $\llbracket (\text{buch}(y) \rightarrow \exists x \text{lesen}(x, y)) \rrbracket^{\mathcal{M}, h} = 1$

gdw. für alle Belegungen $h[y]g$ gilt: $\llbracket \text{buch}(y) \rrbracket^{\mathcal{M}, h} = 0$

oder $\llbracket \exists x \text{lesen}(x, y) \rrbracket^{\mathcal{M}, h} = 1$

gdw. für alle Belegungen $h[y]g$ gilt: $h(y) \notin F(\text{buch})$

oder für mind. eine Belegung $j[x]h$ gilt: $\llbracket \text{lesen}(x, y) \rrbracket^{\mathcal{M}, j} = 1$

Hier müssen wir für alle Belegungen $h[y]g$ (und damit alle Objekte als mögliche Ergebnisse von $h(y)$) prüfen, ob sie (1) die Eigenschaft Buch zu sein nicht haben, oder ob wir (2) eine Belegung $j[x]h$ finden, sodass $\llbracket \text{lesen}(x, y) \rrbracket^{\mathcal{M}, j} = 1$. Für die Objekte Peter, Hans und Johanna ist (1) erfüllt, d.h. sie haben nicht die Eigenschaft zu $F(\text{buch})$ zu gehören. Für die Objekte Duden und Algebra II ist (1) nicht erfüllt und wir müssen (2) überprüfen. Für das Objekt Duden als Wert von y haben wir oben schon gesehen, dass $\exists x \text{lesen}(x, y)$ gilt. Und auch für das

Objekt *Algebra II* als Wert von y gilt $\exists x \text{lesen}(x, y)$, denn wir finden ein x , nämlich Hans, sodass das Paar der Interpretationen von x und y in $F(\text{lesen})$ enthalten ist. Insgesamt ist die Formel damit wahr. Wie man sieht ist die Wirkung von \forall universell: eine derart quantifizierte Formel ist genau dann wahr, wenn im Modell alle Objekte so sind, dass sich mithilfe ihrer der Skopus erfüllen lässt. Die Formel macht also die Aussage: für jedes Buch gilt: es gibt (mind.) ein Individuum, das es liest. \square

Folgerung, Äquivalenz und die Tableaux-Methode

Auch die Definitionen für Folgerung und semantische Äquivalenz stimmen nahezu mit denen der Aussagenlogik überein (siehe Definition 2.1.10). Und auch die aussagenlogischen Äquivalenzen aus Tabelle 2.3 gelten für prädikatenlogische Formeln. Es gibt jedoch auch eine Reihe weiterer Äquivalenzen, die auf die Besonderheiten der Prädikatenlogik eingehen, insbesondere auf die Behandlung von Quantoren. Diese sind in Tabelle 2.5 dargestellt.

1. $\begin{array}{rcl} \forall x\varphi & \equiv & \neg\exists x\neg\varphi \\ \exists x\varphi & \equiv & \neg\forall x\neg\varphi \end{array}$
2. $\begin{array}{rcl} (\forall x\varphi \wedge \forall x\psi) & \equiv & \forall x(\varphi \wedge \psi) \\ (\exists x\varphi \vee \exists x\psi) & \equiv & \exists x(\varphi \vee \psi) \end{array}$
3. $\begin{array}{rcl} \forall x\forall y\varphi & \equiv & \forall y\forall x\varphi \\ \exists x\exists y\varphi & \equiv & \exists y\exists x\varphi \end{array}$
4. Falls x in ψ nicht frei vorkommt:

$$\begin{array}{rcl} (\forall x\varphi \wedge \psi) & \equiv & \forall x(\varphi \wedge \psi) \\ (\forall x\varphi \vee \psi) & \equiv & \forall x(\varphi \vee \psi) \\ (\exists x\varphi \wedge \psi) & \equiv & \exists x(\varphi \wedge \psi) \\ (\exists x\varphi \vee \psi) & \equiv & \exists x(\varphi \vee \psi) \end{array}$$

Tabelle 2.5: Einige prädikatenlogische Äquivalenzen

Anhand der Äquivalenzen in Punkt 1. sieht man, dass ein Quantor auf den anderen zurückgeführt werden kann – man hätte sich also auf eine der beiden Quantorendefinitionen beschränken können. Weiterhin gelten auch der Zusammenhang zwischen Allgemeingültigkeit und Erfüllbarkeit (2.8) und das Deduktionstheorem (2.9).

Das Tableaux-Verfahren der Aussagenlogik kann mittels zusätzlicher Expansionsregeln zur Behandlung der Quantoren, auf die wir hier nicht eingehen können, auch für die Prädikatenlogik verwendet werden, wo es weiterhin korrekt und vollständig bleibt. Allerdings ist nicht garantiert, dass das Verfahren immer terminiert. Obwohl das Verfahren also vollständig ist und es damit für jede

Tautologie eine Ableitung gibt, existiert kein Algorithmus, der allgemein angibt, wie diese Ableitungen vonstatten zu gehen hätten. Dies ist kein Mangel des Tableaux-Verfahrens, sondern eine Eigenschaft der Prädikatenlogik selbst – die Prädikatenlogik ist **unentscheidbar**.

2.1.4 Typenlogik

Wir betrachten noch einmal das Beispiel der Lerngruppe aus Abschnitt 2.1.3. Wir wissen, dass beispielsweise ein Verb wie *lesen* zwei Argumente verlangt, um einen vollständigen Satz zu ergeben. Es fragt zum einen nach dem Subjekt (also demjenigen, der der Tätigkeit des Lesens nachgeht) und zum anderen nach dem Objekt (also dem Gegenstand, der gelesen wird). In der Formalisierung der Prädikatenlogik hat damit das Prädikat *lesen* die Stelligkeit zwei. Es verlangt zwei Argumente – nämlich zwei Individuen der Domäne D –, um dann eine Formel zu bilden, der ein Wahrheitswert zugewiesen werden kann. Man kann *lesen* somit als Funktion betrachten, die zwei Individuen der Domäne auf einen Wahrheitswert abbildet.

Schönfinkel-Darstellung

Um eine einheitliche Behandlung von Prädikaten unabhängig von ihrer Stelligkeit zu erreichen, führt man eine n -stellige Funktion auf n einstellige Funktionen zurück. Wir wollen das am Beispiel der Formel *lesen(peter, duden)* illustrieren, die genau dann als wahr interpretiert wird, wenn Peter und der Duden in einer Lesens-Beziehung zueinander stehen. Statt *lesen* nun auf beide Argumente (entsprechend Subjekt und Objekt) gleichzeitig anzuwenden, wird ein Zwischenschritt eingefügt. Dazu bedienen wir uns einer neuen, einstelligen Funktion *lesen**, die nur auf das letzte Argument, das Objekt, angewandt wird. Dieser Zwischenschritt liefert also das Zwischenergebnis *lesen*(duden)*. Diese Formel soll nun so interpretiert werden, dass sie ein einstelliges Prädikat darstellt, das noch ein Argument (das Subjekt) braucht um einen Wahrheitswert zu ergeben. Dieses Prädikat soll also als die Eigenschaft des „Dudenlesens“ interpretiert werden. Wird es nun noch auf das Subjekt angewandt, ergibt sich die Formel *lesen*(duden)(peter)*. Diese ist dann wahr, wenn Peter die Eigenschaft des „Dudenlesens“ hat. Das sind dieselben Fälle, in denen die ursprüngliche Formel *lesen(peter, duden)* wahr ist, nämlich die, in denen Peter den Duden liest. Eine solche Zerlegung einer n -stelligen Funktion f in n Anwendungen einstelliger Funktionen nennt man die **Schönfinkel-Darstellung** der Funktion nach dem russischen Mathematiker und Logiker Moses Schönfinkel (1889–1942).

Definition 2.1.19

Jede n -stellige Funktion f besitzt eine **Schönfinkel-Darstellung**, so dass

$$f(x_1, \dots, x_n) = f^*(x_n)(x_{n-1}) \dots (x_1)$$

gilt. Dabei sind f^* , $f^*(x_n)$, $f^*(x_n)(x_{n-1})$, usw. einstellige Funktionen. \square

lesen(duden)(peter)* ist also die Schönfinkel-Darstellung von *lesen(peter, duden)*.

In dieser Darstellung nimmt ein Prädikat also nacheinander alle Argumente, wobei mit dem letzten angefangen wird und jeder Zwischenschritt eine einstellige Funktion liefert. Den Prozess der Umwandlung einer n -stelligen Funktion in entsprechend viele einstellige Funktionen nennt man auch **Currying** nach dem amerikanischen Mathematiker und Logiker Haskell Curry (1900–1982), dessen Arbeit auf der von Schönfinkel aufbaut.

Typen informell

Um nicht nur auf einzelne Prädikate wie **lesen** oder Formeln wie **schlafen(peter)** Bezug nehmen zu können, wollen wir das oben Beschriebene etwas allgemeiner fassen und Formeln, Individuen und Prädikate, die ja intuitiv ganz unterschiedlicher Natur sind, in verschiedene Gruppen einteilen. Dies geschieht, indem man ihnen unterschiedliche **Typen** zuordnet.

Als **Basistypen** benutzt man e für Individuen der Domäne (von engl. *entity*) und t für Wahrheitswerte (von engl. *truth value*). Von diesen beiden Basistypen ausgehend können nun die Typen aller Prädikate abgeleitet werden. Ein einstelliges Prädikat wie **schlafen** ist beispielsweise eine Funktion, die ein Individuum der Domäne – d.h. ein Argument vom Typ e – verlangt, um eine Formel – also einen Wahrheitswert vom Typ t – zu ergeben. Damit hat diese Funktion selbst den Typ $\langle e, t \rangle$. In den Typenklammern links steht also der Typ des Arguments und rechts der Typ des Ergebnisses der Funktionsanwendung. Eine andere gebräuchliche Typenschreibweise ist $(e \rightarrow t)$, die noch deutlicher macht, dass e der Argumenttyp und t der Ergebnistyp ist.

Allgemein kann man für die Anwendung einer einstelligen Funktion vom Typ $\langle e, t \rangle$ (wie **schlafen**) auf ein Argument vom Typ e (wie **peter**) schematisch folgende Gleichung aufstellen:

$$\begin{array}{rcl} \text{schlafen} & + & \text{peter} = \text{schlafen(peter)} \\ \langle e, t \rangle & + & e = t \end{array}$$

Auch der Typ für zweistellige Prädikate folgt nun direkt. In der Schönfinkel-Darstellung nimmt eine entsprechende Funktion wie **lesen*** ein Argument vom Typ e wie **duden** und liefert damit eine einstellige Funktion **lesen*(duden)** vom Typ $\langle e, t \rangle$. Damit hat die zweistellige Funktion **lesen*** selbst den Typ $\langle e, \langle e, t \rangle \rangle$ und es ergibt sich folgende informelle Gleichung:

$$\begin{array}{rcl} \text{lesen*} & + & \text{duden} = \text{lesen*(duden)} \\ \langle e, \langle e, t \rangle \rangle & + & e = \langle e, t \rangle \end{array}$$

Im nächsten Schritt kann **lesen*(duden)** nun beispielsweise auf **peter** (vom Typ e) angewendet werden, um eine Formel vom Typ t zu ergeben.

Auf diese Weise kann man nun theoretisch alle möglichen Arten von Typen erzeugen. Man kann sich etwa eine Funktion vorstellen, die ein einstelliges Prädikat (also einen Ausdruck des Typs $\langle e, t \rangle$) verlangt, um eine Formel (also etwas vom Typ t) zu ergeben. Diese Funktion hätte damit den Typ $\langle \langle e, t \rangle, t \rangle$.

Mittels Typen lässt sich nun auch Russells Mengenparadoxon aus Abschnitt 2.1.1 vermeiden. Zunächst scheint es nicht offensichtlich zu sein, wie Mengen mit den

oben angegebenen Typen in Verbindung stehen. Den Schlüssel hierzu bilden die charakteristischen Funktionen aus Definition 2.1.5. Die charakteristische Funktion einer Menge A wird auf ein Objekt x einer Grundmenge $X \supseteq A$ angewandt und gibt den Wert 1 zurück, wenn $x \in A$ gilt, und 0 sonst. Identifiziert man die möglichen Werte 0 und 1 mit Wahrheitswerten vom Typ t und nimmt an, dass die Objekte der Grundmenge X vom Typ τ sind, so ist die charakteristische Funktion entsprechend vom Typ $\langle \tau, t \rangle$. Wenn man nun eine Menge mit ihrer charakteristischen Funktion identifiziert, so hat man das Mengenparadoxon vermieden: Da eine Menge mit Elementen vom Typ τ selbst den Typ $\langle \tau, t \rangle$ hat, kann eine Menge niemals sich selbst enthalten.

Syntax der Typenlogik

Mittels der eben dargelegten Typtheorie ist man nun imstande, eine Sprache zu definieren, die über die im vorigen Abschnitt dargelegte Prädikatenlogik hinausgeht. Während in der Prädikatenlogik nur Individuenvariablen verfügbar sind und so auch nur über Individuen quantifiziert werden kann, ist in der Typenlogik Quantifikation nicht nur über Individuen, sondern auch über Prädikate und Kategorien anderer Typs erlaubt. Zunächst werden die möglichen Typen formal definiert.

Definition 2.1.20

Die Menge der **Typen** ist wie folgt definiert:

1. e ist ein Typ.
2. t ist ein Typ.
3. Sind τ und σ Typen, so ist auch $\langle \tau, \sigma \rangle$ ein Typ.

□

Wie auch in der Prädikatenlogik, stehen in der Typenlogik Variablen zur Verfügung. Allerdings gibt es hier für jeden Typ τ eine eigene Menge von Variablen $\mathcal{V}_\tau = \{v_{\tau,0}, v_{\tau,1}, v_{\tau,2}, \dots\}$.

Nun kann man – entsprechend den Termen und Formeln der Prädikatenlogik – die Menge der syntaktisch **wohlgeformten Ausdrücke** ME_τ vom Typ τ (von engl. *Meaningful Expression*) definieren. Da prädikatenlogische Terme ja für Individuen stehen, werden wir sie ab jetzt als die Menge ME_e betrachten. Prädikatenlogische Formeln entsprechen der Menge ME_t , und Prädikatssymbole entsprechen je nach Stelligkeit den Mengen $ME_{\langle e,t \rangle}$ (für einstellige Prädikate), $ME_{\langle e,\langle e,t \rangle \rangle}$ (für zweistellige Prädikate), usw.

In der Prädikatenlogik hatte man nur einen eingeschränkten Symbolvorrat von Konstantensymbolen und Prädikatssymbolen zur Verfügung. Da in der Typenlogik die Menge der Typen entsprechend erweitert wurde, kann der Symbolvorrat hier aus Konstantensymbolen jeglichen Typs bestehen. Mit diesem Symbolvorrat an Konstanten jeden Typs lassen sich nun die wohlgeformten Ausdrücke definieren:

Definition 2.1.21

Die wohlgeformten Ausdrücke ME_τ vom Typ τ der Typenlogik sind wie folgt definiert:

1. Jede Variable vom Typ τ ist ein Element von ME_τ .
2. Jede Konstante vom Typ τ ist ein Element von ME_τ .
3. Sind $\alpha \in ME_{\langle \tau, \sigma \rangle}$ und $\beta \in ME_\tau$, so ist $\alpha(\beta) \in ME_\sigma$.
4. Sind $\alpha \in ME_\tau$ und $\beta \in ME_\tau$, so ist $\alpha = \beta \in ME_t$
5. Ist $\varphi \in ME_t$, so ist auch $\neg\varphi \in ME_t$
6. Sind $\varphi \in ME_t$ und $\psi \in ME_t$, so sind auch $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, $(\varphi \leftrightarrow \psi)$ in ME_t .
7. Ist $\varphi \in ME_t$ und x eine Variable (von beliebigem Typ), so sind auch $\forall x\varphi$ und $\exists x\varphi$ in ME_t .

Die Elemente aus ME_t heißen **Formeln**. \square

Diese Definition ist im Prinzip nur eine Verallgemeinerung der Definitionen der Syntax der Prädikatenlogik. Die Punkte 1. und 2. entsprechen im Prinzip der Definition von prädikatenlogischen Termen (Definition 2.1.12) und die Punkte 3.–7. der Definition von prädikatenlogischen Formeln (Definition 2.1.13).

Um den Symbolvorrat der Prädikatenlogik aus Abschnitt 2.1.3, Seite 48f in dieses neue System zu übertragen, genügt es, die Konstanten- und Prädikatsymbole als entsprechende getypte Konstantensymbole aufzufassen. Wenn wir der gängigen Praxis folgen, jedes Konstantensymbol mit einem Subskript zu versehen, das seinen Typ angibt, so sieht der Symbolvorrat wie folgt aus:

$$\text{peter}_e, \text{johanna}_e, \text{duden}_e, \text{algebra}_e, \quad \text{schlafen}_{\langle e, t \rangle}, \text{buch}_{\langle e, t \rangle}, \quad \text{lesen}_{\langle e, \langle e, t \rangle \rangle} \quad (2.11)$$

Dieselbe Notationskonvention gilt auch für Variablen. Die folgenden werden wir ab jetzt benutzen:

$$x_e, y_e \quad P_{\langle e, t \rangle}, Q_{\langle e, t \rangle}$$

Beispiel 2.1.13

Folgende Ausdrücke sind wohlgeformte Ausdrücke gemäß obiger Definition:

- | | | |
|-----|--|----------------------------------|
| (1) | $x_e = \text{peter}_e \in ME_t$ | nach Punkt 4. |
| (2) | $\text{lesen}_{\langle e, \langle e, t \rangle \rangle}(\text{duden}_e) \in ME_{\langle e, t \rangle}$ | nach Punkt 3. |
| (3) | $\text{lesen}_{\langle e, \langle e, t \rangle \rangle}(\text{duden}_e)(x_e) \in ME_t$ | nach Punkt 3.
mit (2) |
| (4) | $(x_e = \text{peter}_e \wedge \text{lesen}(\text{duden})(x_e)) \in ME_t$ | nach Punkt 6.
mit (1) und (3) |

Weiterhin lässt sich z. B. folgende Formel herleiten, die Quantifikation höherer Ordnung illustriert (also Quantifikation über Variablen höheren Typs als e) und damit keine prädikatenlogische Entsprechung hat:

$$\forall P_{\langle e, t \rangle} (P(\text{peter}_e) \rightarrow P(\text{johanna}_e)) \quad (2.12)$$

Obwohl wir erst im nächsten Abschnitt die Semantik der Typenlogik im Detail ansehen wollen, können wir schon jetzt in etwa paraphrasieren, was die Formel aussagt: für alle P gilt: wenn P für Peter gilt, so gilt P auch für Johanna. Sieht man Elemente aus $ME_{\langle e, t \rangle}$ wieder als Eigenschaften an, so bedeutet das: Johanna hat alle Eigenschaften, die Peter hat. \square

Semantik der Typenlogik

Da es nun Konstanten von jedem Typ gibt, müssen statt einer einzigen Domäne D nun Domänen für jeden Typ zur Verfügung stehen. Man bezeichnet D_τ als die Menge der möglichen Interpretationen für wohlgeformte Ausdrücke des Typs τ . Also entspricht z. B. D_e gerade dem Individuenbereich des prädikatenlogischen Modells und D_t den beiden Wahrheitswerten *wahr* und *falsch* bzw. der Menge $\{0, 1\}$. Die Domänen für komplexe Typen $\langle \tau, \sigma \rangle$ sind, wie oben angedeutet, die Mengen aller Funktionen von Argumenten des Typs τ in Werte des Typs σ . Beispielsweise ist die Domäne $D_{\langle e, t \rangle}$ die Menge all der Funktionen, die Individuen aus D_e auf Elemente aus D_t , also Wahrheitswerte, abbilden, d.h. $D_{\langle e, t \rangle} = \{f | f : D_e \rightarrow \{0, 1\}\}$. $D_{\langle e, t \rangle}$ ist somit die Menge der charakteristischen Funktionen über der Grundmenge D_e (s. Definition 2.1.5). Identifiziert man wieder Mengen mit ihren charakteristischen Funktionen, könnte man auch sagen, dass z. B. $D_{\langle e, t \rangle}$ die Gesamtheit aller Mengen von Objekten des Typs e ist, $D_{\langle \langle e, t \rangle, t \rangle}$ die Gesamtheit aller Mengen von Mengen von Objekten des Typs e , usw. Ausgehend von einem zugrundeliegenden Individuenbereich D lassen sich diese Domänen nun folgendermaßen definieren.

Definition 2.1.22

Die **Domäne D_τ des Typs τ** ist wie folgt definiert:

1. D_e ist gleich D .
2. D_t ist gleich $\{0, 1\}$.
3. $D_{\langle \tau, \sigma \rangle}$ ist die Menge aller Funktionen von D_τ nach D_σ .

\square

Variablen werden wieder mittels einer Variablenbelegung interpretiert. Die Definition einer solchen ist nahezu identisch zu der in der Prädikatenlogik (vgl. Definition 2.1.16) und unterscheidet sich nur dadurch, dass sie den Typ der Variablen beachtet.

Definition 2.1.23

Eine **Variablenbelegung der Typenlogik** ist eine Funktion g , die jeder Va-

riablen vom Typ τ ein Objekt aus D_τ zuweist, also $g(v_\tau) \in D_\tau$ für alle $v_\tau \in \mathcal{V}_\tau$.
 \square

Nun steht für jeden Typ τ also eine Domäne D_τ zur Verfügung. Man passt nun die Definition 2.1.15 eines prädikatenlogischen Modells $\mathcal{M} = (D, F)$ entsprechend an, sodass auch F die Typen beachtet.

Definition 2.1.24

Ein **Modell der Typenlogik** ist ein Paar $\mathcal{M} = (D, F)$, wobei

- D eine nicht-leere Domäne ist und
- F eine Abbildung ist, die jeder Konstanten vom Typ τ ein Element aus D_τ zuordnet.

\square

Die Interpretation bezüglich eines solchen Modells und einer Variablenbelegung wird dann wie folgt definiert.

Definition 2.1.25

Die **Interpretation eines wohlgeformten Ausdrucks der Typenlogik** bezüglich eines Modells \mathcal{M} und einer Variablenbelegung g ist wie folgt definiert:

1. $\llbracket v \rrbracket^{\mathcal{M}, g} = g(v)$ für alle Variablen v .
2. $\llbracket c \rrbracket^{\mathcal{M}, g} = F(c)$ für alle Konstanten c .
3. $\llbracket \alpha(\beta) \rrbracket^{\mathcal{M}, g} = \llbracket \alpha \rrbracket^{\mathcal{M}, g} (\llbracket \beta \rrbracket^{\mathcal{M}, g})$
4. $\llbracket \alpha = \beta \rrbracket^{\mathcal{M}, g} = 1$ gdw. $\llbracket \alpha \rrbracket^{\mathcal{M}, g} = \llbracket \beta \rrbracket^{\mathcal{M}, g}$
5. – 11. identisch mit Regeln 5.–11. von Definition 2.1.17

\square

Beispiel 2.1.14

An einem Beispiel soll verdeutlicht werden, wie die oben definierte Interpretation funktioniert. Dazu wollen wir das prädikatenlogische Lerngruppenmodell \mathcal{M} auf Seite 48 zugrunde legen, wobei wir von dem oben angepassten Symbolvorrat (2.11) ausgehen.

Die zugrundeliegende Domäne und damit die Domäne D_e für wohlgeformte Ausdrücke vom Typ e ist identisch zu der in (2.10), also

$$D = D_e = \{ \text{Peter, Hans, Johanna, Duden, Algebra II} \}.$$

Die anderen Domänen sind entsprechend Definition 2.1.22 gegeben.

Die Funktion F des typenlogischen Modells, die für die Interpretation der Konstantensymbole zuständig ist, soll auch entsprechend dem prädikatenlogischen Beispiel Zuweisungen vornehmen.

$$\begin{aligned}
F(\text{peter}_e) &= \text{Peter} & F(\text{johanna}_e) &= \text{Johanna} \\
F(\text{duden}_e) &= \text{Duden} & F(\text{algebra}_e) &= \text{Algebra II} \\
F(\text{schlafen}_{\langle e, t \rangle}) &= C_{\{\text{Peter}\}} & F(\text{buch}_{\langle e, t \rangle}) &= C_{\{\text{Duden}, \text{Algebra II}\}} \\
F(\text{lesen}_{\langle e, \langle e, t \rangle \rangle}) &= \text{die Funktion } f : D_e \rightarrow D_{\langle e, t \rangle} \text{ so dass} \\
f(x) &= \begin{cases} C_{\{\text{Peter}, \text{Johanna}\}} & \text{falls } x = \text{Duden} \\ C_{\{\text{Hans}\}} & \text{falls } x = \text{Algebra II} \\ C_{\emptyset} & \text{sonst} \end{cases}
\end{aligned}$$

Eine einfache Prädikation wie in $\text{schlafen}(\text{johanna})$ wird damit bzgl. \mathcal{M} und einer beliebigen Variablenbelegung g wie folgt interpretiert:

$$\begin{aligned}
\llbracket \text{schlafen}(\text{johanna}) \rrbracket^{\mathcal{M}, g} &= 1 \\
\text{gdw. } \llbracket \text{schlafen} \rrbracket^{\mathcal{M}, g} (\llbracket \text{johanna} \rrbracket^{\mathcal{M}, g}) &= 1 \\
\text{gdw. } F(\text{schlafen})(F(\text{johanna})) &= 1 \\
\text{gdw. } C_{\{\text{Peter}\}}(\text{Johanna}) &= 1
\end{aligned}$$

Damit ist die Formel falsch, denn die Anwendung der charakteristischen Funktion der Menge, die nur Peter enthält, liefert bei Anwendung auf das Objekt Johanna 0. Identifiziert man wieder charakteristische Funktionen mit ihren Mengen, so lässt sich letzte Zeile wie folgt umformulieren:

$$\begin{aligned}
\llbracket \text{schlafen}(\text{johanna}) \rrbracket^{\mathcal{M}, g} &= 1 \\
\text{gdw. } \text{Johanna} &\in \{\text{Peter}\}
\end{aligned}$$

Es ist gängige Praxis, beide Sichtweisen austauschbar zu benutzen, was wir auch im folgenden tun werden. Die Interpretation von Formel (2.12) sieht bzgl. dieses Modells und einer beliebigen Variablenbelegung g wie folgt aus:

$$\begin{aligned}
\llbracket \forall P(P(\text{peter}) \rightarrow P(\text{johanna})) \rrbracket^{\mathcal{M}, g} &= 1 \\
\text{gdw. für alle } h[P]g : \llbracket (P(\text{peter}) \rightarrow P(\text{johanna})) \rrbracket^{\mathcal{M}, h} &= 1 \\
\text{gdw. für alle } h[P]g : \llbracket P(\text{peter}) \rrbracket^{\mathcal{M}, h} = 0 \text{ oder } \llbracket P(\text{johanna}) \rrbracket^{\mathcal{M}, h} &= 1 \\
\text{gdw. f. a. } h[P]g : \llbracket P \rrbracket^{\mathcal{M}, h} (\llbracket \text{peter} \rrbracket^{\mathcal{M}, h}) = 0 \text{ oder } \llbracket P \rrbracket^{\mathcal{M}, h} (\llbracket \text{johanna} \rrbracket^{\mathcal{M}, h}) &= 1 \\
\text{gdw. für alle } h[P]g : h(P)(F(\text{peter})) = 0 \text{ oder } h(P)(F(\text{johanna})) &= 1 \\
\text{gdw. für alle } h[P]g : h(P)(\text{Peter}) = 0 \text{ oder } h(P)(\text{Johanna}) &= 1
\end{aligned}$$

Die letzte Zeile gilt nicht bzgl. des obigen Modells. Betrachtet man z.B. die Menge $\{\text{Peter}\}$ (für die der Symbolvorrat sogar das Symbol schlafen bereithält), so gilt für $h[P]g$ mit $h(P) = C_{\{\text{Peter}\}}$ nicht, dass $C_{\{\text{Peter}\}}(\text{Peter}) = 0$ oder $C_{\{\text{Peter}\}}(\text{Johanna}) = 1$. Im Gegenteil, es gilt sogar $C_{\{\text{Peter}\}}(\text{Peter}) = 1$ und

$C_{\{\text{Peter}\}}(\text{Johanna}) = 0$. Die Formel (2.12) ist in diesem Modell also nicht erfüllt. In anderen Worten haben wir eine Eigenschaft gefunden, die auf Peter, aber nicht auf Johanna zutrifft. Diese Eigenschaft hat die Interpretation $C_{\{\text{Peter}\}}$ und man könnte sie mit *Peter sein* paraphrasieren. Peter hat also die Eigenschaft, Peter zu sein, während Johanna diese Eigenschaft nicht hat.

In Anbetracht dieser Tatsache wäre die Formel dann wahr, wenn das Modell so beschaffen wäre, dass $F(\text{peter}) = F(\text{johanna})$ gelten würde, also wenn das Modell beiden Individuenkonstanten dasselbe Objekt d zuweisen würde, z. B. Peter oder *Duden* (man erinnere sich an dieser Stelle nochmals an die Willkürlichkeit der Symbole!). Die Forderung oben würde dann nämlich $h(P)(d) = 0$ oder $h(P)(d) = 1$ lauten, was für alle Variablenbelegungen $h[P]g$ wahr wäre. Selbst unter Betrachtung der sehr speziellen Identitätseigenschaft $C_{\{d\}}$ wäre dann die Forderung erfüllt. Zusammenfassend kann also gesagt werden, dass Formel (2.12) genau in den Modellen wahr ist, in denen *peter* und *johanna* dasselbe Individuum bezeichnen. \square

2.1.5 Der Lambda-Kalkül

Die im vorigen Abschnitt definierte Typenlogik wird nun noch um den wichtigen **Lambda-Kalkül** erweitert. Wir starten wieder mit einem Beispiel im Rahmen des Lerngruppenmodells und wollen einen wohlgeformten Ausdruck finden, dessen Interpretation die Menge aller Bücher ist, die von Peter gelesen werden. Wir suchen also einen Ausdruck aus $ME_{\langle e, t \rangle}$, da es sich bei der gesuchten Menge um eine Menge von Objekten handelt. Wir wissen, dass die Formel $\text{buch}_{\langle e, t \rangle}(x_e)$ etwa als *x ist ein Buch* und die Formel $\text{lesen}_{\langle e, \langle e, t \rangle \rangle}(x_e)(\text{peter}_e)$ etwa als *x wird von Peter gelesen* paraphrasiert werden kann. Durch Konjunktion liese sich daraus die Formel

$$(\text{buch}_{\langle e, t \rangle}(x_e) \wedge \text{lesen}_{\langle e, \langle e, t \rangle \rangle}(x_e)(\text{peter}_e)) \quad (2.13)$$

herleiten, die in etwa als *x ist ein Buch und x wird von Peter gelesen* umschrieben werden kann. Hat man damit nun gefunden, was gesucht wurde? Leider nicht. Da Ausdruck (2.13) in ME_t (d.h. vom Wahrheitswertetyp t) ist, liefert seine Interpretation einen Wahrheitswert und keine Menge von Objekten. Als Interpretation von (2.13) bekäme man also keine Menge von Objekten sondern *wahr* oder *falsch* – und das auch noch abhängig von der Variablenbelegung, da die Variable x ja frei vorkommt.

Die Lösung besteht darin, einen neuen Operator – den **Lambda-Operator** λ – einzuführen und damit den gesuchten Ausdruck aus (2.13) zu konstruieren. Dieser Operator kann, genau wie der Existenz- und der Allquantor, eine Variable binden. Man spricht in diesem speziellen Fall allerdings von einer **Variablen- oder Lambda-Abstraktion**. Analog zur Syntax eines Quantors wird der Lambda-Operator gefolgt von einer Variablen vor einen Ausdruck geschrieben.

Der semantische Effekt dieses Operators (den wir in den folgenden Abschnitten formal definieren werden) ist, aus einem Funktionswert eine Funktion zu bilden, die gerade den ursprünglichen Wert unter Verwendung des Funktionsarguments liefert. An dem einfachen mathematischen Beispiel der Funktion, die

als Wert ihr Argument quadriert, soll dies gezeigt werden. Hier würde der Ausdruck x^2 einen Wert, also eine Zahl darstellen, nämlich das Quadrat der Zahl x (was immer sie auch sein mag). Die entsprechende Funktion, die auf Argumente angewendet werden kann, würde mit $\lambda x(x^2)$ bezeichnet. Angewendet auf ein Argument wie z. B. die Zahl 5, würde sie wieder einen entsprechenden Wert liefern, also $\lambda x(x^2)(5) = 5^2 = 25$, indem sie das Argument in den Wert anstelle der abstrahierten Variable einsetzt. Im Falle von (2.13) sähe das Ergebnis der Abstraktion von x_e wie folgt aus:

$$\lambda x_e (\text{buch}_{\langle e, t \rangle}(x_e) \wedge \text{lesen}_{\langle e, \langle e, t \rangle \rangle}(x_e)(\text{peter}_e)) \quad (2.14)$$

Hier macht der Lambda-Operator aus Ausdruck (2.13) vom Typ t einen Ausdruck, der auf ein Argument vom Typ e angewendet werden kann und dann Ausdruck (2.13) zurückliefert, wobei x_e in (2.13) durch das Argument interpretiert wird. Insgesamt ist Ausdruck (2.14) damit vom Typ $\langle e, t \rangle$. Er beschreibt eine Menge (bzw. charakteristische Funktion) und zwar gerade die Menge der x , für die gilt: x ist ein Buch und x wird von Peter gelesen. Damit wurde also der gesuchte Ausdruck gefunden.

Syntax der λ -Typenlogik

Die formalen Definitionen des Lambda-Kalküls, d.h. der Syntax und Semantik der Lambda-Abstraktion werden wir im Rahmen der Typenlogik geben. Deshalb fallen die neuen Definitionen der Syntax und Semantik recht knapp aus – sie sind nur Erweiterungen der Definitionen des vorangegangenen Abschnitts zur Typenlogik. Zur Definition der Syntax der neuen Logik, die wir **λ -Typenlogik** nennen wollen, fügen wir zu Definition 2.1.21 einen weiteren Punkt hinzu.

Definition 2.1.26

Die wohlgeformten Ausdrücke ME_τ vom Typ τ der λ -Typenlogik sind wie folgt definiert:

1.–7. Wie in Definition 2.1.21.

8. Ist $\varphi \in ME_\sigma$ und x eine Variable vom Typ τ , so ist $\lambda x \varphi \in ME_{\langle \tau, \sigma \rangle}$.

□

Beispiel 2.1.15

Nach voriger Definition ist folgender Ausdruck in $ME_{\langle \langle e, t \rangle, \langle \langle e, t \rangle, t \rangle \rangle}$:

$$\lambda P_{\langle e, t \rangle} \lambda Q_{\langle e, t \rangle} \exists x_e (P(x) \wedge Q(x)) \quad (2.15)$$

Die Herleitung sieht wie folgt aus:

(1)	x_e	$\in ME_e$ wegen Def. 2.1.26, 1.
(2)	$P_{\langle e, t \rangle}$	$\in ME_{\langle e, t \rangle}$ wegen Def. 2.1.26, 1.
(3)	$Q_{\langle e, t \rangle}$.	dto.
(4)	$P(x)$	$\in ME_t$ wegen Def. 2.1.26, 3. mit (1), (2)
(5)	$Q(x)$	$\in ME_t$ wegen Def. 2.1.26, 3. mit (1), (3)
(6)	$(P(x) \wedge Q(x))$	$\in ME_t$ wegen Def. 2.1.26, 6. mit (4), (5)
(7)	$\exists x(P(x) \wedge Q(x))$	$\in ME_t$ wegen Def. 2.1.26, 7. mit (6)
(8)	$\lambda Q \exists x(P(x) \wedge Q(x))$	$\in ME_{\langle \langle e, t \rangle, t \rangle}$ wegen Def. 2.1.26, 8. mit (7)
(9)	$\lambda P \lambda Q \exists x(P(x) \wedge Q(x))$	$\in ME_{\langle \langle e, t \rangle, \langle \langle e, t \rangle, t \rangle \rangle}$ wegen Def. mit (8)

Damit ist beispielsweise folgender Ausdruck in ME_t :

$$\lambda P_{\langle e, t \rangle} \lambda Q_{\langle e, t \rangle} \exists x_e (P(x) \wedge Q(x)) (\text{buch}_{\langle e, t \rangle}) (\lambda y_e \text{lesen}_{\langle e, \langle e, t \rangle \rangle}(y) (\text{peter}_e)) \quad (2.16)$$

Er entsteht aus (2.15) durch Anwendung von Definition 2.1.26, Punkt 3. auf $\text{buch}_{\langle e, t \rangle}$ und $\lambda y_e \text{lesen}_{\langle e, \langle e, t \rangle \rangle}(y) (\text{peter}_e)$, das selbst wiederum vom Typ $\langle e, t \rangle$ ist.
□

Semantik der λ -Typenlogik

Was die Semantik betrifft, so dienen als Modelle gerade die Modelle der Typenlogik aus Definition 2.1.24. Auch die Variablenbelegungen sind dieselben wie in Definition 2.1.23. Mit dem Lambda-Operator ist es nun aber möglich, neue Funktionen zu bilden. Entsprechend wird die Definition der Interpretation um einen Punkt erweitert.

Definition 2.1.27

Die **Interpretation eines wohlgeformten Ausdrucks der λ -Typenlogik** bezüglich eines Modells \mathcal{M} und einer Variablenbelegung g ist wie folgt definiert:

1.–11. Wie in Definition 2.1.25.

12. Ist $\lambda x \varphi \in ME_{\langle \tau, \sigma \rangle}$, dann ist $\llbracket \lambda x \varphi \rrbracket^{\mathcal{M}, g}$ die Funktion $H \in D_{\langle \tau, \sigma \rangle}$, für die gilt: $H(u) = \llbracket \varphi \rrbracket^{\mathcal{M}, h}$, wobei $h[x]g$ und $h(x) = u$.

□

Diese Definition besagt also, dass für einen Ausdruck $\lambda x \varphi$ aus $ME_{\langle \tau, \sigma \rangle}$ die Interpretation $\llbracket \lambda x \varphi \rrbracket^{\mathcal{M}, g}$ eine Funktion ist, die, angewendet auf ein Element u aus D_τ , gerade die Interpretation von φ ergibt, in der ein mögliches freies Vorkommen von x in φ durch dieses u interpretiert wird. Wir können also durch die Anwendung eines Lambda-Ausdrucks eine bestimmte Interpretation einer bis dahin noch freien Variablen erzwingen.

Für den Lambda-Kalkül gibt es einige syntaktische Umformungsmöglichkeiten. Allerdings muss man sorgfältig darauf achten, dass durch die Umformungen nicht versehentlich freie Variablen gebunden werden würden (wobei nun zusätzlich zu den Quantoren \exists und \forall auch der Lambda-Operator λ Variablen binden kann). In folgenden Punkten soll $\varphi[\psi/x]$ den Ausdruck φ nach Ersetzung aller

freien Vorkommen von x durch den Ausdruck ψ bezeichnen. Dabei werden wir den Äquivalenzbegriff aus den vorigen Abschnitten von Formeln auf Ausdrücke beliebigen Typs erweitern und $\varphi \equiv \psi$ schreiben, wenn $\llbracket \varphi \rrbracket^{\mathcal{M}, g} = \llbracket \psi \rrbracket^{\mathcal{M}, g}$ bzgl. aller Modelle \mathcal{M} und Belegungen g gilt. Obwohl der Aufbau komplexer Ausdrücke durch die Typisierung eindeutig ist, werden wir im folgenden zusätzliche Klammern zur Steigerung der Lesbarkeit verwenden und keine Typen angeben.

α -Konversion: Für einen Ausdruck φ und Variablen x und y vom Typ τ gilt

$$\lambda x \varphi \equiv \lambda y (\varphi[y/x]), \quad (2.17)$$

wenn y nicht frei in φ vorkommt. Durch α -Konversion kann man also die abstrahierte Variable umbenennen.

β -Reduktion: Für jeden wohlgeformten Ausdruck der Form $(\lambda x \varphi)(\psi)$ gilt

$$(\lambda x \varphi)(\psi) \equiv \varphi[\psi/x], \quad (2.18)$$

wenn gilt: falls x in φ innerhalb des Skopos eines Quantors steht, der eine freie Variable aus ψ bindet, so ist x selbst gebunden (s. Beispiel 2.1.16).

Man nennt einen Ausdruck der Form $(\lambda x \varphi)(\psi)$ auch **β -Redex** (von engl. *reducible expression*). Den Übergang zu $\varphi[\psi/x]$ bezeichnet man entsprechend als β -Reduktion oder allgemeiner auch als **λ -Konversion**.

η -Reduktion: Für einen Ausdruck φ vom Typ $\langle \tau, \sigma \rangle$ und eine Variable vom Typ τ gilt

$$\lambda x (\varphi(x)) \equiv \varphi, \quad (2.19)$$

wenn x nicht frei in φ vorkommt.

Die β -Reduktion erlaubt es, auf syntaktischer Seite eine Vereinfachung durchzuführen: Statt einen β -Redex direkt über die Semantik interpretieren zu müssen, kann man ihn durch β -Reduktion umformen, bis man einen Ausdruck erhält, der keinen β -Redex mehr enthält. Ein solcher Ausdruck befindet sich in **β -Normalform**.

Beispiel 2.1.16

Die oben erwähnte versehentliche Bindung würde z. B. im Falle des folgenden β -Redex Zustände kommen:

$$(\lambda v_t \exists y_e (\text{buch}(y) \wedge v_t))(\text{lesen}(y)(\text{peter}))$$

Würde man die Bedingungen für gebundene Variablen außer Acht lassen, würde die β -Reduktion zum Ausdruck

$$\exists y_e (\text{buch}(y) \wedge \text{lesen}(y)(\text{peter}))$$

führen. Die Variable y in $\text{lesen}(y)(\text{peter})$ würde hierbei versehentlich durch den Existenzquantor gebunden werden, da v_t im Skopus des Existenzquantors frei

vorkommt. Eine praktische Lösung besteht natürlich darin, alle frei vorkommenden Variablen eines β -Redex vor der β -Reduktion entsprechend umzubenennen.

Ein Beispiel für eine η -Reduktion wäre $\lambda y \text{schlafen}(y) \equiv \text{schlafen}$. Würden beide Ausdrücke beispielsweise auf `peter` angewandt, wäre das Ergebnis (nach β -Reduktion beim ersten Ausdruck) `schlafen(peter)`.

Zur Illustration des Zusammenhangs in (2.18) sollen die Interpretationen des Ausdrucks $(\lambda x \text{buch}(x))(\text{algebra})$ und seiner β -Normalform `buch(algebra)` bzgl. eines beliebigen Modells \mathcal{M} und einer beliebigen Variablenbelegung g verglichen werden. Es gilt:

$$\begin{aligned}
 & \llbracket \lambda x \text{buch}(x)(\text{algebra}) \rrbracket^{\mathcal{M}, g} \\
 &= \llbracket \lambda x \text{buch}(x) \rrbracket^{\mathcal{M}, g} (\llbracket \text{algebra} \rrbracket^{\mathcal{M}, g}) \\
 &= \llbracket \lambda x \text{buch}(x) \rrbracket^{\mathcal{M}, g} (F(\text{algebra})) \\
 &= H(F(\text{algebra})) \text{ wobei } H \in D_{\langle e, t \rangle} \text{ mit: } H(u) = \llbracket \text{buch}(x) \rrbracket^{\mathcal{M}, h}, \\
 & \quad \text{wobei } h[x]g \text{ und } h(x) = u \\
 &= \llbracket \text{buch}(x) \rrbracket^{\mathcal{M}, h}, \text{ wobei } h[x]g \text{ und } h(x) = F(\text{algebra}) \\
 &= \llbracket \text{buch} \rrbracket^{\mathcal{M}, h} (\llbracket x \rrbracket^{\mathcal{M}, h}), \text{ wobei } h[x]g \text{ und } h(x) = F(\text{algebra}) \\
 &= F(\text{buch})(h(x)), \text{ wobei } h[x]g \text{ und } h(x) = F(\text{algebra}) \\
 &= F(\text{buch})(F(\text{algebra})) \\
 \\
 & \llbracket \text{buch}(\text{algebra}) \rrbracket^{\mathcal{M}, g} \\
 &= \llbracket \text{buch} \rrbracket^{\mathcal{M}, g} (\llbracket \text{algebra} \rrbracket^{\mathcal{M}, g}) \\
 &= F(\text{buch})(F(\text{algebra}))
 \end{aligned}$$

Wie von (2.18) vorausgesagt sind die Interpretation beider Formeln gleich, d.h. die Formeln sind äquivalent.

Als weiteres Beispiel für die β -Reduktion soll nochmals der wohlgeformte Ausdruck (2.16) dienen. Vor der Interpretation führt man insgesamt drei β -Reduktionen durch, bis man den Ausdruck in β -Normalform gebracht hat:

$$\begin{aligned}
 & \lambda P \lambda Q \exists x (P(x) \wedge Q(x))(\text{buch})(\lambda y \text{lesen}(y)(\text{peter})) \\
 & \equiv \lambda Q \exists x (\text{buch}(x) \wedge Q(x))(\lambda y \text{lesen}(y)(\text{peter})) \\
 & \equiv \exists x (\text{buch}(x) \wedge (\lambda y \text{lesen}(y)(\text{peter}))(x)) \\
 & \equiv \exists x (\text{buch}(x) \wedge \text{lesen}(x)(\text{peter}))
 \end{aligned}$$

Die Interpretation bezüglich des Lerngruppenmodells und einer beliebigen Variablenbelegung g lässt sich nun wie folgt berechnen:

$$\llbracket \exists x (\text{buch}(x) \wedge \text{lesen}(x)(\text{peter})) \rrbracket^{\mathcal{M}, g} = 1$$

gdw. für mind. ein $h[x]g$ gilt: $\llbracket (\text{buch}(x) \wedge \text{lesen}(x)(\text{peter})) \rrbracket^{\mathcal{M}, h} = 1$

gdw. für mind. ein $h[x]g$ gilt: $\llbracket \text{buch}(x) \rrbracket^{\mathcal{M}, h} = 1$ und $\llbracket \text{lesen}(x)(\text{peter}) \rrbracket^{\mathcal{M}, h} = 1$

gdw. für mind. ein $h[x]g$ gilt: $\llbracket \text{buch} \rrbracket^{\mathcal{M}, h}(\llbracket x \rrbracket^{\mathcal{M}, h}) = 1$

und $\llbracket \text{lesen} \rrbracket^{\mathcal{M}, h}(\llbracket x \rrbracket^{\mathcal{M}, h})(\llbracket \text{peter} \rrbracket^{\mathcal{M}, h}) = 1$

gdw. für mind. ein $h[x]g$ gilt: $F(\text{buch})(h(x)) = 1$

und $F(\text{lesen})(h(x))(F(\text{Peter})) = 1$

gdw. für mind. ein $h[x]g$ gilt: $h(x) \in \{\text{Duden}, \text{Algebra II}\}$

und Peter $\in F(\text{lesen})(h(x))$

Damit ist (2.16) bzgl. des Lerngruppenmodells wahr, denn man kann eine Belegung $h[x]g$ finden, für die die letzte Zeile gilt, nämlich mittels $h(x) = \text{Duden}$. Paraphrasiert sagt die Formel, dass ein Objekt existiert, das ein Buch ist und das von Peter gelesen wird. \square

Wir haben damit ein Beispiel einer Formel gegeben, die im Prinzip aus drei Komponenten bestand – dem Ausdruck (2.15), und den Argumenten *buch* und $\lambda y \text{lesen}(y)(\text{peter})$. Wenn man jetzt den Ausdruck (2.15) als logische Übersetzung des natürlichsprachlichen Artikels *Ein*, das erste Argument als Übersetzung des Nomens *Buch* und das zweite Argument als Übersetzung von *wird von Peter gelesen* versteht, so hat man damit schon einen ersten Eindruck, wie sich die Semantik des natürlichsprachlichen Satzes *Ein Buch wird von Peter gelesen* aus seinen Bestandteilen berechnen lässt. Dies wird Thema des Semantik-Unterkapitels 3.6 sein.

2.1.6 Literaturhinweise

Ein Standardwerk für eine mathematische Einführung in die Prädikatenlogik ist Ebbinghaus, Flum und Thomas (1992). Für eine weniger mathematische Herangehensweise bietet Schöning (1995) eine gute Einführung. Darstellungen dieser beiden Logiken sind auch in Partee, ter Meulen und Wall (1990) zu finden. Dieses Buch enthält auch eine Einführung in die Mengenlehre und den Lambda-Kalkül. Unsere Darstellung der Logik höherer Stufe und des Lambda-Kalküls orientiert sich an Dowty, Wall und Peters (1981), das im Rahmen der Einführung in die Montague-Semantik nach und nach immer mächtigere Logiken einführt. Ähnlich gehen auch die Autoren von Gamut (1987) vor. Im Hinblick auf den Lambda-Kalkül bietet Barendregt (1992) eine gute Möglichkeit, mathematische tiefergehende Zusammenhänge nachzulesen. In den ersten Kapiteln von Blackburn und Bos (2005) wird gezeigt, wie logische Konzepte wie Modelle und Folgerungen in der Programmiersprache PROLOG implementiert können. Außerdem ist dort eine einführende Beschreibung der Tableaux-Methode für die Aussagen- und Prädikatenlogik samt Implementierung zu finden.

2.2 Automatentheorie und Formale Sprachen

Ralf Klabunde

Die Automatentheorie und die Theorie der formalen Sprachen sind Teilbereiche der theoretischen Informatik, die für die Computerlinguistik von großer Bedeutung sind. Für die maschinelle Verarbeitung natürlicher Sprache ist es notwendig, die jeweilige Sprache bzw. die relevanten Ausschnitte in eine Hierarchie formaler Sprachen einzuordnen, um zu wissen, mit welchen Mitteln und wie effizient diese Sprache bzw. dieser Sprachausschnitt analysiert werden kann. Die hierfür notwendigen Konzepte und Aussagen zur strukturellen und Verarbeitungskomplexität liefert die theoretische Informatik.

Die Eigenschaften der in diesem Kapitel vorgestellten formalen Sprachen, Grammatiken und Automaten sind jedoch nicht nur für Theoretiker interessant, sondern auch für die praktische Realisierung entsprechender Parser oder Generierer. Während in der Linguistik die Theorie der formalen Sprachen primär ein formales Standbein für die jeweilige (Syntax-) Theorie darstellt, stellen insbesondere die Automatentheorie sowie die regulären Sprachen nützliche Konzepte für die Computerlinguistik bereit.

Auf formale Beweise wird in diesem Beitrag weitgehend verzichtet. Allerdings können insbesondere konstruktive Beweise helfen, aus einem Automaten eine Grammatik zu konstruieren oder umgekehrt. Wo solche Beweise sinnvoll sind, werden sie daher semi-formal dargestellt. Sämtliche relevanten Beweise gehören zum Standardrepertoire der Grundlagen der theoretischen Informatik und können z. B. in Bucher und Maurer (1984) und Hopcroft und Ullman (1994) nachgelesen werden.

2.2.1 Grundlegende Definitionen

Zuerst müssen die notwendigen Basisbegriffe eingeführt werden, auf denen dieses gesamte Unterkapitel aufbaut.

Ein **Alphabet** ist eine nicht-leere Menge. Die Elemente eines Alphabets werden **Zeichen** genannt. Alphabete werden im Folgenden immer mit Σ (Sigma), Φ (Phi) oder Γ (Gamma) bezeichnet. Eine Folge $x_1 \dots x_n$ von Zeichen $x_i \in \Sigma$ eines Alphabets Σ heißt ein **Wort der Länge n über Σ** , also $|x_1 \dots x_n| = n$. Das Wort der Länge 0 wird **leeres Wort** genannt und mit ε (Epsilon) bezeichnet. Das leere Wort ist ein konkret vorhandenes Wort. Aus diesem Grund sind die Mengen $\{\varepsilon\}$ und \emptyset verschieden.

Die Menge aller Worte über einem Alphabet Σ heißt der **Stern von Sigma** und wird als Σ^* (sprich „Sigma Stern“) bezeichnet. Ist z. B. $\Sigma = \{a\}$, dann ist $\Sigma^* = \{\varepsilon, a, aa, aaa, \dots\}$. Ist hingegen $\Sigma = \{a, b\}$, dann ist $\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, \dots\}$. Σ^* wird auch manchmal nach dem Mathematiker Steven Cole Kleene als **Kleene-Stern** bezeichnet. Als **Kleene-Plus** oder **Sigma-Plus** wird die Menge $\Sigma^+ = \Sigma^* - \{\varepsilon\}$ bezeichnet. Σ^+ ist somit die Menge aller nicht-leeren Wörter. Eine **formale Sprache** ist dann definiert als eine Teilmenge von Σ^* .

Die einfachste Operation mit Wörtern ist die **Konkatenation** oder **Verkettung**, notiert als \bullet . Meistens wird das Verkettungssymbol \bullet einfach weggelassen. Ist z. B. $w_1 = ab$ und $w_2 = bb$, dann ist die Verkettung von w_1 mit $w_2 = ab \bullet bb$ (oder einfach nur $abbb$) ein Wort aus $\{a, b\}^*$. Die Verkettung kann auch auf Mengen angewandt werden. Wenn $M, N \subseteq \Sigma^*$ gilt, dann ist $M \bullet N = \{u \bullet v \mid u \in M \text{ und } v \in N\}$.

Schließlich ist es für kürzere Notationen sinnvoll, die Potenzschreibweise für ein Wort w einzuführen. So kann z. B. statt $aaabbbbababab$ die kürzere Notation $a^3b^4(ab)^3$ verwendet werden.

Zuletzt werden noch die Begriffe des Algorithmus sowie der Berechenbarkeit und Entscheidbarkeit benötigt. Intuitiv ist ein **Algorithmus** eine deterministische Prozedur, die, mechanisch angewandt, zur Lösung eines Problems führt. Ein Algorithmus sollte aus diskreten Schritten bestehen und endlich beschreibbar sein. Für den Begriff „Algorithmus“ existiert keine formal präzise Definition im mathematischen Sinn. Es gibt aber mehrere Versuche, diesen Begriff mathematisch zu explizieren. Ein Ergebnis dieses Versuchs stellt die Turingmaschine dar, die in diesem Unterkapitel noch vorgestellt wird. Ein Algorithmus kann als Konstrukt angesehen werden, das einer Turingmaschine mit besonderen Eigenschaften entspricht.

Eine Funktion ist **berechenbar**, wenn für jedes Argument der Funktion der Wert in endlich vielen Schritten bestimmt wird. Sonst heißt sie **nicht-berechenbar**. Ein Problem ist **entscheidbar**, wenn ein Algorithmus vorliegt, der bei Eingabe einer Instantiierung des Problems immer dessen Lösung oder Nicht-Lösung angibt. Sonst ist ein Problem nicht entscheidbar. Die beiden Begriffe Berechenbarkeit und Entscheidbarkeit hängen eng zusammen und werden in Abschnitt 2.2.6 noch einmal aufgegriffen.

2.2.2 Grammatiken

Grammatiken erzeugen Worte, und die Menge aller von einer Grammatik erzeugten Worte bilden eine formale Sprache. Als Erzeugungsmechanismus wird eine endliche Menge R von Regeln angegeben, mit deren Hilfe unter Rückgriff auf zwei Alphabete eine prinzipiell abzählbare Menge von Wörtern erzeugt wird. Diese Grammatiken arbeiten binär, denn entweder ist eine bestimmte Zeichenkette generierbar und gehört damit zu der von der Grammatik erzeugten Sprache oder nicht. Nur bedingt akzeptable Zeichenketten sind für diese Grammatiken nicht definiert.

Formal werden Grammatiken als Quadrupel definiert. Sie bestehen aus zwei Alphabeten, einem Alphabet Σ so genannter **Terminalsymbole** und einem Alphabet Φ von **Nichtterminalsymbolen** oder **Variablen**. Beide Mengen sind disjunkt. Weiterhin wird ein **Startsymbol** $S \in \Phi$ benötigt sowie eine **Regelmenge** R zur Generierung der aus Terminalsymbolen bestehenden Zeichenketten.

Definition 2.2.1

Eine Grammatik $G = \langle \Phi, \Sigma, R, S \rangle$ besteht aus

1. Einem Alphabet Φ von Nichtterminalsymbolen,
2. Einem Alphabet Σ von Terminalsymbolen mit $\Phi \cap \Sigma = \emptyset$,

3. Einer Menge $R \subseteq \Gamma^* \times \Gamma^*$ von Ersetzungsregeln $\langle \alpha, \beta \rangle$ (Γ ist das Gesamtalphabet $\Phi \cup \Sigma$), wobei zusätzlich gilt: $\alpha \neq \varepsilon$ und $\alpha \notin \Sigma^*$,
4. Einem Startsymbol $S \in \Phi$.

□

Regeln sind nach dieser Definition Paare von Zeichenketten $\langle \alpha, \beta \rangle$. Statt $\langle \alpha, \beta \rangle$ werden Grammatikregeln üblicherweise als $\alpha \rightarrow \beta$ geschrieben.

Diese Definition einer Grammatik legt eine so genannte **allgemeine Regelgrammatik** (auch **Typ-0-Grammatik** genannt) fest, deren einzige Bedingung für die einzelnen Regeln in der Regelmenge ist, dass mindestens ein nichtterminales Symbol durch eine beliebige Zeichenkette über dem Gesamtalphabet Γ ersetzt wird. Das Wort α darf also weder das leere Wort sein noch ein Wort, das nur aus Terminalsymbolen besteht. Auf Grund dieser Bedingung können die von allgemeinen Regelgrammatiken erzeugten Sprachen sehr komplex sein. In diesem Unterkapitel werden jedoch noch weitere Grammatiktypen mit wesentlich spezifischeren Regeldefinitionen angegeben. Dabei wird noch deutlich werden, dass eine direkte Beziehung zwischen diesen Grammatiktypen, verschiedenen formalen Sprachen und entsprechenden Automaten besteht.

Die von einer Grammatik beschriebene formale Sprache wird als die Menge derjenigen Zeichenketten festgelegt, die durch Regelanwendungen aus dem Startsymbol abgeleitet werden können:

Definition 2.2.2

Sei $G = \langle \Phi, \Sigma, R, S \rangle$ eine Grammatik und seien $u, v \in (\Phi \cup \Sigma)^* = \Gamma^*$.

1. v ist aus u **direkt ableitbar** (notiert als: $u \Rightarrow v$), falls gilt:
 $u = u_1 w u_2, v = u_1 z u_2$ und $w \rightarrow z$ ist eine Regel aus R .
2. v ist aus u **ableitbar** (notiert als: $u \xrightarrow{*} v$), falls es Wörter u_0, \dots, u_k gibt ($k \geq 0$), so dass $u = u_0, v = u_k$ und $u_{i-1} \Rightarrow u_i$ ($1 \leq i \leq k$) gilt. v ist also aus u ableitbar, wenn es Zwischenwörter gibt, die jeweils direkt ableitbar sind.

□

Ableitungen lassen sich graphisch als Bäume darstellen. Bäume sind besondere Graphen und werden im folgenden Unterkapitel 2.3 formal definiert. Grammatiken lassen die Beziehungen zwischen Ableitungen und Bäumen oft uneindeutig.

Beispiel 2.2.1

Dies zeigt die folgende Grammatik $G_1 = \langle \{S, NP, EN, VP, V, Pron, N\}, \{\text{Heinz, Auftritt, seinen, inszeniert}\}, R, S \rangle$ mit der Regelmenge $R = \{S \rightarrow NP\ VP, NP \rightarrow EN, NP \rightarrow Pron\ N, VP \rightarrow V\ NP, EN \rightarrow \text{Heinz}, N \rightarrow \text{Auftritt}, V \rightarrow \text{inszeniert}, Pron \rightarrow \text{seinen}\}$.

Das Wort *Heinz inszeniert seinen Auftritt* wird unter anderem durch die folgenden zwei Ableitungen erzeugt:

- (1) $S \Rightarrow NP VP \Rightarrow EN VP \Rightarrow Heinz VP \Rightarrow Heinz inszeniert NP \Rightarrow Heinz inszeniert Pron N \Rightarrow Heinz inszeniert seinen N \Rightarrow Heinz inszeniert seinen Auftritt$
- (2) $S \Rightarrow NP VP \Rightarrow NP V Pron N \Rightarrow NP V Auftritt \Rightarrow NP V seinen Auftritt \Rightarrow NP inszeniert seinen Auftritt \Rightarrow EN inszeniert seinen Auftritt \Rightarrow Heinz inszeniert seinen Auftritt$

□

Beiden Ableitungen entspricht graphisch der in Abbildung (2.1) dargestellte Baum. Es gibt also im Allgemeinen keine 1:1-Beziehung zwischen einer Ableitung und der Baumdarstellung.

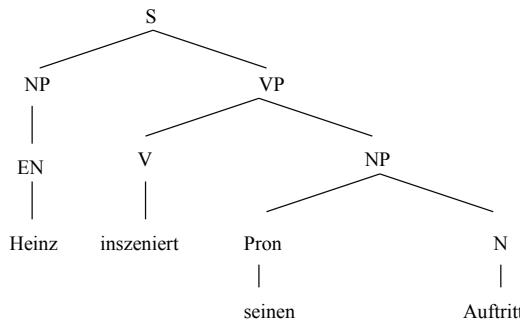


Abbildung 2.1: Ableitungsbaum für das Wort *Heinz inszeniert seinen Auftritt*

Eine Zeichenkette oder ein Wort gehört zu der von einer Grammatik erzeugten Sprache, wenn es aus dem Startsymbol ableitbar ist und nur aus Terminalsymbolen besteht.

Definition 2.2.3

Sei $G = \langle \Phi, \Sigma, R, S \rangle$ eine Grammatik. Dann heißt

$$L(G) = \{w \in \Sigma^* \mid S \xrightarrow{*} w\}$$

die von G erzeugte formale Sprache. □

Diese Definition besagt nicht, dass für jede Sprache nur eine Grammatik angegeben werden kann. Das Gegenteil ist der Fall: Für jede Sprache können im

Prinzip beliebig viele Grammatiken angegeben werden. Für eine Sprache $L(G_i)$, die von der Grammatik G_i erzeugt wird, kann auch eine andere Grammatik G_j formuliert werden mit $L(G_i) = L(G_j)$. In diesem Fall, dass zwei Grammatiken dieselbe Sprache erzeugen, heißen die beiden Grammatiken **äquivalent**.

Beispiel 2.2.2

Die folgenden zwei Grammatiken sind äquivalent, da sie beide die Sprache $L(G_1) = L(G_2) = \{a\}^*$ erzeugen:

$$G_1 = \langle \{S_1\}, \{a\}, \{S_1 \rightarrow \varepsilon, S_1 \rightarrow aS_1\}, S_1 \rangle$$

$$G_2 = \langle \{S_2\}, \{a\}, \{S_2 \rightarrow \varepsilon, S_2 \rightarrow a, S_2 \rightarrow aS_2a\}, S_2 \rangle \quad \square$$

Während Grammatiken Sprachen erzeugen, werden Worte einer Sprache mittels **Automaten** erkannt. Die Erkennung von Worten bedeutet, dass ein gegebenes Wort analysiert wird, und es wird entschieden, ob dieses Wort zu einer von einer Grammatik festgelegten Sprache gehört oder nicht. Im einfachsten Fall wird bei der Erkennung nur mitgeteilt, ob das jeweilige Wort als Element der jeweiligen formalen Sprache akzeptiert wurde oder nicht. Neben der bloßen Information über Akzeptanz kann bei der Erkennung auch eine Ausgabe „angefertigt“ werden. Im Fall einer Ausgabe werden die Automaten als **Maschinen** bezeichnet.

2.2.3 Endliche Automaten, einseitig-lineare Grammatiken und reguläre Sprachen

In diesem Abschnitt sollen diejenigen Konzepte vorgestellt werden, die mit den so genannten regulären Sprachen assoziiert sind. Dies sind die endlichen Automaten und einseitig-linearen Grammatiken. Endliche Automaten sind aber nicht nur aus einer theoretischen Perspektive interessant, sie stellen auch grundlegende und einfache Konzepte für viele Anwendungen in der maschinellen Sprachverarbeitung bereit, so z. B. in der regulären Morphologie (Unterkapitel 3.3), in der Computerphonologie (Unterkapitel 3.1) oder beim Chunk-Parsing (Unterkapitel 3.4). Die regulären Sprachen werden durch reguläre Ausdrücke beschrieben, die wiederum für viele computerlinguistische Verfahren eingesetzt werden.

Reguläre Sprachen

Mengen, die durch die Operationen der Vereinigung, Konkatenation und Sternbildung entstehen, heißen **reguläre Mengen**. Eine reguläre Menge ist in dem folgenden Beispiel angegeben.

Beispiel 2.2.3

$$(\{\text{manche}\} \bullet \{\text{Menschen}\} \bullet \{\text{sehen}\} \bullet \{\text{schlecht}\}) \cup (\{\text{ein, das}\} \bullet \{\text{Auto, Geschäft}\} \bullet \{\text{träumt, angelt}\} \bullet \{\text{unter, über}\} \bullet \{\text{einem, dem}\} \bullet \{\text{Stuhl, Menschen}\}) \\ \square$$

Dieser Ausdruck beschreibt eine Menge korrekter deutscher Sätze wie z. B. *manche Menschen sehen schlecht* oder *das Auto angelt unter dem Stuhl*. Reguläre Mengen können über **reguläre Ausdrücke** definiert werden.

Definition 2.2.4

Es sei $\Sigma = \{a_1, a_2, \dots, a_n\}$. Die folgenden Ausdrücke über Σ sind regulär:

1. \emptyset ist regulär.
2. $\{a_i\}$ ($1 \leq i \leq n$) ist regulär.
3. Wenn die Mengen L_1 und L_2 regulär sind, dann auch $(L_1 \cup L_2)$.
4. Wenn die Mengen L_1 und L_2 regulär sind, dann auch $(L_1 \bullet L_2)$.
5. Ist die Menge L regulär, dann auch L^* .

□

Eine formale Sprache heißt regulär, wenn sie durch einen regulären Ausdruck beschrieben werden kann. Als Beispiel mögen die folgenden Sprachen L_1, L_2 und L_3 dienen. Sie sind alle **reguläre Sprachen** über dem Alphabet $\Sigma = \{a, b, c\}$. Reguläre Sprachen werden auch als **Typ-3-Sprachen** bezeichnet.

Beispiel 2.2.4

$$L_1 = (((\{a\} \bullet \{b\}) \cup \{c, a\}^*) \bullet \{c\})$$

$$L_2 = \Sigma^*$$

$$L_3 = (((\{a\} \bullet (\{c\} \cup \{a\})) \cup \{b\}^*) \bullet ((\{c\} \cup \{b\}^*) \bullet \{a\})) \quad \square$$

Um übersichtlichere reguläre Ausdrücke mit wenigen Klammern zu erhalten, gilt die folgende Vorrangregel: $*$ geht vor \bullet geht vor \cup .

Reguläre Sprachen sind in ihrer Ausdrucksfähigkeit sehr beschränkt. Aufgrund ihres einfachen Aufbaus können reguläre Sprachen nicht Information über die Länge von Zeichenketten weiterreichen. Dies bedeutet, dass insbesondere Ausdrücke mit Klammerstrukturen nicht mehr regulär sind. Klammerstrukturen sind – analog zu sich symmetrisch öffnenden und schließenden Klammern – alle symmetrisch auftauchenden Symbolvorkommen. Daher ist $L = \{a^i b a^i \mid i \in \mathbb{N}_0\}$ die prototypische nicht mehr reguläre Sprache. Worte dieser Sprache L bestehen aus einer bestimmten Anzahl von Vorkommen des Symbols a gefolgt von einem b und wieder gefolgt von derselben Anzahl des Symbols a . Diese Sprache lässt sich nicht als regulärer Ausdruck angeben.

Reguläre Ausdrücke in der Computerlinguistik

Aus einer theoretischen Perspektive werden mittels regulärer Ausdrücke reguläre Sprachen beschrieben. Aber reguläre Ausdrücke spielen auch in Programmiersprachen wie Perl oder Python eine Rolle, die für viele computerlinguistische Bereiche als Standardprogrammiersprachen anzusehen sind (siehe hierzu Unterkapitel 3.9). Die Verwendung regulärer Ausdrücke ermöglicht z. B. die effektive Suche in Korpora nach diversen Ausdrücken.

Allerdings sind die in diesen Programmiersprachen verwendeten Notationen für reguläre Ausdrücke umfassender als die Angaben zu regulären Ausdrücken in der obigen Definition 2.2.4. Dies heißt nicht, dass die regulären Ausdrücke in Python oder Perl nicht in die formale Notation übersetzt werden können, aber für eine praktikable Darstellung wären diese Übersetzungen nicht hilfreich. Tabelle 2.6 stellt nach Richter (2004) einige erweiterte reguläre Ausdrücke (ERAs), die z. B. auch Python verwendet, der formalen Notation gegenüber.

Ausdruck	formale Notation	ERA
einzelnes Zeichen z. B. a	a	a
leere Menge	\emptyset	– fehlt –
Konkatenation, z. B. ab	ab	ab
Sternbildung bzgl. Ausdruck x	$\{x\}^*$	x*
Quantifizierer: optionaler Ausdruck	– fehlt –	x?
Quantifizierer: Sigma-Plus (mind. 1-malige Wiederholung)	$x\{x\}^*$	x+
Quantifizierer: genau n-malige Wiederholung	– fehlt –	x{n}
Quantifizierer: mindestens n-fache Wiederholung	– fehlt –	x{n,}
Quantifizierer: m- bis n-fache Wiederholung	– fehlt –	x{m,n}

Tabelle 2.6: Vergleich zwischen formaler und ERA-Notation

Als Beispiel für eine Übersetzung eines erweiterten regulären Ausdrucks in die formale Notation soll der Ausdruck $a+b?c\{2,4\}$ genommen werden. Dieser erweiterte reguläre Ausdruck gibt an, dass mindestens einmal ein a vorkommt, dann eventuell ein b und anschließend zwei bis viermal ein c , so dass dieser erweiterte reguläre Ausdruck der formalen Notation $\{a\{a\}^*cc\} \cup \{a\{a\}^*bcc\} \cup \{a\{a\}^*ccc\} \cup \{a\{a\}^*bccc\} \cup \{a\{a\}^*cccc\} \cup \{a\{a\}^*bccc\}$ entspricht. Man sieht an diesem Beispiel, dass sich Ausdrücke in der erweiterten Notation in die formale Notation grundsätzlich übersetzen lassen, dass aber die formale Notation für praktische Anwendungen wenig geeignet ist.

Einseitig-lineare Grammatiken

Reguläre Sprachen sind recht einfache Konstrukte. Für ihre Erzeugung werden daher Grammatiken benötigt, die ebenfalls nur mit einfachen Regeln operieren. Dies sind die **einseitig-linearen Grammatiken**, die auch **Typ-3-**

Grammatiken genannt werden. Einseitig-linear bedeutet, dass der Ableitungsbaum für ein Wort nur auf einer Seite expandiert. Einseitig-lineare Grammatiken sind entweder **links-linear** oder **rechts-linear**.

Definition 2.2.5

Eine Grammatik $G = \langle \Phi, \Sigma, R, S \rangle$ heißt rechts-linear (bzw. links-linear), falls alle Regeln von der Form

- $A \rightarrow w$ oder
- $A \rightarrow wB$ (bzw. $A \rightarrow Bw$)

mit $A, B \in \Phi$ und $w \in \Sigma^*$ sind. \square

Beispiel 2.2.5

Die folgende rechts-lineare Grammatik G mit $\Phi = \{S, A1, A2, A3, A4\}$, $\Sigma = \{\text{un}, \text{be}, \text{lehr}, \text{bar}, \text{keit}\}$, dem Startsymbol S und der Regelmenge $R = \{S \rightarrow \text{un } S, S \rightarrow \text{lehr } A2, S \rightarrow \text{be } A1, A1 \rightarrow \text{lehr } A2, A2 \rightarrow \text{bar } A3, A3 \rightarrow \text{keit } A4, A3 \rightarrow \varepsilon, A4 \rightarrow \varepsilon\}$ erzeugt aus den in Σ angegebenen Morphemen als Symbolen die bildbaren Derivationen. Der strikt nach rechts expandierende Ableitungsbaum für das Wort *unbelehrbar* ist in Abbildung 2.2 angegeben. \square

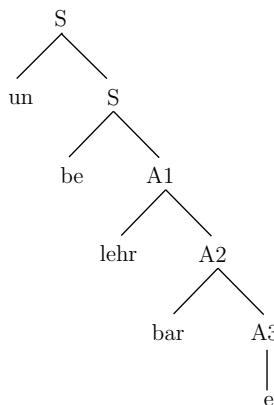


Abbildung 2.2: Rechts-linearer Ableitungsbaum für das Wort *unbelehrbar*

Endliche Automaten

Nachdem einseitig-lineare Grammatiken als einfache generative Mechanismen für reguläre Sprachen vorgestellt wurden, soll jetzt dargestellt werden, wie ein Wort als Element einer regulären Sprache mittels endlicher Automaten erkannt wird.

Zur Motivation für die Definition endlicher Automaten soll das obige Beispiel aus der Morphologie dienen. Im Deutschen basiert die Derivation zum (geringen) Teil nur auf der Konkatenation. So kann das Wort *unbelehrbar* als Verkettung der

Morpheme *un* • *be* • *lehr* • *bar* analysiert werden. Andere Verkettungen dieser Morpheme resultieren aber auch in gültigen Wörtern. So ist *unlehrbar* ebenfalls eine korrekte Derivation. Damit morphologisch nicht wohlgeformte Wörter wie z. B. *keitbar* oder *unlehrkeit* nicht als Worte akzeptiert werden können, werden die Morpheme so als mögliche Übergänge zwischen Knoten modelliert, dass nur die korrekten Worte akzeptiert werden. So kann z. B. das Präfix *un-* nur mit nominalen Elementen kombiniert und *be-* kann nur mit Verben als Stämmen konkateniert werden. Endzustände werden bei allen korrekten Morphemkombinationen erreicht. Abbildung 2.3 zeigt den Graphen, der alle gültigen Derivationen mittels der fünf Morpheme angibt.

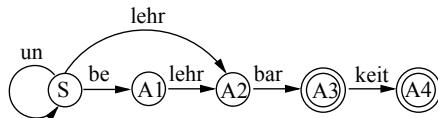


Abbildung 2.3: Akzeptanz aller mit den Morphemen *un-*, *be-*, *lehr-*, *-bar* und *-keit* gebildeten Wörter

Der Automat erhält als Eingabe ein morphologisch komplexes Wort und arbeitet es morphemweise ab. Nur wenn das Wort morphologisch korrekt ist, geht der Automat in einen Endzustand über. Dieser Graph ist wie folgt zu lesen: Jedes Wort fängt beim Startknoten *S* an. Wenn ein Doppelkreis erreicht ist, wurde ein gültiges Wort akzeptiert. Die Kante vom Startknoten zum Startknoten gibt an, dass (im Prinzip) beliebig viele *un-* akzeptiert werden, bevor entweder mittels des Präfix *be-* oder mittels des Stamms *lehr-* zum nächsten Knoten übergegangen wird. Nach dem Präfix *un-* kann das Präfix *be-* oder der Verbstamm *lehr-* als Eingabe kommen. Nach dem Verbstamm muss das Suffix *-bar* kommen, eventuell gefolgt vom Suffix *-keit*.

Dieser dargestellte Wortanalysierer ist ein Beispiel für eine bestimmte Klasse endlicher Automaten, der **deterministischen endlichen Automaten** (DEA). Für eine formale Definition deterministischer endlicher Automaten werden fünf Komponenten benötigt, die wie folgt definiert sind:

Definition 2.2.6

Ein deterministischer endlicher Automat $A = \langle \Phi, \Sigma, \delta, S, F \rangle$ besteht aus

1. Einer Menge von Zuständen Φ , dem Zustandsalphabet
2. Einem Eingabealphabet Σ mit $\Sigma \cap \Phi = \emptyset$
3. Einer Übergangsfunktion $\delta : \Phi \times \Sigma \rightarrow \Phi$
4. Einem Startzustand $S \in \Phi$
5. Einer Menge $F \subset \Phi$ von Endzuständen

□

Die Übergangsfunktion δ gibt an, welcher Folgezustand beim Lesen eines einzelnen Zeichens erreicht wird. Sie lässt sich auf die Übergangsfunktion $\delta^* : \Phi \times \Sigma^* \rightarrow \Phi$ erweitern, die festlegt, welcher Zustand beim Lesen eines Wortes erreicht wird:

$$\delta^*(T, \varepsilon) = T$$

$$\delta^*(T, wx) = \delta(\delta^*(T, w), x)$$

mit $T \in \Phi, w \in \Sigma^*$ und $x \in \Sigma$. Mittels δ^* wird die von einem Automaten akzeptierte Sprache definiert:

Definition 2.2.7

Es sei $A = \langle \Phi, \Sigma, \delta, S, F \rangle$ ein deterministischer endlicher Automat. Dann ist

$$L(A) = \{w \in \Sigma^* \mid \delta^*(S, w) \in F\}$$

die von A akzeptierte Sprache. \square

Diese Definition besagt, dass die von einem Automaten A akzeptierte Sprache durch die Menge aller möglichen Pfade durch den Automaten zu den Endzuständen bestimmt ist.

Neben den DEA gibt es noch eine andere Klasse endlicher Automaten, die **nichtdeterministischen endlichen Automaten** (NDEA). Der Unterschied zwischen diesen Automatentypen liegt in der Definition der Übergangsfunktion δ . Bei Eingabe eines Zeichens x in einem Zustand T gibt es – im Gegensatz zum DEA – im NDEA eine Menge von möglichen Nachfolgezuständen. Der Wertebereich der Übergangsfunktion $\delta : \Phi \times \Sigma \rightarrow \wp(\Phi)$ eines NDEA ist demnach die Potenzmenge von Φ . Der Wortanalysierer in Abbildung 2.4 arbeitet nicht-deterministisch, denn im Startzustand kann der Automat beim Lesen des Präfixes *unter* in zwei mögliche Zustände übergehen.

Die mehrfache Anwendung der Übergangsfunktion δ für den NDEA wird als $\delta^* : \Phi \times \Sigma^* \rightarrow \wp(\Phi)$ definiert. δ^* legt fest, in welche Zustände der NDEA beim Lesen eines Wortes übergehen kann:

$$\begin{aligned} \delta^*(T, \varepsilon) &= \{T\} \quad \text{und} \\ \delta^*(T, wx) &= \bigcup_{T' \in \delta^*(T, w)} \delta(T', x) \end{aligned}$$

für alle $T \in \Phi, w \in \Sigma^*$ und $x \in \Sigma$. Dann ist für irgendein $v \in \Sigma^*$ $\delta^*(T, v)$ die Menge derjenigen Zustände, in die der Automat bei Eingabe des Worts v übergehen kann, wenn er sich im Zustand T befindet.

Determinismus und Nicht-Determinismus sind zwei grundlegende Verhaltenskonzepte. Deterministisches Verhalten bedeutet, dass jeder Schritt vollständig vorher bestimmt ist, so dass klar ist, welcher Schritt auf einen anderen folgt. Grammatiken sind häufig nicht-deterministisch, denn ein Nichtterminalsymbol kann durch eine Menge anderer Symbole ersetzt werden. Die Ersetzungsvorschrift für ein Symbol durch andere Symbole kann also mehrere Möglichkeiten beinhalten.

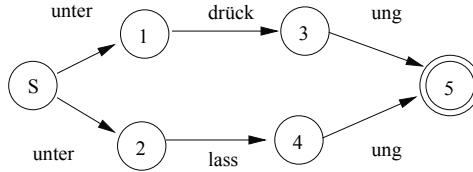


Abbildung 2.4: Ein NDEA, der zwei Nominalisierungen akzeptiert

Bei den bislang definierten Übergangsfunktionen ändert sich der Zustand der Automaten beim Lesen eines Zeichens $x \in \Sigma$. Daneben existieren auch Automaten, die auch ohne das Lesen eines Wortes in einen anderen Zustand übergehen können. Solche Automaten besitzen so genannte **ε -Übergänge**. Endliche Automaten mit ε -Übergängen sind jedoch nicht mächtiger als endliche Automaten ohne ε -Übergänge. Sie gestatten es manchmal nur, einfacheren Übergänge zu formulieren.

Endliche Automaten lassen sich auf mehrere Arten darstellen. In vielen Anwendungen reicht allein die Darstellung der Übergangsfunktion δ als Zustandsgraph. Eine weitere Möglichkeit ist die Darstellung mittels einer **Zustandstafel**. In der Zustandstafel ist für jeden Zustand festgelegt, welcher neue Zustand bei einer Eingabe erreicht wird. Bei Implementierungen von Automaten werden häufig solche Tafeln verwendet.

Die Zustandstafel für den in Abbildung 2.3 angegebenen Automaten ist in Tabelle 2.7 angegeben. Das Symbol \emptyset gibt an, dass die entsprechende Zeichen-Zustand-Kombination (also der Zustandsübergang) nicht definiert ist.

Die erste Zeile der Tabelle besagt, dass beim Lesen von *un* vom Zustand S wieder in den Zustand S gewechselt werden kann und zu keinem anderen Zustand ein Zustandsübergang definiert ist. Die zweite Zeile gibt an, dass beim Lesen von *be* im Zustand S in den Zustand A1 gewechselt wird und für die anderen Zustände bzgl. *be* keine weiteren Zustandsübergänge existieren. Die dritte Zeile gibt an, dass beim Lesen von *lehr* im Zustand S in den Zustand A2 übergegangen wird, beim Lesen von *lehr* im Zustand A1 in den Zustand A2 übergegangen wird und sonst keine Zustandsübergänge für *lehr* definiert sind usw.

	S	A1	A2	A3	A4
un	S	\emptyset	\emptyset	\emptyset	\emptyset
be	A1	\emptyset	\emptyset	\emptyset	\emptyset
lehr	A2	A2	\emptyset	\emptyset	\emptyset
bar	\emptyset	\emptyset	A3	\emptyset	\emptyset
keit	\emptyset	\emptyset	\emptyset	A4	\emptyset

Tabelle 2.7: Zustandstafel zum Automaten von Abbildung 2.3

Vom Automaten zur Grammatik

Man kann nun zeigen, dass die von einem DEA akzeptierte Sprache L regulär ist. Der Beweis dieser Aussage wird mittels der Konstruktion einer rechts-linearen Grammatik aus dem besagten Automaten geführt. Hierfür wird aus dem Automaten $A = \langle \Phi, \Sigma, \delta, S, F \rangle$ mit $L = L(A)$ eine rechts-lineare Grammatik $G = \langle \Phi, \Sigma, R, S \rangle$ mit $L = L(G)$ mit folgender Regelmenge konstruiert:

$$R = \{B \rightarrow xC \mid \delta(B, x) = C\} \cup \{B \rightarrow \varepsilon \mid B \in F\}$$

mit $B, C \in \Phi$ und $x \in \Sigma$. Für jeden Zustandsübergang $\delta(B, x)$ wird also in der Grammatik eine entsprechende Regel angegeben und für die Endzustände B wird eine Regel $B \rightarrow \varepsilon$ formuliert. Durch Induktion beweist man: $S \xrightarrow{*} wT$ gdw. $\delta^*(S, w) = T$ für $w \in \Sigma^*$ und $S, T \in \Phi$ bzw. $S \xrightarrow{*} w$ gdw. $\delta^*(S, w) \in F$.

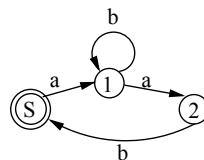
Beispiel 2.2.6

Man schaue sich hierfür noch einmal die Grammatik in Beispiel 2.2.5 und den Automaten in Abbildung 2.3 an, die die aus den Morphemen *un*, *be*, *lehr*, *bar* und *keit* gebildeten Derivationen generiert bzw. akzeptiert. Hier wurde genau dieser Beweis verwendet, um aus dem Automaten die angegebene rechts-lineare Grammatik zu erzeugen. \square

Ein weiteres Beispiel bildet der Automat bzw. die dazu gehörige Grammatik in 2.2.7, der die Sprache $L = \{(ab^*ab)^*\}$ akzeptiert bzw. generiert.

Beispiel 2.2.7

$G(L) = \langle \{S, 1, 2\}, \{a, b\}, R, S \rangle$ mit $R = \{S \rightarrow \varepsilon, S \rightarrow a1, 1 \rightarrow b1, 1 \rightarrow a2, 2 \rightarrow bS\}$



\square

Mit Hilfe der rechts-linearen Grammatiken kann also gezeigt werden, dass eine von einem deterministischen endlichen Automaten akzeptierte Sprache regulär ist. Mit Hilfe der rechts-linearen Grammatiken kann jedoch nicht gezeigt werden, dass jede reguläre Sprache von einem deterministischen endlichen Automaten akzeptiert wird! Um die Grundidee, die hinter dem obigen Beweis steht, übernehmen zu können, müsste zu jeder Regel einer rechts-linearen Grammatik ein Zustandsübergang eines entsprechenden DEAs konstruiert werden. Dies gelingt aber nicht in jedem Fall, denn Grammatiken sind nicht-deterministische Konzepte. So können z. B. die Regeln $A \rightarrow bA, A \rightarrow bB$ zum Regelinvantar einer Grammatik gehören. Um aus diesen Regeln Zustandsübergänge zu konstruieren, müsste ein DEA im Zustand A beim Lesen des Symbols b in den Zustand A oder

B übergehen. Dies bedeutet, der Automat müsste gleichzeitig sowohl $\delta(A, b) = A$ als auch $\delta(A, b) = B$ als Zustandsübergänge besitzen oder allgemeiner: Zustandsübergänge müssen als Relationen angegeben werden können. Genau dies lässt die Konzeption eines DEA nicht zu, denn dort wurden Zustandsübergänge als Funktionen definiert. Um den obigen Satz umzukehren, muss also zuerst gezeigt werden, dass NDEAs äquivalent zu DEAs sind. Erst dann kann mit Hilfe des Konzepts der rechts-linearen Grammatiken gezeigt werden, dass es zu jeder regulären Sprache L einen deterministischen endlichen Automaten gibt, der L akzeptiert.

Man kann tatsächlich zeigen, dass es sich bei NDEAs und DEAs um bzgl. der Ausdrucksstärke äquivalente Konzepte handelt. Die Idee hinter dem Beweis ist wie folgt: Für einen NDEA wird ein äquivalenter DEA konstruiert, indem die Potenzmenge der Zustandsmenge des NDEA als neue Zustandsmenge gewählt wird und entsprechende Übergänge eingeführt werden. Anschließend kann man anhand der fünf Teildefinitionen für reguläre Ausdrücke zeigen, dass für jeden dieser Teildefinitionen ein entsprechender Automat konstruiert werden kann. Aus Platzgründen wird der Beweis hier nicht gezeigt (siehe hierzu z. B. Hopcroft und Ullman 1979). Die Ergebnisse können jedoch zu dem Hauptsatz zusammengefasst werden, dass die Menge der regulären Sprachen gleich der Menge der von deterministischen sowie nicht-deterministischen endlichen Automaten akzeptierten Sprachen ist.

Transduktoren

Eine für die computerlinguistische Analyse wichtige Variante der endlichen Automaten sind die **Transduktoren** (engl. *Finite State Transducer*; FST). Ein FST ist eine Art endlicher Automat, der eine Zeichenkette als Ausgabe erzeugt, während eine Eingabe-Zeichenkette erkannt wird. FSTs können aber auch als endliche Automaten betrachtet werden, die gleichzeitig zwei Symbole bearbeiten und in entsprechende Zustände übergehen.¹ Der Unterschied zwischen Transduktoren und Automaten besteht also darin, dass Automaten eine Sprache über einem endlichen Alphabet von Einzelsymbolen akzeptieren, während Transduktoren Sprachen über Symbolpaaren $\langle x, y \rangle$ akzeptieren. Die Paare lassen sich übersichtlicher als $x:y$ angeben. Ein Beispiel wäre die Sprache $L = \{a:a\}^* \bullet a:\varepsilon \bullet \{b:b\}^*$ über dem Alphabet $\Sigma = \{a:\varepsilon, a:a, b:b\}$. Ein Transduktor, der L akzeptiert, ist in Abbildung 2.5 angegeben.

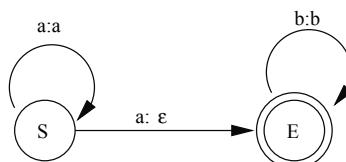


Abbildung 2.5: Transduktor, der $L = \{a:a\}^* \bullet a:\varepsilon \bullet \{b:b\}^*$ akzeptiert

¹Diese Beschränkung auf zwei Symbole ist aus theoretischer Sicht nicht notwendig. Ein FST kann prinzipiell n Symbole parallel bearbeiten.

Statt mit einem Alphabet von Symbolpaaren kann die Sprache auch auf der Basis zweier Alphabete formuliert werden. Der Transduktor in Abbildung 2.5 akzeptiert z. B. die folgenden zwei Zeichenketten-Paare: $\langle aaabbb, aabb \rangle, \langle abb, bb \rangle$. Transduktoren wurden bereits von Johnson (1972) für die Angabe phonologischer Regeln benutzt, aber erst Anfang der achtziger Jahre wurde ihr Einsatz für die Modellierung phonologischer und morphophonologischer Regeln in der maschinellen Analyse „wiederentdeckt“. Die Unterkapitel 3.1 zur Phonetik und Phonologie und 3.3 zur Morphologie stellen Methoden vor, die auf der Verwendung von Transduktoren aufbauen. FSTs stellen den einfachsten Ansatz zur Verarbeitung natürlicher Sprache dar, der in der Lage ist, sinnvolle Regularitäten zu modellieren. Sie sind zudem auch einfach algorithmisch umzusetzen.

Abschlusseigenschaften regulärer Sprachen

Wenn man weiß, dass ein Ausschnitt einer natürlichen Sprache durch eine einseitig-lineare Grammatik erzeugt werden kann – und somit regulär ist und durch einen endlichen Automaten erkannt wird – und dasselbe auch für einen anderen Ausschnitt derselben natürlichen Sprache gilt, dann ist es sehr nützlich zu wissen, wie sich diese beiden regulären Sprachen bzgl. mengentheoretischer Operationen wie der Vereinigung oder des Schnitts verhalten. Wenn das Ergebnis einer solchen Operation auf Elemente einer Menge wieder innerhalb dieser Menge ist, wird sie als **abgeschlossen** bzgl. dieser Operation bezeichnet. Wenn die Abschlusseigenschaft bzgl. einer Operation gilt, ist dies für die Entwicklung einer Grammatik bzw. eines Automaten für beide Sprachen vorteilhaft, denn man kann denselben Grammatik- bzw. Automatentypen weiterverwenden.

Die regulären Sprachen sind abgeschlossen unter Verkettung ($L_1 \bullet L_2$), Vereinigung ($L_1 \cup L_2$), Schnitt ($L_1 \cap L_2$), Komplement ($\Sigma^* \setminus L$) und Sternbildung (L^*).

Dass die Operationen Vereinigung, Verkettung und Sternbildung abgeschlossen sind, folgt bereits aus der Definition regulärer Ausdrücke. Für die Komplementbildung muss gezeigt werden, dass $\Sigma^* \setminus L$ eine reguläre Menge ist, wenn $L \subseteq \Sigma^*$ eine reguläre Menge ist. Wenn L von dem deterministischen endlichen Automaten $A = \langle \Phi, \Sigma, \delta, S, F \rangle$ erkannt wird, wird $\Sigma^* \setminus L$ von $A' = \langle \Phi, \Sigma, \delta, S, \Phi \setminus F \rangle$ akzeptiert. Der Automat A' enthält also als Endzustände das Komplement der Endzustände von A . Der Automat A' akzeptiert genau $\Sigma^* \setminus L$, denn für jedes Wort aus L erreicht er einen Zustand aus F , die bei A' jedoch keine Endzustände sind. Für Wörter, die nicht in L sind, erreicht A' einen Zustand, der nicht in F ist, und dies sind genau die Endzustände von A' . Der Durchschnitt wird nach dem Gesetz von DeMorgan (siehe hierzu das Unterkapitel 2.1) auf die Mengenvereinigung und Komplementbildung zurückgeführt, deren Abgeschlossenheit schon dargestellt wurde.

2.2.4 Kontextfreie Sprachen und Grammatiken

Einseitig-lineare Grammatiken erlauben auf der linken Seite einer Regel nur ein nichtterminales Symbol und auf der rechten Seite entweder nur ein Wort aus Σ^* oder ein nichtterminales Symbol links bzw. rechts eines Wortes aus Σ^* . Für

kontextfreie Sprachen (auch **Typ-2-Sprachen** genannt) wird die Restriktion auf der rechten Seite der Regeln aufgehoben: Ein nichtterminales Symbol geht in ein Wort aus dem Gesamtalphabet über. Das heißt, auf der rechten Seite einer Regel kann eine beliebige Folge von Terminal- und Nichtterminalsymbolen stehen.

Definition 2.2.8

Eine Grammatik $G = \langle \Phi, \Sigma, R, S \rangle$ heißt kontextfrei (oder auch Typ-2), falls alle Regeln von der Form

$$A \rightarrow \alpha$$

mit $A \in \Phi$ und $\alpha \in \Gamma^* = (\Phi \cup \Sigma)^*$ sind. \square

Die Form der Regeln kontextfreier Grammatiken zeigt, dass einseitig-lineare Grammatiken auch kontextfrei sind. Dies legt natürlich die Vermutung nahe, dass die regulären Sprachen eine echte Teilmenge der kontextfreien Sprachen sind. Die höhere Ausdrucksmächtigkeit kontextfreier Sprachen gegenüber regulären Sprachen zeigt sich insbesondere durch die Möglichkeit, Klammerstrukturen zu beschreiben. So wird die Sprache $L = \{a^i b a^i \mid i \geq 0\}$ von der kontextfreien Grammatik $G = \langle \{S\}, \{a, b\}, \{S \rightarrow b, S \rightarrow aSa\}, S \rangle$ erzeugt. Eine einseitig-lineare Grammatik lässt sich für diese Sprache nicht formulieren.

Kontextfreie Sprachen lassen zwar Klammerstrukturen zu, nicht aber beliebig viele Verkettungen verschiedener Zeichen mit gleicher Anzahl. So ist insbesondere die Sprache $L = \{a^i b^i a^i \mid i > 0\}$ nicht kontextfrei.

Für kontextfreie Grammatiken G gilt, dass für ein Wort $w \in L(G)$ in der Regel mehrere Ableitungen $S \xrightarrow{*} w$ existieren, da nichtterminale Symbole nicht in einer strikten Reihenfolge verwendet werden müssen. Ableitungen, bei denen immer das am weitesten links stehende Nichtterminalsymbol ersetzt wird, heißen **Linksableitungen**. Entsprechend heißen Ableitungen, bei denen immer das am weitesten rechts stehende Nichtterminalsymbol ersetzt wird, **Rechtsableitungen**.

Die Regelanwendungen der Grammatik aus Beispiel 2.2.1 sind Links- bzw. Rechtsableitungen, die zur Erzeugung des Worts *Heinz inszeniert seinen Auftritt* führen. Darauf hinaus bestehen noch weitere Möglichkeiten, dieses Wort zu erzeugen. Im Gegensatz zu den kontextfreien Grammatiken besitzen einseitig-lineare Grammatiken eindeutige Ableitungen, da pro Regel auf der linken Seite der Regel immer nur ein nichtterminales Symbol auftaucht.

Kellerautomaten

Die Erkennungsmechanismen für kontextfreie Sprachen sind die **Kellerautomaten**. Der Engpass der endlichen Automaten war bisher deren beschränktes Gedächtnis: Nur die Zustände speichern Information und zwar auch nur von einem Erkennungsschritt zum nächsten. Für kontextfreie Sprachen benötigt man jedoch ein prinzipiell unbeschränktes Gedächtnis. Für die prototypische kontextfreie Sprache $\{a^i b a^i \mid i \in \mathbb{N}_0\}$ kann ja i einen beliebigen Wert aus \mathbb{N}_0 annehmen. Die Information über die Größe von i muss über das b hinweg beim Lesen des

Wortes von links nach rechts behalten werden können. Dies kann durch eine recht einfache Operation erreicht werden. Es muss nämlich jeweils die letztgenannte Größe aus dem Gedächtnis gelesen werden können. Genau dies leisten Kellerautomaten. Der Keller ist bei diesem Automaten als ein Speicherband vorstellbar, auf dem nur das zuletzt gelesene Zeichen gelesen oder gelöscht werden kann, und auch nur bei diesem letzten Zeichen kann neue Information auf das Band geschrieben werden. Diese Operationen auf dem Speicherband muss die Definition der Übergangsfunktion leisten.

Im Gegensatz zu den endlichen Automaten sind deterministische und nichtdeterministische Kellerautomaten nicht äquivalent. Nur die **nicht-deterministischen Kellerautomaten** akzeptieren genau die kontextfreien Sprachen. **Deterministische Kellerautomaten** akzeptieren eine kleinere Sprachklasse, die sog. **deterministischen kontextfreien Sprachen**, die echt zwischen den regulären und den kontextfreien Sprachen liegt: $\mathcal{L}_3 \subset \mathcal{L}_{det.\text{kontextfrei}} \subset \mathcal{L}_2$. Die Grammatiken von Programmiersprachen sind häufig deterministisch-kontextfrei.

Definition 2.2.9

Ein **nicht-deterministischer Kellerautomat** $K = \langle \Phi, \Sigma, \Delta, \diamond, \delta, S, F \rangle$ besteht aus

1. einem Alphabet Φ von Zuständen
2. einem Alphabet Σ von Eingabesymbolen ($\Phi \cap \Sigma = \emptyset$)
3. einem Alphabet Δ von Kellersymbolen ($\Phi \cap \Delta = \emptyset$)
4. einem Kelleranfangssymbol $\diamond \notin \Phi \cup \Delta$
5. einer Übergangsfunktion $\delta : \Phi \times \Sigma \times (\Delta \cup \{\diamond\}) \rightarrow \wp(\Phi \times \Delta^*)$
6. einem Startzustand $S \in \Phi$
7. einer Menge $F \subset \Phi$ von Endzuständen

□

Die Übergangsfunktion soll abhängig vom Zustand des Kellerautomaten, dem gelesenen Zeichen und einem Zeichen auf dem Speicherband, i.e. dem Keller, (neue) Zustände bestimmen und Wörter auf das Speicherband schreiben. Hierbei wird immer das am weitesten rechts stehende Zeichen auf dem Speicherband durch ein Wort ersetzt. Da die Übergangsfunktion für einen nicht-deterministischen Kellerautomaten definiert wurde, und somit ihr Wertebereich die Potenzmenge von $\Phi \times \Delta^*$ wäre, die Potenzmenge abzählbar unendlicher Mengen jedoch ebenfalls unendlich ist, muss sie auf endliche Potenzmengen eingeschränkt werden, weil sonst der Automat nicht mehr beschrieben werden könnte. Die Übergangsfunktion sei als Tripel $\langle T, x, p \rangle$ angegeben. Es gibt den gegenwärtigen Zustand T des Automaten an, das zu lesende Zeichen x sowie das letzte Zeichen p auf dem Kellerspeicher. Nachfolgezustände geben dann entsprechend den evtl. neuen Zustand sowie anstelle des alten Zeichens das neue Wort auf dem Kellerspeicher

an; $\delta(\langle T, x, p \rangle)$ ist der Nachfolgezustand. Im Anfangszustand, wenn noch kein Zeichen gelesen wurde, steht in dem Kellerspeicher nur das Kelleranfangssymbol \diamondsuit .

Ein Kellerautomat, der die kontextfreie Sprache $L = \{a^i b a^i \mid i \in \mathbb{N}_0\}$ akzeptiert, ist $K = \langle \{A, B, C\}, \{a, b\}, \{\ast, a\}, \diamondsuit, \delta, A, \{C\} \rangle$ mit der in Tabelle 2.8 angegebenen Übergangsfunktion δ .

Fall	$\langle T, x, p \rangle$	$\delta(\langle T, x, p \rangle)$	Kommentar
1	$\langle A, a, \diamondsuit \rangle$	$\{\langle A, \ast \rangle\}$	erstes a wird gelesen und dafür \ast gespeichert
2	$\langle A, a, \ast \rangle$	$\{\langle A, \ast a \rangle\}$	zweites a wird durch $a\ast$ ersetzt
3	$\langle A, a, a \rangle$	$\{\langle A, aa \rangle\}$	weitere a 's werden konkateniert
4	$\langle A, b, \diamondsuit \rangle$	$\{\langle C, \varepsilon \rangle\}$	für den Fall $i = 0$
5	$\langle A, b, \ast \rangle$	$\{\langle B, \ast \rangle\}$	für den Fall $i = 1$
6	$\langle A, b, a \rangle$	$\{\langle B, a \rangle\}$	für den Fall $i > 1$
7	$\langle B, a, \ast \rangle$	$\{\langle C, \varepsilon \rangle\}$	falls der Automat beim Abbau auf das erste a stößt
8	$\langle B, a, a \rangle$	$\{\langle B, \varepsilon \rangle\}$	falls der Automat beim Abbau auf ein gespeichertes a trifft

Tabelle 2.8: Übergangsfunktion für Kellerautomaten, der $\{a^i b a^i\}$ akzeptiert

Dieser Automat ist zwar deterministisch, aber dies liegt an den Eigenschaften der vom Automaten akzeptierten Sprache. Es gibt kontextfreie Sprachen, die einen nicht-deterministischen Kellerautomaten erfordern. Der Kellerautomat liest im Startzustand A entweder nur ein b und geht direkt in den Endzustand C über (Fall 4), oder er liest ein a . In diesem Fall schreibt er das Symbol \ast auf den Keller und bleibt im Zustand A (Fall 1). Falls jetzt ein b gelesen wird, geht der Automat in den Zustand B und lässt das Symbol \ast stehen (Fall 5). Falls der Automat jedoch ein weiteres a liest, bleibt er im Zustand A und verkettet das Symbol \ast mit a zum Wort $\ast a$ (Fall 2). Wenn weitere Vorkommen des Symbols a gelesen werden, wird im Zustand A für jedes gelesene a im Kellerspeicher ein a an das letzte im Keller gespeicherte a angehängt (Fall 3). Liest der Automat im Zustand A ein b und hat er ein a als letztes Zeichen auf dem Keller, geht er in den Zustand B über und lässt das a stehen (Fall 6). Wenn der Automat im Zustand B ein a liest und das letzte Zeichen des Kellerworts ein a ist, bleibt er im Zustand B und löscht dieses a ; d.h. er ersetzt es durch das leere Wort (Fall 8). Wenn der Automat im Zustand B als letztes Zeichen auf dem Kellerspeicher das \ast besitzt und ein a liest, löscht er das \ast und geht in den Endzustand C über (Fall 7).

Um nachzuvollziehen, wie der Kellerautomat die Sprache L akzeptiert, muss beschrieben werden, wie sich bei jedem Zeichen aus dem Eingabewort w der Zustand des Automaten und der Speicherinhalt ändern. Für den obigen Automaten wurde dies nur informell getan. Eine formale Beschreibung dieser Änderungen wird durch die Angabe der jeweiligen **Konfiguration** und der **Konfigurationsübergänge** geleistet. Konfigurationen geben sämtliche Informationen über den jeweiligen Kellerautomaten zu einem bestimmten Zeitpunkt an.

Definition 2.2.10

Sei K ein Kellerautomat. Eine Konfiguration $\langle T, v, p \rangle$ von K besteht aus:

1. einem Zeichen $T \in \Phi$, dem Zustand des Automaten K
2. einem Wort $v \in \Sigma^*$, dem noch nicht gelesenen Teil des Eingabeworts
3. einem Wort $p \in \Delta^* \cup \{\diamond\}$, dem Kellerinhalt

□

Die Startkonfiguration vor der Erkennung eines Wortes $w \in \Sigma^*$ ist durch $\langle S, w, \diamond \rangle$ gegeben. Eine Endkonfiguration ist durch $\langle T, \varepsilon, z \rangle$ gegeben mit $T \in F$ und $z \in \Delta^* \cup \{\diamond\}$. Das Wort w ist akzeptiert, wenn von der Startkonfiguration aus nach seiner Bearbeitung eine Endkonfiguration erreicht ist.

Der direkte Übergang von einer Konfiguration zur nächsten ist wie folgt definiert:

Definition 2.2.11

Eine Konfiguration $s = \langle T, v, p \rangle$ kann **direkt** in eine Konfiguration $t = \langle T', v', p' \rangle$ übergehen (in Zeichen: $\langle T, v, p \rangle \Rightarrow \langle T', v', p' \rangle$) falls gilt:

- $v = av'$ mit $a \in \Sigma \cup \{\varepsilon\}$ und $v' \in \Sigma^*$
- $p = ab$ mit $a \in \Delta^*$ und $b \in \Delta$
- $p' = a\gamma$ mit $\gamma \in \Delta^*$
- $\langle T', \gamma \rangle \in \delta(\langle T, a, b \rangle)$

Eine Konfiguration $s = \langle T, v, p \rangle$ geht in eine Konfiguration $t = \langle T', v', p' \rangle$ über (in Zeichen: $\langle T, v, p \rangle \xrightarrow{*} \langle T', v', p' \rangle$), wenn $\langle T, v, p \rangle = \langle T', v', p' \rangle$ gilt oder wenn es für ein $n \geq 1$ Konfigurationen $s = s_0, s_1, \dots, s_n = t$ gibt mit $s_{i-1} \Rightarrow s_i$ für $i = 1, \dots, n$. □

Beim Übergang wird ein (evtl. neuer) Zustand eingenommen, der noch nicht gelesene Teil v des Eingabeworts wird um das am weitesten links stehende Zeichen a verkürzt und beim Kellerinhalt p wird das letzte Zeichen b durch ein Wort γ aus Δ^* ersetzt. Die von einem Kellerautomaten K akzeptierte Sprache ist

$$L(K) = \{w \in \Sigma^* \mid \langle S, w, \diamond \rangle \xrightarrow{*} \langle T, \varepsilon, z \rangle \text{ mit } T \in F, z \in \Delta^* \cup \{\diamond\}\}$$

Die Konfigurationsübergänge für die Worte b und $aabaa$ der Sprache $L = \{a^i b a^i \mid i \in \mathbb{N}_0\}$ sind mit dem oben beschriebenen Kellerautomaten:

$$\begin{aligned} b : \quad & \langle A, b, \diamond \rangle \Rightarrow \langle C, \varepsilon, \varepsilon \rangle \\ aabaa : \quad & \langle A, aabaa, \diamond \rangle \Rightarrow \langle A, abaa, * \rangle \\ & \Rightarrow \langle A, baa, *a \rangle \\ & \Rightarrow \langle B, aa, *a \rangle \\ & \Rightarrow \langle B, a, * \rangle \\ & \Rightarrow \langle C, \varepsilon, \varepsilon \rangle \end{aligned}$$

Für den Beweis, dass die Menge der kontextfreien Sprachen gleich der Menge der von nicht-deterministischen Kellerautomaten akzeptierten Sprachen über einem Alphabet Σ ist, sei auf Hopcroft und Ullman (1994, 121ff.) verwiesen.

Abschlusseigenschaften kontextfreier Sprachen

Im Gegensatz zu den regulären Sprachen sind kontextfreie Sprachen nicht unter allen genannten mengentheoretischen Operationen abgeschlossen. Während sie unter der Vereinigung, der Verkettung und der Sternbildung abgeschlossen sind, sind sie unter Durchschnitt, Komplement und Differenz nicht abgeschlossen. Man kann mittels der beiden kontextfreien Sprachen $L_1 = \{a^i b^i a^* \mid i \geq 1\}$ und $L_2 = \{a^* b^i a^i \mid i \geq 1\}$ zeigen, dass der Schnitt $L_1 \cap L_2 = \{a^i b^i a^i \mid i \geq 1\}$ eine nicht-kontextfreie Sprache ist. Aufgrund der Nichtabgeschlossenheit unter Schnittbildung sind kontextfreie Sprachen auch unter der Komplementbildung und der Differenz nicht abgeschlossen. Allerdings ist der Schnitt einer kontextfreien Sprache mit einer regulären Sprache immer eine kontextfreie Sprache.

Deterministisch-kontextfreie Sprachen sind jedoch unter Komplementbildung abgeschlossen. Sie sind aber nicht unter Schmitt und Vereinigung abgeschlossen.

2.2.5 Nicht-kontextfreie Sprachen und Grammatiken

Abschließend sollen verschiedene Sprachen vorgestellt werden, die mächtiger sind als die kontextfreien Sprachen. Dies sind zum einen die **kontextsensitiven Sprachen** (auch **Typ-1-Sprachen** genannt) und zum anderen die noch mächtigere Klasse der **allgemeinen Regelsprachen**, die auch **Typ-0-Sprachen** genannt werden. Zwischen den kontextfreien und den kontextsensitiven Sprachen liegt die Klasse der **schwach kontextsensitiven Sprachen**, die insbesondere aus computerlinguistischer Sicht interessant ist, da sie oft hinreichend ist für die Darstellung nicht-kontextfreier Strukturen natürlicher Sprachen. Von diesen schwach kontextsensitiven Sprachen werden in diesem Abschnitt die durch **Baumadjunktions-Grammatiken** erzeugten Sprachen vorgestellt.

Kontextsensitive Sprachen und Grammatiken

Die **kontextsensitiven Sprachen** liegen echt zwischen den kontextfreien und den allgemeinen Regelsprachen. Sie werden von **kontextsensitiven Grammatiken** erzeugt (auch Typ-1-Grammatik genannt).

Definition 2.2.12

Eine Grammatik $G = \langle \Phi, \Sigma, R, S \rangle$ heißt kontextsensitiv, falls alle Regeln die Gestalt $\alpha A \gamma \rightarrow \alpha \beta \gamma$ mit $\alpha, \beta, \gamma \in \Gamma^*, A \in \Phi, \beta \neq \varepsilon$ oder die Gestalt $S \rightarrow \varepsilon$ haben. Falls die Regel $S \rightarrow \varepsilon$ Element von R ist, darf S nicht auf der rechten Seite einer Regel vorkommen. \square

Ein Nichtterminalsymbol A darf in kontextsensitiven Grammatiken nur dann durch ein Wort β aus dem Gesamtalphabet ersetzt werden, wenn es im Kontext $\alpha - \gamma$ auftritt. Das prototypische Beispiel einer kontextsensitiven Sprache ist $L = \{a^i b^i a^i \mid i \geq 0\}$.

Da das leere Wort in kontextsensitiven Grammatiken nur aus dem Startsymbol abgeleitet werden darf, steigt in den Ableitungen die Anzahl der Symbole an. Kontextsensitive Grammatiken sind daher **längenmonoton**. Man kann sogar umgekehrt zeigen, dass jede längenmonotone Sprache kontextsensitiv ist.

Definition 2.2.13

Eine Grammatik heißt längenmonoton, falls alle Regeln die Gestalt

$$\alpha \rightarrow \beta \text{ mit } \alpha, \beta \in \Gamma^* \text{ und } \alpha \neq \varepsilon \text{ und } |\alpha| \leq |\beta|$$

oder

$$S \rightarrow \varepsilon$$

haben. Im zweiten Fall darf S nicht auf einer rechten Seite vorkommen. \square

Für die Erkennung kontextsensitiver Sprachen wird der Begriff der **Turingmaschine** benötigt. Da Turingmaschinen aber äquivalent zu den allgemeinen Regelsprachen sind, werden sie erst im übernächsten Abschnitt vorgestellt. Vorab sei aber bereits erwähnt, dass kontextsensitive Sprachen von speziellen Turingmaschinen akzeptiert werden, nämlich denjenigen Turingmaschinen, bei denen die Begrenzungssymbole nicht verschoben werden müssen. Man spricht daher auch statt von Turingmaschinen von **linear beschränkten Automaten**, die die kontextsensitiven Sprachen akzeptieren. Kontextsensitive Sprachen sind unter Komplementbildung, Schnitt und Vereinigung abgeschlossen.

Baumadjunktions-Grammatiken

Kontextfreie Grammatiken können große Bereiche natürlicher Sprachen abdecken. Allerdings ist in mehreren Arbeiten gezeigt worden, dass natürliche Sprachen auch strukturelle Eigenschaften besitzen, die über Kontextfreiheit hinausgehen. Zum Schweizerdeutschen hat dies Schieber (1985) gezeigt. Diese Eigenschaften können mittels Grammatiken für **schwach kontextsensitive Sprachen**

beschrieben werden, die echt zwischen den kontextfreien und den kontextsensitiven Sprachen liegen. Zu diesen schwach kontextsensitiven Sprachen gehören insbesondere die mittels **Baumadjunktions-Grammatiken (Tree Adjoining Grammars, TAGs)** erzeugten Sprachen (cf. Joshi 1985).

TAGs erzeugen Sätze nicht wie bei den bisherigen Grammatiken durch die Anwendung von Ersetzungsregeln für Zeichen, sondern durch Regeln für die Konstruktion von Bäumen. Ein TAG $G = \langle I, A \rangle$ besteht aus einer endlichen Menge I initialer Bäume und einer endlichen Menge A auxiliarer Bäume und verwendet eine **Adjunktionsoperation** für die Konstruktion komplexerer Bäume. Die Bäume aus I und A werden **elementare Bäume** genannt.

Ein Baum $\alpha \in I$ besitzt als Wurzelknoten das Startsymbol. Die Endknoten von α sind Terminalsymbole. Ein Baum $\beta \in A$ besitzt als Wurzelknoten irgend ein Nichtterminalsymbol $N \neq S$. Die Endknoten sind bis auf einen Knoten, der wiederum mit dem Symbol N versehen ist, Terminalsymbole. Dieser mit N bezeichnete Knoten wird Fußknoten von β genannt und erlaubt die Beschreibung rekursiver Strukturen.

Die **Adjunktion** eines auxiliaren Baumes β mit Wurzelknoten N mit einem Baum γ , der einen mit N bezeichneten Knoten besitzt, resultiert in einem Baum γ' , bei dem der Teilbaum t von γ , der von N dominiert wird, aus γ entfernt wird. Der Baum β wird mit dem Knoten N verbunden, t wird am Fußknoten von β angehängt und der Wurzelknoten von t wird mit dem Fußknoten von β gleichgesetzt. Abbildung 2.6 zeigt diese Operation graphisch.

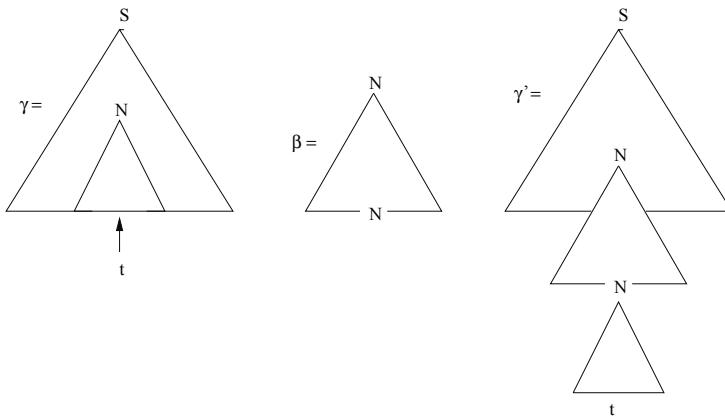
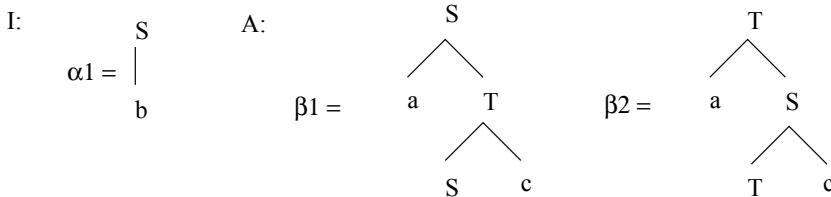


Abbildung 2.6: Beispiel einer Baumadjunktion

Die Menge der von G erzeugten Bäume ist die Menge der Bäume, die von jedem Baum aus I durch die wiederholte Anwendung der Adjunktion von Bäumen aus A aufgebaut werden.

Da TAGs mächtiger sind als kontextfreie Sprachen, können sie unter anderem auch kontextfreien Sprachausschnitten Strukturen zuweisen, die mittels kontextfreier Grammatiken nicht realisierbar sind. Dies macht sie für computerlinguistische Anwendungen ebenfalls sehr attraktiv. So lassen sich überkreuzen-

de Abhängigkeiten, die in verschiedenen natürlichen Sprachen auftreten, mittels TAGs beschreiben. Solche Abhängigkeiten zeigen sich in Strukturen der Form $a^n b^m x c^n d^m$. Wenn solche Abhängigkeiten in der kontextfreien Sprache $\{a^n b c^n \mid n \geq 1\}$ auftauchen, können sie nach Joshi (1990) mittels der in Abbildung 2.7 angegebenen TAG, die diese Sprache generiert, beschrieben werden.



$$\text{adjunktion}(\alpha_1, \beta_1) =$$

$$\text{adjunktion}((\alpha_1, \beta_1), \beta_2) =$$

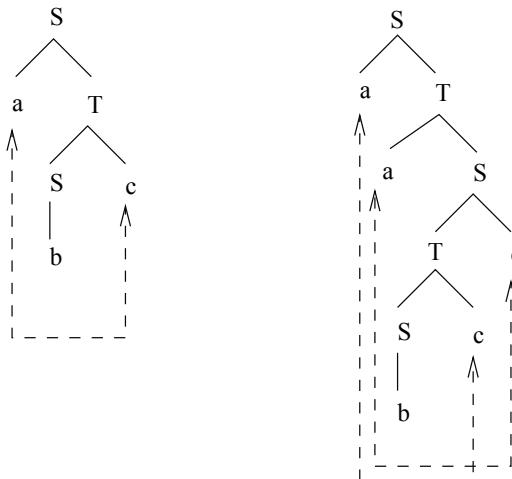


Abbildung 2.7: Baumadjunktion für die Darstellung sich überkreuzender Abhängigkeiten

Mittels TAGs erzeugte Sprachen sind unter Vereinigung, Konkatenation, Sternbildung sowie Schnitt mit regulären Sprachen abgeschlossen, nicht aber unter Schnittbildung mit anderen durch TAGs erzeugte Sprachen und unter Komplementbildung sowie Differenz.

Allgemeine Regelsprachen

Damit können wir zu den allgemeinen Regelsprachen bzw. Typ-0-Sprachen als weiterem klassischen Sprachtyp übergehen. Diese Sprachen werden von allgemeinen Regelgrammatiken erzeugt (auch Typ-0-Grammatiken genannt).

Allgemeine Regelgrammatiken wurden bereits in Abschnitt 2.2.2 bei der Einführung des Grammatikbegriffs vorgestellt. Diese Grammatiken müssen die Bedingung erfüllen, dass auf der linken Seite der Regeln mindestens ein nichtterminales Symbol steht. Ansonsten können sowohl auf der linken als auch auf der rechten Seite der Regeln beliebige Wörter aus dem Gesamtalphabet Γ stehen. Kontextsensitive Grammatiken sind eingeschränktere Formen der allgemeinen Regelgrammatiken, denn sie müssen die Bedingung der Längenmonotonie erfüllen.

Beispiel 2.2.8

Ein Beispiel für eine allgemeine Regelgrammatik ist die Grammatik $G = \langle \{S, A, B, C, D, E\}, \{a\}, R, S \rangle$ mit der folgenden Regelmenge R , die die Sprache $L = \{a^{2^i} \mid i \geq 1\}$ generiert (vgl. Hopcroft und Ullman 1994, S. 240).

$$\begin{array}{lll} S \rightarrow ACaB & Ca \rightarrow aaC & CB \rightarrow DB \\ CB \rightarrow E & aD \rightarrow Da & AD \rightarrow AC \\ aE \rightarrow Ea & AE \rightarrow \epsilon & \end{array}$$

□

Das Wort $a^{2^2} = aaaa$ wird wie folgt abgeleitet:

$$\begin{array}{llll} S & \rightarrow ACaB & \rightarrow AaaCB & \rightarrow AaaDB \\ & \rightarrow ADaaB & \rightarrow ACaaB & \rightarrow AaaaaCB \\ & \rightarrow AaaaaE & \rightarrow AaaaEa & \rightarrow AaaEaa \\ & \rightarrow AEaaaa & \rightarrow aaaa & \end{array}$$

Allgemeine Regelgrammatiken und kontextsensitive Grammatiken sind viel zu komplex, als dass sie für computerlinguistische Anwendungen interessant sein könnten. Der Nutzen der von allgemeinen Regelgrammatiken erzeugten Sprachen liegt vielmehr in einem anderen Bereich, nämlich in ihrer Äquivalenz zu den so genannten **rekursiv aufzählbaren Sprachen**, und hängt mit ihrer Akzeptanz durch Turingmaschinen zusammen. Turingmaschinen sind nämlich nicht nur Akzeptoren für allgemeine Regelsprachen, sie stellen auch gleichzeitig ein allgemeines Rechnermodell für Aussagen zur Verarbeitungskomplexität und Ressourcenbelegung dar.

Turingmaschinen

Die Turingmaschine hatte ursprünglich einen anderen Zweck als die endlichen Automaten oder die Kellerautomaten. Alan Turing wollte mit der nach ihm benannten abstrakten Maschine den Begriff der Berechnung explizieren. Turings Intention war es, ein System zu entwickeln, mit dem jeder Prozess modelliert werden kann, der im üblichen Sinn als Berechnung betrachtet werden kann (cf. Turing 1936), d.h. Turingmaschinen explizieren den Begriff der Berechenbarkeit.

Turingmaschinen erkennen allgemeine Regelsprachen bzw. rekursiv aufzählbare Sprachen. Kellerautomaten besitzen zwar einen unendlich großen Speicher, aber es sind nur recht eingeschränkte Operationen zugelassen. Diese Einschränkung muss für allgemeine Regelsprachen fallengelassen werden. Zwar dient die

Turingmaschine als Akzeptor einer Sprache, aber nach Abschluss der Erkennung eines Wortes steht auf dem Speicherband in der Regel noch ein Wort, das als Ausgabe interpretiert werden kann. In diesem Sinn handelt es sich bei der Turingmaschine um eine Maschine und nicht um einen Automaten.

Ein anschauliches Modell einer Turingmaschine besteht aus einer Kontrolle der Zustände und Übergänge, einem Eingabeband, das in einzelne Felder unterteilt und sowohl links als auch rechts prinzipiell unendlich ist, sowie einem Lese-/Schreibkopf, der zu jedem Zeitpunkt ein Feld des Bandes bearbeiten kann. Jedes Feld des Eingabebandes enthält genau ein Symbol. Das Eingabewort wird links und rechts durch das Begrenzungssymbol \diamond begrenzt. Bei der Bearbeitung des Wortes kann das Begrenzungssymbol nach links oder rechts verschoben werden. Die Turingmaschine arbeitet wie folgt auf dem Band: Abhängig von dem Zustand der Maschine und dem gelesenen Symbol ändert sie ihren Zustand, schreibt ein Symbol in das gelesene Bandfeld und bewegt den Lese-/Schreibkopf ein Zeichen nach links (-), nach rechts (+) oder gar nicht (\circ). Die formale Definition lautet:

Definition 2.2.14

Eine (nicht-deterministische) Turingmaschine $T = \langle \Phi, \Sigma, \diamond, \delta, S, F \rangle$ besteht aus:

1. Einem Alphabet Φ von Zuständen
2. Einem Alphabet Σ von Band- oder Eingabesymbolen mit $\Sigma \cap \Phi = \emptyset$
3. Einem Begrenzungssymbol \diamond
4. Einer Übergangsfunktion $\delta : \Phi \times (\Sigma \cup \{\diamond\}) \rightarrow \wp(\Phi \times (\Sigma \cup \{\diamond\})) \times \{-, +, \circ\}$
5. Einem Startzustand $S \in \Phi$
6. Einer Menge $F \subset \Phi$ von Endzuständen

□

Neben der obigen Definition einer Turingmaschine existieren noch viele andere Definitionen. So kann das Eingabeband auf einer Seite begrenzt sein, und das Eingabewort der Länge n steht auf den ersten n linken Feldern. Eine andere Möglichkeit besteht darin, mehrere Bänder zuzulassen, so dass z. B. ein Band nur ein Eingabeband darstellt, während auf die anderen Bänder geschrieben werden darf. Dies sind aber alles nur definitorische Varianten, denn an der Ausdrucksstärke der Turingmaschinen ändern sie nichts. Diese Varianten sind äquivalent zu der angegebenen Turingmaschine, und alle sind äquivalent zu den allgemeinen Regelsprachen. Insbesondere sind deterministische und nicht-deterministische Turingmaschinen – im Gegensatz zu den Kellerautomaten – wieder äquivalente Konzepte. Daher gilt der Satz, dass die Menge der allgemeinen Regelsprachen über einem Alphabet Σ gleich der Menge der von deterministischen und nicht-deterministischen Turingmaschinen akzeptierten Sprachen über Σ ist.

2.2.6 Komplexitäts- und Entscheidbarkeitseigenschaften

Die Komplexität wird über die Ressourcen ermittelt, die bei der Berechnung einer Lösung für eine Probleminstanz verwendet werden. Sie betrifft also den Speicher- und Zeitbedarf. Die Zeit, die für die Durchführung einer Berechnung erforderlich ist, wird als **Zeitkomplexität** bezeichnet. Entsprechend ist die **Speicherkomplexität** durch den Speicher gegeben, der für die Berechnung belegt werden muss. Um Aussagen zu erhalten, die unabhängig sind von irgendwelchen konkreten Rechnern mit ihren spezifischen hardwarebedingten Ressourcenbeschränkungen, wird als allgemeines Berechnungsmodell für die Komplexität die Turingmaschine verwendet.

Ein Übergang in einer Turingmaschine entspricht dann einem Schritt bei der Berechnung. Die Zeitkomplexität einer Turingmaschine wird als die Anzahl der Schritte definiert, die die Turingmaschine durchführen muss, um zu einer Lösung zu gelangen. Die Speicherkomplexität wird über die Anzahl der Zellen auf dem Speicherband definiert, die für die Berechnung erforderlich sind.

Zeit- und Speicherkomplexität sind nicht unabhängig voneinander, denn bei n Schritten hat die Turingmaschine höchstens Zugriff auf $n + 1$ Zellen, d.h. wenn die Zeitkomplexität einer Turingmaschinen-Berechnung n ist, dann ist die Speicherkomplexität höchstens $n + 1$. Im Folgenden wird allein die Zeitkomplexität genauer betrachtet.

Die Größenordnung der Komplexität wird typischerweise in der \mathcal{O} -Notation ausgedrückt, die ein wichtiges Konzept in der Komplexitätstheorie darstellt und das asymptotische Verhalten von Funktionen beschreibt. Zeit- und Speicherkomplexität werden hierfür als Funktionen der Größe der Eingabe n betrachtet:

Definition 2.2.15

Seien $f, g : N \rightarrow N$.

$$\mathcal{O}(g) = \{\exists k > 0, n_0 > 0 : f(n) \leq k \cdot g(n) \text{ für alle } n > n_0\} \quad \square$$

Die Funktion f wächst nicht schneller als die Funktion g . Ein Beispiel: Angenommen, der Zeitbedarf für das Lösen eines Problems der Größe n ist $T(n) = 4 \cdot n^2 - 2n + 3$. Dann ist $T(n) = \mathcal{O}(n^2)$, denn die Konstanten dürfen ignoriert werden sowie Teilausdrücke, die abhängig von n nicht schnell wachsen.

Die Analyse des Rechenaufwands für eine Problemlösung soll nicht von der jeweiligen Probleminstanz abhängen. So sollte z. B. die Komplexität der Multiplikation nicht davon abhängen, ob $1 * 2$ berechnet wird oder $652109 * 498740$. Daher wird für Komplexitätsanalysen in der Regel die Situation des worst-case Falls betrachtet, also die zeitlich längste Berechnung. Die Komplexität von Algorithmen lässt sich nun mit n als der Länge der Eingabe wie folgt klassifizieren:

konstant:	Der Arbeitsaufwand ist unabhängig von n	$\mathcal{O}(1)$
logarithmisch:	$\log_k(n)$ (für eine Konstante k)	$\mathcal{O}(\log(n))$
polynomial:	n^k (für eine Konstante k) $k = 1$ heißt linear ($\mathcal{O}(n)$), $k = 2$ quadratisch ($\mathcal{O}(n^2)$) und $k = 3$ kubisch ($\mathcal{O}(n^3)$)	$\mathcal{O}(n^k)$
exponentiell:	k^n (für eine Konstante k)	$\mathcal{O}(k^n)$

Von besonderer Bedeutung für die Komplexitätstheorie und die Entscheidbarkeitseigenschaften formaler Sprachen ist die Klasse P, die von Turingmaschinen in polynomialem Zeit akzeptiert werden kann. Die Menge P ist deswegen interessant, weil sie die Sprachen enthält, die in einer vertretbaren Zeit akzeptiert werden. Wenn man annehmen würde, dass ein Rechenschritt eine Mikrosekunde an Zeit benötigt, zeigt sich für $k = 2$ nach Brookshear (1989, 260) folgende polynomiale und exponentielle Zeitkomplexität:

n	n^2	2^n
10	.0001 Sekunden	.0001 Sekunden
20	.0004 Sekunden	1.05 Sekunden
30	.0009 Sekunden	17.92 Minuten
40	.0016 Sekunden	12.74 Tage
50	.0025 Sekunden	35.75 Jahre

Neben Komplexität spielen **Entscheidbarkeitseigenschaften** formaler Sprachen eine wichtige Rolle für deren Einsatz. Aus computerlinguistischer Sicht sind die Entscheidbarkeitseigenschaften des **Wortproblems** besonders interessant. Dieses Wortproblem lautet bei Eingabe einer Grammatik $G = \langle \Phi, \Sigma, R, S \rangle$ und einem Wort $w \in \Sigma^*$: Wird entschieden, ob $w \in L(G)$ oder $w \notin L(G)$ gilt? Falls diese Frage bejaht wird, ist das Wortproblem entscheidbar. Bei Verneinung ist das Wortproblem nicht entscheidbar.

Da der Berechenbarkeitsbegriff auf Funktionen zugeschnitten ist, lässt sich die Beziehung zwischen Berechenbarkeit und Entscheidbarkeit durch die Angabe der **charakteristischen Funktion** explizieren. Die charakteristische Funktion ist im Unterkapitel 2.1 zur Mengenlehre und Logik definiert.

Definition 2.2.16

Eine formale Sprache $S \subseteq \Sigma^*$ heißt entscheidbar, falls die charakteristische Funktion von S ($C_S: \Sigma^* \rightarrow \{0, 1\}$) berechenbar ist. Für alle $w \in \Sigma^*$ gilt:

$$C_S(w) = \begin{cases} 1, & \text{wenn } w \in S \\ 0, & \text{wenn } w \notin S \end{cases}$$

S heißt semi-entscheidbar, wenn die halbe charakteristische Funktion C'_S berechenbar ist:

$$C'_S(w) = \begin{cases} 1, & \text{wenn } w \in S \\ \text{undefiniert}, & \text{wenn } w \notin S \end{cases}$$

□

Das Wortproblem ist für Typ-3, Typ-2 und Typ-1 Sprachen entscheidbar. Typ-0-Sprachen sind hingegen nicht entscheidbar. Die Komplexität bei der Entscheidung, ob eine Zeichenkette Element einer formalen Sprache ist, ist für die anderen Sprachen wie folgt bei n als Länge der Zeichenkette: Die Typ-1-Sprachen sind exponentiell und Typ-2-Sprachen sind kubisch (n^3). Deterministisch-kontextfreie und Typ-3-Sprachen sind linear (n). Letztere sind linear, wenn die Sprache durch einen deterministischen endlichen Automaten gegeben ist, denn dann

müssen nur zeichenweise die Übergänge im Automaten verfolgt werden bis zu einem Endzustand.

Auf der Basis des Turingmaschinen-Modells können rekursiv aufzählbare von den rekursiven Mengen unterschieden werden. Eine formale Sprache, die von einer Turingmaschine akzeptiert wird, heißt **rekursiv aufzählbar**. **Rekursive Sprachen** sind Teilmengen der rekursiv aufzählbaren Sprachen. Dies sind die Sprachen, die von mindestens einer Turingmaschine erkannt werden, die auf allen Eingaben, also auch den Zeichenketten, die nicht zu der von der Turingmaschine akzeptierten Sprache gehören, hält.

Um den Unterschied zwischen rekursiven und rekursiv aufzählbaren Mengen zu verstehen, muss man wissen, was sich hinter diesem angedeuteten Halteproblem der Turingmaschine verbirgt. Analog zu den endlichen Automaten und den Kellerautomaten geht eine Turingmaschine für eine spezielle Sprache L irgendwann in einen Endzustand über, wenn sie ein Wort $w \in L$ akzeptiert. Die Turingmaschine „hält an“. Falls die Turingmaschine eine Zeichenkette liest, die nicht Element der von der Maschine akzeptierten Sprache ist, kann es der Fall sein, dass die Turingmaschine nicht in irgendeinem Zustand (außer den Endzuständen) hält, sondern überhaupt nicht. Dieses Halteproblem einer Turingmaschine stellt letztlich nur eine andere Sicht auf Entscheidbarkeit dar. Entscheidbare Probleme sind solche, bei denen eine klare ja/nein-Antwort auf die Frage existiert, ob eine Instanz des Problems von einer Turingmaschine akzeptiert wird. Berechenbarkeit bedeutet, dass das Problem durch eine Funktion beschrieben wird, also für jede Eingabe genau eine Ausgabe berechnet wird.

Die allgemeinen Regelsprachen sind somit äquivalent zu den rekursiv aufzählbaren Sprachen. Die rekursiven Sprachen sind gegen Vereinigung, Verkettung, Sternbildung, Schnitt, Differenz und Komplement abgeschlossen. Die rekursiv aufzählbaren Sprachen sind gegen Vereinigung, Verkettung, Sternbildung und Schnitt abgeschlossen, nicht aber gegen Komplementbildung und Differenz. Die Beziehung zwischen einer rekursiv aufzählbaren Sprache und ihrem Komplement betrifft die Unterscheidung zwischen rekursiven und rekursiv aufzählbaren Sprachen. Ein Algorithmus, der für jedes Wort $w \in \Sigma^*$ feststellt, ob w Element einer rekursiv aufzählbaren Sprache L oder Element des Komplements \overline{L} ist, würde immer ein Ergebnis liefern. Dann wären aber die rekursiv aufzählbaren Sprachen rekursiv und die Unterscheidung zwischen beiden Sprachklassen überflüssig. Das Komplement einer rekursiv aufzählbaren Sprache ist also noch nicht einmal rekursiv aufzählbar. Die Klasse der rekursiv aufzählbaren Sprachen ist somit gegen Komplementbildung nicht abgeschlossen. Aus diesem Grund sind die rekursiv aufzählbaren Sprachen auch nicht bzgl. Differenz abgeschlossen.

2.2.7 Zusammenfassung

Die vier wichtigsten Sprachklassen sind die regulären, kontextfreien, kontextsensitiven und die allgemeinen Regelsprachen. Diese Sprachen werden auch als Typ-3-, Typ-2-, Typ-1- und Typ-0-Sprachen bezeichnet. Zwischen diesen Sprachklassen existiert eine echte Teilmengenbeziehung, es gilt also: $\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0$. Aus historischen Gründen wird diese Einteilung der Sprachtypen auch als

Chomsky-Hierarchie bezeichnet. Zusätzlich wurden die deterministisch kontextfreien Sprachen und die durch TAGs erzeugten Sprachen charakterisiert, so dass insgesamt die folgende Beziehung gilt:

$$\mathcal{L}_3 \subset \mathcal{L}_{det.\,kontextfrei} \subset \mathcal{L}_2 \subset \mathcal{L}_{TAGs} \subset \mathcal{L}_1 \subset \mathcal{L}_0$$

Die Typ-3-Sprachen werden von rechts- und links-linearen Grammatiken erzeugt und von den deterministischen sowie den nichtdeterministischen endlichen Automaten akzeptiert. Die deterministisch kontextfreien Sprachen werden von den deterministischen Kellerautomaten erkannt, während die Typ-2-Sprachen genau von kontextfreien Grammatiken generiert und von den nichtdeterministischen Kellerautomaten akzeptiert werden. Baumadjunktions-Grammatiken als schwach kontextsensitive Grammatiken erzeugen Sprachen, die echt zwischen den Typ-2- und den Typ-1-Sprachen liegen. Typ-1-Sprachen werden von kontextsensitiven Grammatiken erzeugt und von linear beschränkten Automaten akzeptiert. Typ-0- bzw. rekursiv aufzählbare Sprachen schließlich werden von allgemeinen Regelgrammatiken generiert und von (deterministischen und nicht-deterministischen) Turingmaschinen akzeptiert.

2.2.8 Literaturhinweise

Die Ergebnisse zu den Eigenschaften formaler Sprachen und Automaten haben im Gegensatz zu anderen Gebieten eine lange Aktualität. Daher ist die Bibel der theoretischen Informatik, Hopcroft und Ullman (1979) bzw. Hopcroft und Ullman (1994), immer noch eine grundlegende Quelle für relevante Aussagen und Beweise. Sehr gute deutschsprachige Darstellungen der Grundlagen der theoretischen Informatik bieten Bucher und Maurer (1984), Schöning (1999) und Erk und Priese (2002). Eine auch für Nicht-Informatiker verständliche englischsprachige Darstellung liefert Brookshear (1989). Levelt (2008) ist eine überarbeitete Neuauflage eines Bands aus einem dreiteiligen Buch über formale Grammatiken in der (Psycho)Linguistik aus dem Jahr 1974. Dass dieses Buch nach 35 Jahren wieder aufgelegt wird, zeigt zum einen die Gültigkeit der formulierten Aussagen und Verfahren und zum anderen, dass diese Konzepte aus der theoretischen Informatik noch immer für die Methodenentwicklung relevant sind.

Aspekte der theoretischen Informatik, die über diesen einführenden Artikel hinausgehen – insbesondere der gesamte Bereich der Komplexitätstheorie – sind ebenfalls in Hopcroft und Ullman (1979) sowie in Brookshear (1989) zu finden.

2.3 Graphentheorie und Merkmalsstrukturen

Peter Kolb

Die Graphentheorie ist in der Computerlinguistik eine unentbehrliche Grundlage für die Beschreibung linguistischer Objekte und Strukturen. Die Repräsentation von Information in Form eines Graphen ermöglicht es insbesondere, hierarchische Beziehungen zu beschreiben, so dass vor allem strukturelle Darstellungen auf der Graphentheorie fußen. Dies zeigte sich bereits im vorigen Unterkapitel 2.2 über formale Sprachen bei der Darstellung von Ableitungsbäumen und endlicher Automaten, die beides spezielle Graphen sind. Graphen spielen aber auch in der Morphologie und insbesondere in der Syntax eine gewichtige Rolle.

Merkmalstrukturen sind gerichtete Graphen für computerlinguistisch relevante Repräsentationen linguistischer Information. Sie werden bei der syntaktischen Analyse eingesetzt (vgl. das Unterkapitel 3.5), aber auch in der Phonologie (siehe 3.1), der Morphologie (3.3) und sogar der Semantik (3.6) bilden Merkmalsstrukturen die Datenstrukturen für die jeweilige Information.

2.3.1 Graphen und Bäume

Angenommen man möchte auf einer Zugreise eine Reihe von Städten besuchen. Gesucht ist dazu ein Weg, der genau einmal über jede Stadt führt. Um einen solchen Weg zu finden, könnte man sich die zwischen den einzelnen Städten bestehenden Zugverbindungen in einer Skizze wie Abbildung 2.8 aufzeichnen. Mit Hilfe dieser Zeichnung ist es einfach, einen geeigneten Weg zu finden.

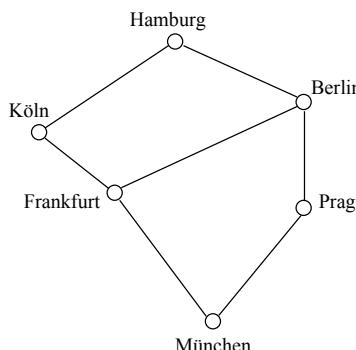


Abbildung 2.8: Städteverbindungen als Graph

Es gibt viele Probleme, die sich am einfachsten als Menge von Objekten und Beziehungen zwischen diesen darstellen lassen. Ein mathematisches Hilfsmittel um solche Situationen exakt zu modellieren, ist ein **Graph**. Bei Abbildung 2.8 handelt es sich um einen Graphen. Die Objekte (hier die Städte) werden als **Knoten**, die Verbindungen (hier Zugverbindungen) als **Kanten** bezeichnet.

Definition 2.3.1

Ein **Graph** ist ein geordnetes Paar $G = \langle N, E \rangle$ aus Knoten und Kanten.

- N ist eine endliche Menge aus Knoten.
- E ist eine endliche Menge aus Kanten. Kanten sind ungeordnete Paare aus Knoten.

□

Für den Graphen in Abbildung 2.8 gilt:

$$\begin{aligned} N &= \{\text{Hamburg, Köln, Berlin, Frankfurt, München, Prag}\} \\ E &= \{(\text{Hamburg, Berlin}), (\text{Hamburg, Köln}), (\text{Köln, Frankfurt}), \\ &\quad (\text{Berlin, Frankfurt}), (\text{Frankfurt, München}), (\text{München, Prag}), \\ &\quad (\text{Prag, Berlin})\} \end{aligned}$$

Ein **Pfad** von Knoten x zu Knoten y ist eine Reihe von Knoten, die durch Kanten im Graphen miteinander verbunden sind. Beispielsweise ist der Weg von Hamburg über Köln nach Frankfurt ein Pfad.

Ein Graph ist **zusammenhängend**, wenn von jedem Knoten zu jedem anderen Knoten im Graphen ein Pfad führt.

Ein Pfad, auf dem sich kein Knoten wiederholt, ist ein **einfacher Pfad**. Ein **Zyklus** ist ein einfacher Pfad, bei dem der erste und der letzte Knoten identisch sind. Die Lösung des Reiseproblems ist ein Zyklus, der z. B. von Hamburg ausgehend über alle Städte wieder zurück nach Hamburg führt.

Ein Graph G wird als **Untergraph** eines Graphen H bezeichnet, wenn alle Knoten von G auch zu H gehören und durch dieselben Kanten verbunden sind.

Die Reihenfolge innerhalb der Kanten spielte bisher keine Rolle. Die Kante (Hamburg, Köln) ist identisch mit (Köln, Hamburg), da es sich um ungeordnete Paare handelt. Wenn es zwar eine Zugverbindung von Köln nach Hamburg, nicht aber von Hamburg nach Köln gibt, kann man dieses Verhältnis in einem **gerichteten** Graphen ausdrücken.

Definition 2.3.2

Ein **gerichteter Graph** ist ein geordnetes Paar $G = \langle N, E \rangle$ aus Knoten und Kanten.

- N ist eine endliche Menge aus Knoten.
- E ist eine endliche Menge aus gerichteten Kanten. Eine gerichtete Kante ist ein geordnetes Paar aus Knoten.

□

Eine gerichtete Kante wird als Pfeil dargestellt. Abbildung 2.9 besteht aus den Kanten

$$E = \{\langle \text{Köln, Hamburg} \rangle, \langle \text{Hamburg, Berlin} \rangle, \langle \text{Berlin, Hamburg} \rangle\}$$

In Abbildung 2.9 sind die Kanten zusätzlich mit den Entfernungen zwischen den Städten markiert. Man spricht von einem **markierten Graphen**.

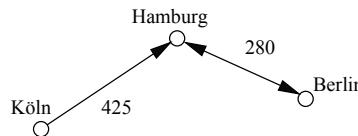


Abbildung 2.9: Ein gerichteter markierter Graph

Die Darstellung und Verarbeitung von Graphen im Computer ist ein grundlegendes Gebiet der Informatik. Standardwerke sind Knuth (1969) und Wirth (1983), eine neuere Darstellung bietet Sedgewick (1992).

Bäume

Ein wichtiger Spezialfall der Graphen sind die Bäume. Treten in einem Graphen keine Zyklen auf, so handelt es sich um einen **Baum**. Zwischen je zwei beliebigen Knoten in einem Baum besteht genau ein Pfad, der sie verbindet.

Einer der Knoten im Baum wird **Wurzel** genannt. Bäume werden mit der Wurzel an der Spitze dargestellt (in Abbildung 2.10 ist *A* die Wurzel). Der Knoten *y* befindet sich unter dem Knoten *x*, wenn *x* auf dem Pfad von *y* zur Wurzel liegt. In diesem Fall sagt man, der Knoten *x* **dominiert** den Knoten *y*. Liegt zwischen dem Knoten *y* und dem dominierenden Knoten *x* kein weiterer Knoten, dann **dominiert x y unmittelbar**. In Abbildung 2.10 dominiert *A* alle Knoten, jedoch nur *B*, *C* und *D* unmittelbar.

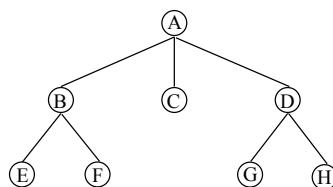


Abbildung 2.10: Ein Baum

Jeder Knoten in einem Baum (außer der Wurzel) besitzt genau einen Knoten, der ihn unmittelbar dominiert und als sein **Mutterknoten** bezeichnet wird. Die Knoten unmittelbar unter einem Knoten werden als seine **Töchter** bezeichnet. Tochterknoten desselben Mutterknotens heißen **Schwestern**.

Knoten ohne Nachfolger werden **Blätter** oder **Endknoten** genannt. Endknoten werden auch als **äußere Knoten**, Nichtendknoten als **innere Knoten** bezeichnet. In Abbildung 2.10 handelt es sich bei *A*, *B* und *D* um innere Knoten, alle übrigen sind äußere Knoten.

Jeder Knoten ist die Wurzel eines **Unterbaumes**, welcher aus ihm und den Knoten unter ihm besteht.

Die Knoten eines Baumes befinden sich auf verschiedenen **Ebenen**. Die Ebene eines Knotens ergibt sich durch die Anzahl der Knoten auf dem Pfad vom jeweiligen Knoten zur Wurzel (ihn selbst nicht mitgezählt). Die Höhe eines Baumes ist der maximale Abstand zwischen einem beliebigen Knoten und der Wurzel. Der Baum in Abbildung 2.10 hat die Höhe 2, der Knoten *A* hat die Ebene 0, die Knoten *B*, *C* und *D* sind auf Ebene 1, die restlichen Knoten auf Ebene 2.

Man kann festlegen, dass jeder Knoten eine bestimmte Anzahl von Töchtern haben muss, die in einer bestimmten Reihenfolge angeordnet sind. In diesem Fall handelt es sich um einen **n-ären Baum**. Der einfachste Typ eines n-ären Baums ist der **binäre Baum**, bei dem jeder Knoten maximal zwei Tochterknoten besitzt.

2.3.2 Merkmalsstrukturen

Die in kontextfreien Grammatiken (siehe Unterkapitel 2.2) verwendeten Nichtterminalsymbole sind aus linguistischer Sicht nicht ausreichend, um Generalisierungen darzustellen. Z. B. müsste eine Grammatik, die singulare und pluri le Nominalphrasen wie *das Schaf – die Schafe* erzeugt, hierfür unterschiedliche Nichtterminalsymbole verwenden wie z. B. NP_{sg} und NP_{pl} . Formal sind dies völlig unterschiedliche Variablen, die auf Grund ihrer Ähnlichkeit eine strukturelle Ähnlichkeit nur suggerieren. Man erhält einen flexibleren und ausdrucksstärkeren Formalismus, wenn man die Nichtterminalsymbole als atomare Kategorien in **komplexe Kategorien** (siehe Unterkapitel 3.5) aufspaltet.

Zur Modellierung komplexer Kategorien werden **Merkmalsstrukturen** (auch **feature structure**, **Attribut-Wert-Struktur** oder **attribute-value matrix** genannt) verwendet. Eine Merkmalsstruktur besteht aus einer Menge von Merkmalspezifikationen. Das sind Paare grammatischer Merkmale und zugehöriger Werte. Merkmalsstrukturen werden meist als Matrizen dargestellt:

$$(2.20) \quad S = \begin{bmatrix} \text{MERKMAL1} & \text{WERT1} \\ \text{MERKMAL2} & \text{WERT2} \end{bmatrix}$$

Diese Merkmalsstruktur besteht aus zwei Merkmals-Wert-Paaren. Im ersten Paar wird dem MERKMAL1 der Wert WERT1 zugewiesen.

Statt mit der atomaren Kategorie *Verb* kann ein Wort wie *bellt* mit Hilfe einer Merkmalsstruktur genauer beschrieben werden:

$$(2.21) \quad S_{3sg} = \begin{bmatrix} \text{KAT} & \text{VERB} \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \end{bmatrix}$$

Neben der Information, dass *bellt* zur Kategorie (abgekürzt als KAT) Verb gehört, wurde in dessen Beschreibung zusätzlich aufgenommen, dass es in der dritten Person (PERS) steht und der Numerus (NUM) Singular ist. Die Merkmalsstruktur S_{3sg} repräsentiert alle Verben in der dritten Person Singular.

Unterspezifikation

Sieht man Merkmalsstrukturen als Repräsentationen linguistischer Objekte an, ist interessant, dass sie im Allgemeinen nur partielle Informationen über deren Merkmale liefern. Vom Wort *Kollegen*, isoliert betrachtet, ist z. B. nur bekannt, dass es der Wortart Nomen angehört und ein Maskulinum ist, Numerus und Kasus sind nicht festgelegt. In einer Merkmalsstruktur können Merkmale, deren Werte noch nicht bekannt sind, unspezifiziert gelassen werden:

$$(2.22) \quad S_{\text{Kollegen}} = \begin{bmatrix} \text{KAT} & \text{NOMEN} \\ \text{GEN} & \text{MASK} \end{bmatrix}$$

Damit ist es möglich, ein Modell eines Objekts herzustellen mit dem, was bisher darüber bekannt ist und abzuwarten, bis durch den Kontext des Wortes in einer sprachlichen Äußerung weitere Information hinzukommt.

Durch diese **Unterspezifikation** können Verallgemeinerungen vorgenommen werden. Man kann durch Angabe aller Merkmalswerte ein spezifisches linguistisches Objekt benennen, gleichzeitig aber auch durch Weglassen der Merkmale immer allgemeinere Klassen von Objekten zusammenfassen. Abbildung 2.11 zeigt, wie durch Merkmalsstrukturen Mengen von Objekten beschrieben werden. Die Angabe eines zusätzlichen Merkmals ergibt eine neue Teilmenge der Ausgangsstruktur. Beispielsweise ist die Menge der Verben im Singular eine Teilmenge aller Verben.

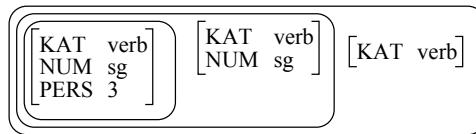


Abbildung 2.11: Merkmalsstrukturen beschreiben Mengen

Je mehr Merkmale in einer Merkmalsstruktur angegeben werden, desto kleiner wird die Menge der durch diese Merkmalsstruktur repräsentierten Objekte, denn die einzelnen Merkmale einer Merkmalsstruktur sind implizit durch „und“ verknüpft (Konjunktionen, siehe Unterkapitel 2.1). Je mehr Merkmals-Wert-Kombinationen angegeben werden, desto weniger Objekten gelingt es, alle diese Einschränkungen gleichzeitig zu erfüllen.

Man verfügt also über einen flexiblen Formalismus, in dem man mit beliebigen Unterklassen arbeiten und trotzdem noch die Kategorien als Ganze ansprechen kann.

Atomare und komplexe Werte

Merkmale können zwei Arten von Werten annehmen. Erstens solche, die nicht weiter aufteilbar sind, wie z. B. VERB oder SG. Diese heißen **atomare Werte**.

Zweitens kann ein Merkmal als Wert wiederum eine vollständige Merkmalsstruktur erhalten. Man spricht dann von einem **komplexen Wert**. Es treten also Merkmalsstrukturen innerhalb von Merkmalsstrukturen auf.

Mittels komplexer Werte lassen sich z. B. die für die Kongruenz relevanten Merkmale NUM (Numerus), GEN (Genus) und KAS (Kasus) unter einem Merkmal KGR (Kongruenz) zusammenfassen:

$$(2.23) \quad \begin{bmatrix} \text{KAT} & \text{NOMEN} \\ \text{KGR} & \begin{bmatrix} \text{NUM} & \text{SG} \\ \text{GEN} & \text{MASK} \\ \text{KAS} & \text{AKK} \end{bmatrix} \end{bmatrix}$$

Graphennotation

Merkmalsstrukturen können als gerichtete Graphen dargestellt werden, wobei die Merkmale die Kanten bezeichnen. Die Kanten einer Merkmalsstruktur gehen von einem gemeinsamen Wurzelknoten aus. Die Endknoten werden mit den Merkmalswerten annotiert, die übrigen Knoten haben keine Bezeichnungen. Ein Weg vom Wurzelknoten zu einem Endknoten ist ein **vollständiger Pfad**. Jeder atomare Wert in einer Merkmalsstruktur wird durch einen vollständigen Pfad identifiziert.

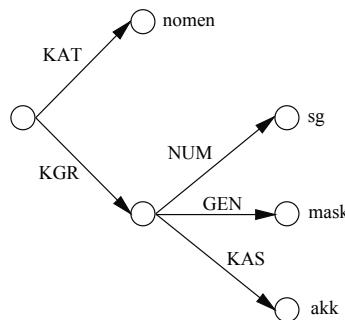


Abbildung 2.12: Die Merkmalsstruktur aus Beispiel 2.23 als gerichteter Graph

Da Merkmalsstrukturen sehr umfangreich und unübersichtlich werden können, gibt es in der Matrixnotation eine häufig verwendete Abkürzung. Interessiert z. B. an der Struktur in Beispiel 2.23 nur der Wert des Merkmals KAS, kann die Struktur abkürzend so geschrieben werden:

$$(2.24) \quad \begin{bmatrix} \text{KAT} & \text{NOMEN} \\ \text{KGR} | \text{KAS} & \text{AKK} \end{bmatrix}$$

Es wird lediglich der Pfad von der höchsten Ebene der Struktur bis zum jeweils relevanten Merkmal angegeben. Die auf diesem Pfad liegenden Merkmale werden durch einen senkrechten Strich voneinander getrennt.

Koreferenz

Betrachtet man einzelne Wörter wie *Hund*, *der* und *bellt*, kann man die Werte vieler ihrer grammatischen Merkmale nicht festlegen. Dennoch weiß man, dass in Satz 2.25 die Phrase *der Hund* in Person und Numerus mit dem Verb *bellt* übereinstimmen muss, weil im Deutschen Subjekt und Prädikat eines Satzes in Person und Numerus kongruieren.

(2.25) *der Hund bellt*

Derartiges Wissen kann man innerhalb von Merkmalsstrukturen durch **Koreferenz** ausdrücken. Bei der Koreferenz besitzen zwei Merkmale denselben Wert. Dies wird dargestellt durch nummerierte Kästchen. Gleiche Indizes bei verschiedenen Werten geben an, dass ein Wert für mehrere Merkmale gilt. Die Information selbst wird zur Verdeutlichung nur bei einem der koreferenten Werte eingetragen, die anderen koreferenten Werte verweisen darauf. Durch die Koreferenz in der folgenden Merkmalsstruktur wird erzwungen, dass die Werte der Merkmale KGR bei Subjekt und Prädikat stets identisch sind:

(2.26)	$\begin{bmatrix} \text{KAT} & \text{SATZ} \\ \text{SUBJ} & \left[\begin{array}{cc} \text{KAT} & \text{NP} \\ \text{KGR} & \boxed{1} \left[\begin{array}{cc} \text{NUM} & \text{sg} \end{array} \right] \end{array} \right] \\ \text{PRÄD} & \left[\begin{array}{cc} \text{KAT} & \text{VP} \\ \text{KGR} & \boxed{1} \end{array} \right] \end{bmatrix}$
--------	--

Die Merkmale SUBJ|KGR und PRÄD|KGR teilen sich einen gemeinsamen Wert (nämlich den für Numerus). Wird der Wert bei einem der koreferenten Merkmale geändert, ist davon auch das andere Merkmal betroffen.

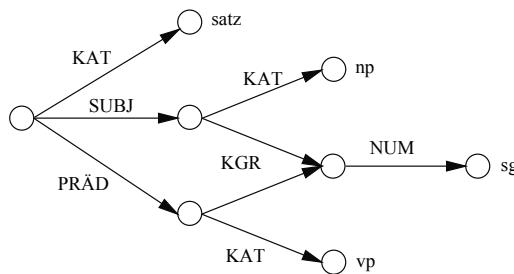


Abbildung 2.13: Koreferenz im Graphen

Zwischen Merkmalsstrukturen mit koreferenten Werten und Merkmalsstrukturen mit nur zufällig den gleichen Werten muss genau unterschieden werden. Eine Koreferenz innerhalb einer Merkmalsstruktur heißt auch **Strukturteilung** oder **Pfadäquivalenz**. Der Wert wird nicht von einem Merkmal zum anderen kopiert,

sondern die Merkmale teilen sich ein und denselben Wert. In der Graphennotation in Abbildung 2.13 wird das deutlicher: mehrere Kanten führen zu einem Knoten.

Disjunkte Werte

Im Fall einer isolierten Artikelform wie *den* kennt man zwar die tatsächlichen Werte von Numerus, Genus und Kasus nicht, man kann die möglichen Werte aber einschränken. Die Artikelform *den* kann entweder ein Maskulinum im Akkusativ Singular sein, oder in allen Genera ein Dativ Plural. In einer Merkmalsstruktur wird dieses „oder“ durch eine **disjunktive Verknüpfung** (siehe Unterkapitel 2.1) der Werte ausgedrückt:

$$(2.27) \quad S_{\text{den}} = \left[\begin{array}{cc} \text{KAT} & \text{ART} \\ \text{KGR} & \left[\begin{array}{cc} \text{NUM} & \text{SG} \\ \text{GEN} & \text{MASK} \\ \text{KAS} & \text{AKK} \end{array} \right] \end{array} \right] \vee \left[\begin{array}{cc} \text{NUM} & \text{PL} \\ \text{KAS} & \text{DAT} \end{array} \right]$$

Subsumptionsrelation

Es ist möglich, Merkmalsstrukturen nach ihrem Informationsgehalt anzurufen. Beschreibt man beispielsweise eine Verbalphrase durch

$$(2.28) \quad \left[\begin{array}{cc} \text{KAT} & \text{VP} \\ \text{PERS} & 2 \\ \text{NUM} & \text{SG} \end{array} \right]$$

dann legt man fest, dass diese Phrase bestimmte Werte für Numerus und Person hat. Die Merkmalsstruktur

$$(2.29) \quad \left[\begin{array}{cc} \text{KAT} & \text{VP} \end{array} \right]$$

beschreibt ebenfalls eine Verbalphrase, lässt aber die Frage nach Person und Numerus offen. Die erste Merkmalsstruktur enthält offensichtlich mehr Information als die zweite, die zweite ist allgemeiner und beschreibt eine größere Anzahl von Objekten. Man sagt, die allgemeinere Merkmalsstruktur **subsumiert** die spezifischere.

Definition 2.3.3

Eine Merkmalsstruktur S_1 **subsumiert** eine Merkmalsstruktur S_2 ($S_1 \sqsubseteq S_2$), wenn in S_2 mindestens die Information aus S_1 enthalten ist. \square

Einige Beispiele sollen das Gesagte verdeutlichen:

$$(2.30) \quad S_1 = \begin{bmatrix} \text{KAT} & \text{VP} \end{bmatrix}$$

$$(2.31) \quad S_2 = \begin{bmatrix} \text{KAT} & \text{VP} \\ \text{KGR} & \begin{bmatrix} \text{NUM} & \text{SG} \end{bmatrix} \end{bmatrix}$$

$$(2.32) \quad S_3 = \begin{bmatrix} \text{KAT} & \text{VP} \\ \text{KGR} & \begin{bmatrix} \text{NUM} & \text{SG} \\ \text{PERS} & 2 \end{bmatrix} \end{bmatrix}$$

$$(2.33) \quad S_4 = \begin{bmatrix} \text{KAT} & \text{VP} \\ \text{KGR} & \begin{bmatrix} \text{NUM} & \text{SG} \\ \text{PERS} & 2 \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{KGR} & \begin{bmatrix} \text{NUM} & \text{SG} \\ \text{PERS} & 2 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$(2.34) \quad S_5 = \begin{bmatrix} \text{KAT} & \text{VP} \\ \text{KGR} & \boxed{1} \begin{bmatrix} \text{NUM} & \text{SG} \\ \text{PERS} & 2 \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{KGR} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

Es gilt: $S_1 \sqsubseteq S_2 \sqsubseteq S_3 \sqsubseteq S_4 \sqsubseteq S_5$.

Man beachte, dass $S_4 \sqsubseteq S_5$. In S_4 besitzen die Merkmale KGR und OBJ|KGR nur zufällig gleiche Werte. In S_5 dagegen wird durch die Koreferenz die zusätzliche Information gegeben, dass die beiden Merkmale *stets* denselben Wert haben.

Die allgemeinste Merkmalsstruktur, die alle anderen Merkmalsstrukturen subsumiert, ist diejenige, die überhaupt keine Information enthält, nämlich die **leere Merkmalsstruktur**: $[] \sqsubseteq S$ für jede beliebige Merkmalstruktur S .

Eine Ordnungsrelation wie die Subsumption besitzt die folgenden Eigenschaften:

1. Reflexivität: Jede Struktur S subsumiert sich selbst: $S \sqsubseteq S$ für alle S .
2. Transitivität: Wenn $S_1 \sqsubseteq S_2$ und $S_2 \sqsubseteq S_3$, dann $S_1 \sqsubseteq S_3$.
3. Antisymmetrie: Wenn $S_1 \sqsubseteq S_2$ und $S_2 \sqsubseteq S_1$, dann gilt $S_1 = S_2$.

2.3.3 Unifikation

Auf einer geordneten Menge können Operationen eingeführt werden, die zwei Elemente der Menge so miteinander verknüpfen, dass ein neues Element der Menge entsteht.

Weil nicht alle beliebigen Merkmalsstrukturen miteinander vergleichbar sind, bildet die Subsumption keine vollständige Ordnung, sondern eine partielle Ordnung. Die folgenden beiden Merkmalsstrukturen enthalten unterschiedliche, aber miteinander zu vereinbarende Information:

$$(2.35) \quad S_a = \begin{bmatrix} \text{KAT} & \text{NP} \\ \text{KGR} & \begin{bmatrix} \text{NUM} & \text{SG} \end{bmatrix} \end{bmatrix}$$

$$(2.36) \quad S_b = \begin{bmatrix} \text{KAT} & \text{NP} \\ \text{KGR} & \begin{bmatrix} \text{GEN} & \text{FEM} \end{bmatrix} \end{bmatrix}$$

Hier ist es nicht möglich zu sagen, dass eine Struktur die andere subsumieren würde. Die beiden Strukturen sind anhand der Subsumptionsrelation nicht vergleichbar.

Die folgenden beiden Merkmalsstrukturen enthalten inkompatible Information und stehen ebenfalls in keiner Subsumptionsrelation zueinander:

$$(2.37) \quad S_a = \begin{bmatrix} \text{KAT} & \text{NP} \\ \text{KGR} & \begin{bmatrix} \text{NUM} & \text{SG} \end{bmatrix} \end{bmatrix}$$

$$(2.38) \quad S_c = \begin{bmatrix} \text{KAT} & \text{NP} \\ \text{KGR} & \begin{bmatrix} \text{NUM} & \text{PL} \end{bmatrix} \end{bmatrix}$$

Der Unterschied zwischen den beiden Fällen ist, dass es im ersten Fall eine spezifischere Merkmalsstruktur gibt, die von beiden Merkmalsstrukturen subsumiert wird, nämlich:

$$(2.39) \quad S_d = \begin{bmatrix} \text{KAT} & \text{NP} \\ \text{KGR} & \begin{bmatrix} \text{NUM} & \text{SG} \\ \text{GEN} & \text{FEM} \end{bmatrix} \end{bmatrix}$$

während im zweiten Fall keine solche Merkmalsstruktur existiert. Diese Idee, die Information, die in zwei Merkmalsstrukturen enthalten ist, in einer einzigen Struktur zu vereinigen, wird durch die **Unifikation** realisiert.

Es gibt weitere Merkmalsstrukturen, die von den beiden Merkmalsstrukturen S_a und S_b subsumiert werden, beispielsweise:

$$(2.40) \quad S_e = \begin{bmatrix} \text{KAT} & \text{NP} \\ & \begin{bmatrix} \text{NUM} & \text{SG} \\ \text{GEN} & \text{FEM} \\ \text{KAS} & \text{DAT} \end{bmatrix} \\ \text{KGR} & \end{bmatrix}$$

Man ist aber an der allgemeinsten Merkmalsstruktur dieser Art interessiert; denjenigen, die die ganze Information der beiden unifizierten Merkmalsstrukturen enthält und sonst nichts.

Definition 2.3.4

Die Merkmalsstruktur S_3 heißt **Unifikation** von S_1 und S_2 genau dann, wenn S_3 sowohl von S_1 als auch von S_2 subsumiert wird und S_3 alle anderen Merkmalsstrukturen subsumiert, die ebenfalls von S_1 und S_2 subsumiert werden: $S_1 \sqcup S_2 = S_3$. \square

Abbildung 2.14 zeigt eine Subsumptionshierarchie mit den obigen und einer weiteren Merkmalsstruktur, dargestellt als **Hasse-Diagramm**.

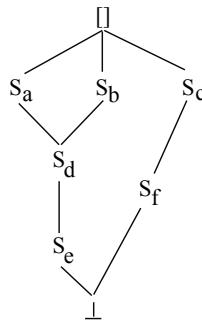


Abbildung 2.14: Eine Subsumptionshierarchie

An der Spitze steht die leere Merkmalsstruktur, die alle anderen Merkmalsstrukturen subsumiert. Gelangt man von einer Merkmalsstruktur S_i entlang der Kanten abwärts zu einer Merkmalsstruktur S_k , dann gilt $S_i \sqsubseteq S_k$. Beispielsweise subsumiert S_c S_f , aber nicht S_e .

Die Unifikation von S_a und S_b muss eine Merkmalsstruktur ergeben, die von beiden subsumiert wird. Es kommen also nur S_d und S_e in Frage. Außerdem muss die Ergebnis-Struktur alle anderen Merkmalsstrukturen subsumieren, die ebenfalls noch von den beiden Ausgangs-Strukturen subsumiert werden. Da S_d S_e subsumiert, ist das Ergebnis der Unifikation von S_a und S_b die Merkmalsstruktur S_d : $S_a \sqcup S_b = S_d$.

In einer Ordnungsrelation wird das Ergebnis einer Operation wie der Unifikation auch als das **Infimum** der unifizierten Elemente bezeichnet. Damit die Unifikation wohldefinierte Eigenschaften besitzt, ist es wichtig, dass zu zwei beliebigen Merkmalsstrukturen stets ein Infimum existiert. Um die Existenz eines Infimums für zwei beliebige Merkmalsstrukturen zu garantieren, führt man das

Symbol \perp (Bottom) ein und setzt es an das untere Ende der Subsumptionshierarchie. \perp bezeichnet die inkonsistente Merkmalsstruktur. \perp ist das Ergebnis der Unifikation zweier Strukturen mit widersprüchlicher Information, wie z. B. S_c und S_e . Durch diesen Trick haben zwei beliebige Merkmalsstrukturen als Unifikationsergebnis entweder eine Struktur, die die Information aus beiden vereint, oder \perp . In jedem Fall ist die Existenz eines Infimums garantiert.

Beispiel 2.3.1

Beispiele für die Unifikation von Merkmalsstrukturen sind:

$$(2.41) \quad \left[\begin{array}{cc} \text{KAT} & \text{NP} \end{array} \right] \sqcup \left[\begin{array}{cc} \text{KGR} & \left[\begin{array}{cc} \text{NUM} & \text{SG} \\ \text{GEN} & \text{FEM} \end{array} \right] \end{array} \right] = \left[\begin{array}{cc} \text{KAT} & \text{NP} \\ \text{KGR} & \left[\begin{array}{cc} \text{NUM} & \text{SG} \\ \text{GEN} & \text{FEM} \end{array} \right] \end{array} \right]$$

$$(2.42) \quad \left[\begin{array}{cc} \text{KAT} & \text{VP} \\ \text{KGR} & \left[\begin{array}{cc} \text{NUM} & \text{SG} \end{array} \right] \\ \text{OBJ} & \left[\begin{array}{cc} \text{KAT} & \text{NP} \\ \text{KGR} & \left[\begin{array}{cc} \text{KAS} & \text{AKK} \end{array} \right] \end{array} \right] \end{array} \right] \sqcup \left[\begin{array}{cc} \text{KGR} & \left[\begin{array}{cc} \text{NUM} & \text{PL} \end{array} \right] \end{array} \right] = \perp$$

$$(2.43) \quad \left[\begin{array}{cc} \text{KGR} & \left[\begin{array}{cc} \text{NUM} & \text{SG} \end{array} \right] \\ \text{SUBJ} & \left[\begin{array}{cc} \text{KGR} & \left[\begin{array}{cc} \text{NUM} & \text{SG} \end{array} \right] \end{array} \right] \end{array} \right] \sqcup \left[\begin{array}{cc} \text{SUBJ} & \left[\begin{array}{cc} \text{KGR} & \left[\begin{array}{cc} \text{PERS} & 3 \end{array} \right] \end{array} \right] \end{array} \right] =$$

$$\left[\begin{array}{cc} \text{KGR} & \left[\begin{array}{cc} \text{NUM} & \text{SG} \end{array} \right] \\ \text{SUBJ} & \left[\begin{array}{cc} \text{KGR} & \left[\begin{array}{cc} \text{NUM} & \text{SG} \\ \text{PERS} & 3 \end{array} \right] \end{array} \right] \end{array} \right]$$

$$(2.44) \quad \left[\begin{array}{cc} \text{KGR} & \boxed{1} \left[\begin{array}{cc} \text{NUM} & \text{SG} \end{array} \right] \\ \text{SUBJ} & \left[\begin{array}{cc} \text{KGR} & \boxed{1} \end{array} \right] \end{array} \right] \sqcup \left[\begin{array}{cc} \text{SUBJ} & \left[\begin{array}{cc} \text{KGR} & \left[\begin{array}{cc} \text{PERS} & 3 \end{array} \right] \end{array} \right] \end{array} \right] =$$

$$\left[\begin{array}{cc} \text{KGR} & \boxed{1} \left[\begin{array}{cc} \text{NUM} & \text{SG} \\ \text{PERS} & 3 \end{array} \right] \\ \text{SUBJ} & \left[\begin{array}{cc} \text{KGR} & \boxed{1} \end{array} \right] \end{array} \right]$$

□

Die letzten beiden Beispiele zeigen das Verfahren bei der Unifikation koreferenter Werte. Die Unifikation zweier Strukturen mit koreferenten Werten führt im Vergleich zur Unifikation derselben Strukturen mit bloß kopierten Werten zu einem unterschiedlichen Ergebnis.

2.3.4 Generalisierung

Die Generalisierung (\sqcap) liefert diejenige Information, die zwei Merkmalsstrukturen gemeinsam haben:

$$(2.45) \quad \begin{bmatrix} \text{KAT} & \text{VERB} \\ \text{NUM} & \text{SG} \end{bmatrix} \sqcap \begin{bmatrix} \text{KAT} & \text{VERB} \\ \text{NUM} & \text{PL} \end{bmatrix} = \begin{bmatrix} \text{KAT} & \text{VERB} \end{bmatrix}$$

Ohne Verwendung disjunkter Werte führt die Generalisierung nicht zum gewünschten Ergebnis:

$$(2.46) \quad \begin{bmatrix} \text{KAT} & \text{VERB} \\ \text{KGR} & \begin{bmatrix} \text{NUM} & \text{SG} \\ \text{PERS} & 2 \end{bmatrix} \end{bmatrix} \sqcap \begin{bmatrix} \text{KAT} & \text{VERB} \\ \text{KGR} & \begin{bmatrix} \text{NUM} & \text{SG} \\ \text{PERS} & 1 \end{bmatrix} \end{bmatrix} \\ = \begin{bmatrix} \text{KAT} & \text{VERB} \\ \text{KGR} & \begin{bmatrix} \text{NUM} & \text{SG} \end{bmatrix} \end{bmatrix}$$

Hierbei geht zuviel Information verloren, was zur Folge hat, dass die Generalisierung ohne disjunkte Werte nicht dem Distributivitätsgesetz gehorcht. Das Ergebnis der Generalisierung bei Verwendung disjunkter Werte lautet:

$$(2.47) \quad \begin{bmatrix} \text{KAT} & \text{VERB} \\ \text{KGR} & \begin{bmatrix} \text{NUM} & \text{SG} \\ \text{PERS} & 1 \vee 2 \end{bmatrix} \end{bmatrix}$$

Definition 2.3.5

S_3 heißt **Generalisierung** ($S_3 = S_1 \sqcap S_2$) von S_1 und S_2 genau dann, wenn S_3 S_1 und S_2 subsumiert, und S_3 von allen anderen Merkmalsstrukturen subsumiert wird, die ebenfalls S_1 und S_2 subsumieren. \square

Analog zur Unifikation bezeichnet man das Ergebnis einer Generalisierung zweier Elemente einer geordneten Menge als ihr **Supremum**. Durch die leere Merkmalsstruktur, analog zu \perp (Bottom) auch als \top (Top) bezeichnet, am oberen Ende der Subsumptionshierarchie ist die Existenz eines Supremums für je zwei beliebige Merkmalsstrukturen sichergestellt.

Eigenschaften von Unifikation und Generalisierung

Wie schon gesagt bildet die Menge der Merkmalsstrukturen zusammen mit der Subsumptionsrelation eine geordnete Menge. Existiert in einer geordneten Menge für je zwei Elemente stets ein Supremum und ein Infimum, so nennt man dies einen **Verband** (engl. *lattice*).

Ein Verband besitzt bzgl. der beiden Operationen \sqcup und \sqcap die folgenden Eigenschaften:

1. a. $a \sqcup b = b \sqcup a$
b. $a \sqcap b = b \sqcap a$ (Kommutativgesetz)
2. a. $(a \sqcup b) \sqcup c = a \sqcup (b \sqcup c)$
b. $(a \sqcap b) \sqcap c = a \sqcap (b \sqcap c)$ (Assoziativgesetz)
3. a. $a \sqcup (a \sqcap b) = a$
b. $a \sqcap (a \sqcup b) = a$ (Absorptionsgesetz)
4. a. $a \sqcup a = a$
b. $a \sqcap a = a$ (Idempotenz)
5. Die Distributivität gilt nur, wenn disjunkte Werte in Merkmalsstrukturen erlaubt sind:

$$a \sqcap (b \sqcup c) = (a \sqcap b) \sqcup (a \sqcap c)$$

$$a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c)$$
6. a. $a \sqcup \perp = \perp, a \sqcup \top = a$
b. $a \sqcap \perp = a, a \sqcap \top = \top$ (Verhalten von Top und Bottom)

Aufgrund dieser Eigenschaften sind Unifikation und Generalisierung **monotonie** Operationen. Das bedeutet, sie können nur in einer „Richtung“ verlaufen. Ein bei einer Generalisierung weggefallenes Merkmal kann im Verlauf weiterer Generalisierungen derselben Merkmalsstruktur nicht wieder hinzukommen. Entsprechend kann bei der Unifikation ein einmal in die Struktur aufgenommenes Merkmal nicht wieder wegfallen; es sei denn, die Unifikation schlägt fehl und das Ergebnis ist \perp . Durch Unifikation kann Information nur hinzugefügt werden. Deswegen kann man partielle Beschreibungen (siehe 2.3.2) linguistischer Objekte herstellen und warten, dass sich im Verlauf der Unifikation die Lücken auffüllen. Außerdem spielt die Reihenfolge, in der man mehrere Merkmalsstrukturen unifiziert, keine Rolle, was den Entwurf eines Algorithmus erleichtert.

Die Eigenschaften von Verbänden und anderen in der Computerlinguistik verwendeten Begriffen der Algebra werden ausführlich in Partee, ter Meulen und Wall (1990) beschrieben.

Unifikation von Merkmalsstrukturen mit disjunkten Werten

Anhand eines Beispiels soll gezeigt werden, wie die Unifikation disjunktiver Merkmalsstrukturen abläuft. Es gilt die Regel: $(S_1 \vee S_2) \sqcup S_3 = (S_1 \sqcup S_3) \vee (S_2 \sqcup S_3)$. Will man diese Regel anwenden auf:

$$(2.48) \quad \left[\begin{array}{cc} \text{KAT} & \text{ART} \\ \text{KGR} & \left[\begin{array}{cc} \text{NUM} & \text{SG} \\ \text{GEN} & \text{MASK} \\ \text{KAS} & \text{AKK} \end{array} \right] \end{array} \right] \vee \left[\begin{array}{cc} \text{NUM} & \text{PL} \\ \text{KAS} & \text{DAT} \end{array} \right] \sqcup \left[\begin{array}{cc} \text{KGR} & \left[\begin{array}{cc} \text{GEN} & \text{FEM} \end{array} \right] \end{array} \right]$$

so muss zuerst die Disjunktion im Inneren der ersten Merkmalsstruktur aufgelöst werden. Die Disjunktion von Werten eines Merkmals innerhalb einer Merkmalsstruktur ist nur eine abkürzende Schreibweise, die wie im folgenden Beispiel in eine Disjunktion von Merkmalsstrukturen umgewandelt werden kann:

$$(2.49) \quad \left[\begin{array}{cc} \text{KAS} & \text{NOM} \\ \text{GEN} & \text{FEM} \vee \text{MASK} \end{array} \right] = \left[\begin{array}{cc} \text{KAS} & \text{NOM} \\ \text{GEN} & \text{FEM} \end{array} \right] \vee \left[\begin{array}{cc} \text{KAS} & \text{NOM} \\ \text{GEN} & \text{MASK} \end{array} \right]$$

Verfährt man ebenso mit 2.48 erhält man:

$$(2.50) \quad \left(\left[\begin{array}{cc} \text{KAT} & \text{ART} \\ \text{KGR} & \left[\begin{array}{cc} \text{NUM} & \text{SG} \\ \text{GEN} & \text{MASK} \\ \text{KAS} & \text{AKK} \end{array} \right] \end{array} \right] \vee \left[\begin{array}{cc} \text{KAT} & \text{ART} \\ \text{KGR} & \left[\begin{array}{cc} \text{NUM} & \text{PL} \\ \text{KAS} & \text{DAT} \end{array} \right] \end{array} \right] \right) \sqcup \left[\begin{array}{cc} \text{KGR} & \left[\begin{array}{cc} \text{GEN} & \text{FEM} \end{array} \right] \end{array} \right]$$

Nun kann man die obige Regel anwenden. Die Ergebnis-Struktur lautet:

$$(2.51) \quad \left[\begin{array}{cc} \text{KAT} & \text{ART} \\ \text{KGR} & \left[\begin{array}{cc} \text{NUM} & \text{PL} \\ \text{GEN} & \text{FEM} \\ \text{KAS} & \text{DAT} \end{array} \right] \end{array} \right]$$

Allgemein gilt, dass zwei Disjunktionen $S_1 = a_1 \vee a_2 \vee \dots$ und $S_2 = b_1 \vee b_2 \vee \dots$ unifiziert werden, indem zuerst alle Elemente von S_1 mit allen Elementen von S_2 unifiziert werden und dann die Disjunktion aus den so entstandenen Elementen gebildet wird.

2.3.5 Typisierte Merkmalsstrukturen

Merkmalssstrukturen dienen der Repräsentation linguistischer Objekte. Im Verlauf einer Unifikation kann z. B. eine Merkmalsstruktur, die ein finites Verb repräsentieren soll, ein Merkmal KASUS erhalten. Aber ebenso könnte einem Merkmal wie PERSON der Wert AKKUSATIV zugewiesen werden. Da so etwas nicht erwünscht ist, liegt es nahe, die erlaubten Merkmale und deren Werte einzuschränken. Linguistische Objekte lassen sich aufgrund gemeinsamer Merkmale

in Kategorien aufteilen. Diese Aufteilung kann mit **Typen** nachgebildet werden. Bei typisierten Merkmalsstrukturen lässt sich auch unterscheiden, ob ein Merkmal bloß nicht angegeben (unterspezifiziert), oder für diese Art Objekt gar nicht vorgesehen ist.

Welche Merkmale einem Typ zukommen, und welche Werte diese Merkmale annehmen dürfen, wird über eine **Angemessenheitsfunktion** app (von engl. *appropriateness*) festgelegt. Diese Funktion liefert für jedes Merkmal eines Typs die Menge der erlaubten Werte oder \uparrow (undefiniert), falls das Merkmal für diesen Typ nicht vorgesehen ist.

Als Beispiel soll ein Typ namens *kongruenz* definiert werden. Strukturen dieses Typs sollen die drei Merkmale NUM (Numerus), GEN (Genus) und KAS (Kasus) besitzen. Als Werte für das Merkmal NUM sollen SG und PL erlaubt sein:

$$app(kongruenz, \text{NUM}) = \{\text{SG}, \text{PL}\}$$

Obige Funktion besagt, dass der Typ *kongruenz* ein Merkmal NUM besitzt, das die angegebenen Werte (und sonst keine) annehmen darf. Entsprechend verfährt man für die beiden anderen Merkmale:

$$app(kongruenz, \text{GEN}) = \{\text{MASK}, \text{FEM}, \text{NEUT}\}$$

$$app(kongruenz, \text{KAS}) = \{\text{NOM}, \text{GEN}, \text{DAT}, \text{AKK}\}$$

Damit ist ein Typ *kongruenz* festgelegt bei dem jederzeit erkennbar ist, ob er unterspezifiziert ist, d.h. ob durch Unifikation fehlende Merkmale hinzukommen können oder nicht. In Matrixnotation gibt man den Typ einer Merkmalsstruktur in kursiven Kleinbuchstaben an:

$$(2.52) \quad \begin{bmatrix} \text{NUM} & sg \\ \text{GEN} & neut \\ \text{KAS} & dat \end{bmatrix}_{kongruenz}$$

Wie in untypisierten Strukturen können die Werte in typisierten Strukturen atomar oder komplex sein. Bei den Werten der Merkmale in der obigen Definition von *kongruenz* handelt es sich ausschließlich um **atomare Typen** (z. B. *sg*, *pl*), die selbst keine innere Struktur besitzen. Die atomaren Werte einfacher Merkmalsstrukturen werden in typisierten Strukturen durch atomare *Typen* ersetzt (daher sind die Werte in Struktur 2.52 kursiv gesetzt.).

Bei dem soeben definierten Typ *kongruenz* dagegen handelt es sich um einen **komplexen Typ**, der eine innere Struktur besitzt. Dieser Typ kann als komplexer Wert in einer Merkmalsstruktur eines entsprechend definierten Typs dienen. Dies könnte so aussehen:

$$app(dekliniert, \text{KAT}) = \{\text{pronomen}, \text{adjektiv}, \text{artikel}\}$$

$$app(dekliniert, \text{KGR}) = \{kongruenz\}$$

Hier werden für einen Typ *dekliniert* zwei Merkmale festgelegt. Ein Merkmal KAT, das die atomaren Typen *pronomen*, *adjektiv* und *artikel* als Werte anneh-

men kann, und ein weiteres Merkmal KGR, das den oben definierten komplexen Typ *kongruenz* als Wert erhalten darf. Eine Merkmalsstruktur des Typs *dekliniert* könnte z. B. so aussehen:

(2.53)	$\begin{array}{c} \left[\begin{array}{cc} \text{KAT} & \text{pronomene} \\ \text{KGR} & \end{array} \right] \\ \text{dekliniert} \quad \left[\begin{array}{cc} \text{kongruenz} & \left[\begin{array}{cc} \text{NUM} & \text{sg} \\ \text{GEN} & \text{neut} \\ \text{KAS} & \text{dat} \end{array} \right] \end{array} \right] \end{array}$
--------	---

Abbildung 2.15 zeigt eine typisierte Merkmalsstruktur in Graphendarstellung. Im Gegensatz zu untypisierten Strukturen, bei denen nur die Endknoten markiert sind, sind in typisierten Strukturen auch die inneren Knoten markiert. Jeder innere Knoten entspricht einem komplexen Typ, jeder Endknoten einem atomaren Typ.

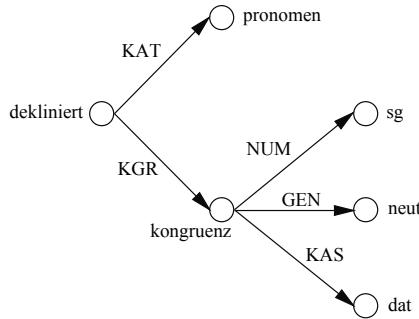


Abbildung 2.15: Eine typisierte Merkmalsstruktur

Vererbungshierarchien

Im Vergleich zu einer kontextfreien Grammatik mit atomaren Kategorien bestehen die Lexikoneinträge sogenannter Unifikationsgrammatiken (siehe hierzu den Syntax-Beitrag 3.5) aus sehr umfangreichen Merkmalsstrukturen. Während eine herkömmliche kontextfreie Grammatik im Lexikon lediglich die Eigenschaften der Wörter, die systematisch nicht zu erschließen sind, auflistet, befindet sich der Großteil des sprachlichen Wissens bei Unifikationsgrammatiken im Lexikon. Man spricht daher auch von lexikalisierten Grammatiken. Die hohe Komplexität und Redundanz erfordert eine durchdachte Organisation des Lexikons. Auch hierfür bieten Typen eine Lösung: Ihre Anordnung in einer **Vererbungshierarchie**. Diese Vererbungshierarchie ist zunächst nichts anderes als eine Subsumptionshierarchie aus Typen. Wir müssen also zuerst eine Subsumptionsrelation zwischen Typen definieren. Dies geschieht analog zur Subsumption zwischen Merkmalsstrukturen. Es gibt allgemeinere und speziellere Typen, beim spezielleren Typ muss mehr Information angegeben werden als beim allgemeineren.

Definition 2.3.6

Ein Typ t_1 (**Supertyp**) subsumiert einen Typ t_2 (**Subtyp**), wenn t_2 mindestens die Merkmale von t_1 verlangt. Dies muss rekursiv auch für komplexe Werte gelten. □

Wir erhalten eine Hierarchie aus Typen. Die Typen geben an, welche aus der Menge aller möglichen Merkmalsstrukturen ein linguistisches Objekt repräsentieren und welche Merkmalskombinationen keiner sprachlichen Einheit entsprechen, also bezüglich der Theorie nicht wohlgeformt sind. Abbildung 2.16 zeigt, wie ein Ausschnitt aus einer solchen Hierarchie aussehen könnte.

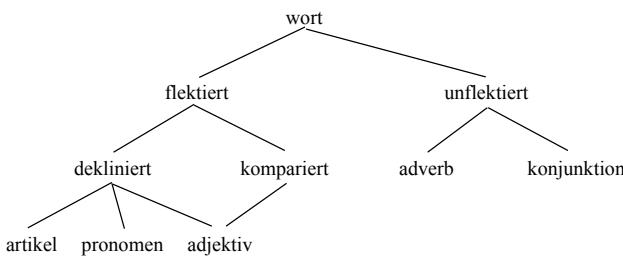


Abbildung 2.16: Eine Typhierarchie

Über eine entsprechende Angemessenheitsfunktion könnte z. B. festgelegt sein, dass der Typ *flektiert* ein Merkmal KGR (Kongruenz) verlangt, das von *unflektiert* nicht verlangt wird. Die Subsumptionsrelation besagt, dass Subtypen mindestens die Merkmale ihrer Supertypen besitzen. Daher verfügen auch alle Typen unterhalb von *flektiert* über das Merkmal KGR. Man sagt, Subtypen **erben** die Merkmale ihrer Supertypen. Beim Typ *dekliniert* wird angegeben, dass KGR als Werte die Merkmale KAS (Kasus), GEN (Genus) und NUM (Numerus) besitzt, so dass alle Subtypen von *dekliniert* diese erben. Der Typ *pronomem* beispielsweise erbt diese drei Merkmale von *dekliniert*, und fügt diesen ererbten Merkmalen seinerseits noch ein Merkmal PERS (Person) hinzu.

Durch Vererbung kann die Redundanz von Lexikoneinträgen enorm reduziert werden. Man „hängt“ ein Wort in der richtigen Stelle der Hierarchie ein, und es erbt alle Eigenschaften der Supertypen. Es muss nur noch idiosynkratische Information beim einzelnen Lexikoneintrag angegeben werden, etwa dass durch die Zeichenfolge *sie* ein Pronomen bezeichnet wird.

Wie aus Abbildung 2.16 hervorgeht, erbt der Typ *adjektiv* gleichzeitig die Merkmale von *dekliniert* und *kompariert*. Ist bei *kompariert* ein Merkmal KOMP (für Komparation) spezifiziert, dann besitzen Merkmalsstrukturen des Typs *adjektiv* die Kongruenz-Merkmale KAS, GEN und NUM von *dekliniert* und zusätzlich das bei *kompariert* angegebene KGR-Merkmal KOMP. Dies nennt man **Mehrzahlvererbung**. Dabei werden die geerbten Merkmale unifiziert.

Aus Abbildung 2.16 ist ersichtlich, dass eine Vererbungshierarchie, in der Mehrfachvererbung erlaubt ist, nicht mehr als Baum dargestellt werden kann

(der Knoten *adjektiv* besitzt zwei Mütter). Es handelt sich bei Abbildung 2.16 um einen allgemeinen Graphen (siehe 2.3.1).

Linguistisch existieren zu formulierten Regeln häufig entsprechende Ausnahmen, die sogenannten Subregularitäten. Ausnahmen können in einer Vererbungshierarchie sehr einfach modelliert werden: Ein beim Typ selbst angegebenes Merkmal (*default*) überschreibt ein ererbtes Merkmal. Dies führt allerdings dazu, dass die Vererbung nicht mehr monoton verläuft.

Man nennt eine typisierte Merkmalsstruktur **wohltypisiert**, wenn sie nicht mehr Merkmale hat, als ihr Typ vorschreibt und die Werte mindestens so allgemein sind wie vorgeschrieben. Eine Merkmalsstruktur darf also weniger Merkmale haben als ihr Typ verlangt (Unterspezifikation).

Eine typisierte Merkmalsstruktur ist **vollständig**, wenn alle Merkmale, die ihr Typ vorschreibt, auch vorhanden sind. Anders ausgedrückt: wenn eine wohlgeformte Merkmalsstruktur nicht unterspezifiziert ist, so ist sie vollständig.

Unifikation typisierter Merkmalsstrukturen

Für typisierte Merkmalsstrukturen ist analog zu den untypisierten Merkmalsstrukturen die Subsumption und die Unifikation definiert.

Definition 2.3.7

Der Typ t_3 ist das Ergebnis der **Unifikation der Typen** t_1 und t_2 genau dann, wenn t_3 von t_1 und t_2 subsumiert wird und t_3 alle anderen Typen subsumiert, die ebenfalls von t_1 und t_2 subsumiert werden. \square

Das Ergebnis der Unifikation von S_1 vom Typ t_1 mit S_2 vom Typ t_2 ist $S_3 = S_1 \sqcup S_2$, sofern der Typ von S_3 , t_3 , das Ergebnis der Unifikation von t_1 und t_2 ist.

Bei der Unifikation typisierter Merkmalsstrukturen werden also zusätzlich zu den einzelnen Merkmals-Wert-Paaren auch die Typen unifiziert.

Negation

Bei der Definition von Typen mit der Angemessenheitsfunktion wurde für einige Merkmale die Menge der zugelassenen Werte durch Aufzählung bestimmt. Beispielsweise ist durch

$$app(kongruenz, GEN) = \{\text{MASK}, \text{FEM}, \text{NEUT}\}$$

für den Typen *kongruenz* die Menge der Werte beim Merkmal GEN genau festgelegt. Daher ist es dasselbe zu sagen, dass GEN entweder die Werte MASK oder FEM besitzt, wie zu sagen, dass GEN nicht den Wert NEUT hat. Wir können einen Negationsoperator \neg einführen (siehe Unterkapitel 2.1):

$$mask \vee fem = \neg neut$$

Implikation

Die Implikation → wird – wie im Logik-Beitrag 2.1 dargelegt – mit Hilfe von Negation und Disjunktion wie folgt definiert:

$$S_1 \rightarrow S_2 \Leftrightarrow \neg S_1 \vee S_2$$

Implikation wird in manchen Formalismen verwendet, um allgemeine Prinzipien auszudrücken, beispielsweise, dass ein finites Verb ein Merkmal TEMPUS besitzen muss.

2.3.6 Literaturhinweise

Die Unifikation wurde zuerst in der Functional Unification Grammar (Kay 1979) eingesetzt. Ein anderes bekanntes System ist PATR-II (Shieber 1986). Weitere Formalismen sind die Definite Clause Grammar (DCG; Pereira und Warren 1980), STUF (Uszkoreit 1987), CUF (Dörre und Dorna 1993), CFS (Böttcher 1996), TDL (Krieger und Schäfer 1994), ALE (Carpenter und Penn 1994) und ConTroll (Götz et al. 1997).

Es existieren diverse Erweiterungen des Unifikationsformalismus, z. B. können Merkmalswerte aus Listen oder Mengen von Strukturen bestehen. Am weitesten entwickelt ist in dieser Hinsicht die Head-driven Phrase Structure Grammar (HPSG), die in Pollard und Sag (1987), Pollard und Sag (1994) und Müller (2008) dargestellt ist.

Für die Systeme LKB (Copestake 2002) und PET (Callmeier 2001) ist mit GG (<http://gg.opendfki.de/>) eine HPSG des Deutschen frei im Web verfügbar.

Die Unifikation ist auch fester Bestandteil der Programmiersprache Prolog. Der DCG-Formalismus ist ebenfalls in viele Prologsysteme bereits eingebaut.

Sag et al. (2003) führt in die Beschreibung syntaktischer Phänomene mit Merkmalsstrukturen ein. Weiterführende Lektüre über die formalen Eigenschaften von Merkmalsstrukturen bieten Johnson (1988), Carpenter (1992), King (1994) und Richter (2004). Ein einfacher Unifikationsalgorithmus wird in Jurafsky und Martin (2009) angegeben.

2.4 Statistische Grundlagen

Stefan Evert, Bernhard Frötschl und Wolf Lindstrot

Lange Zeit war die symbolische Verarbeitung das primäre Paradigma der Computerlinguistik, denn natürliche Sprachen sind symbolbasierte Kommunikationsmittel. Die Phoneme und Morpheme einer Sprache sind elementare Symbole. Die Phonologie, die Morphologie und die Syntax klären, wie diese Elementarsymbole zu komplexen Symbolen kombiniert werden. Die Semantik schließlich befasst sich mit der Interpretation dieser Symbole. Anfang der 80er Jahre jedoch erwachte im Zusammenhang mit der Auswertung großer maschinenlesbarer Korpora das Interesse an stochastischen Verfahren wieder, nachdem erste Ansätze in den 50er Jahren – motiviert von Erfolgen in der Informationstheorie – an der unzureichenden Leistungsfähigkeit der damaligen Computertechnik gescheitert waren. Heute bildet die Wahrscheinlichkeitstheorie eine unabdingbare Grundlage für viele computerlinguistische Anwendungen.

Die Wahrscheinlichkeitstheorie spielt u.a. eine Rolle bei der Entwicklung und Verwendung probabilistischer Grammatiken, bei der automatischen Annotierung von Texten mit Wortklassen und anderen linguistischen Merkmalen auf Wortebene (sog. Tagging), bei der maschinellen Übersetzung, der Textklassifikation, dem Textmining, der Spracherkennung, der Computerlexikographie, der Analyse und Bereinigung von Webseiten, sowie bei der Evaluation sprachverarbeitender Systeme. Man vergleiche hierzu die entsprechenden Beiträge in diesem Buch.

Zu den erfolgreichsten wahrscheinlichkeitsbasierten Ansätzen in der Computerlinguistik gehören Hidden-Markov-Modelle, die insbesondere für Tagging-Aufgaben (siehe Unterkapitel 3.4) und in der Spracherkennung (siehe Unterkapitel 5.4) eingesetzt werden. Erst seit Ende der 90er Jahre werden sie durch komplexere statistische Methoden und maschinelle Lernverfahren abgelöst. Angesichts ihrer Effizienz und einfachen Implementierung zählen Hidden-Markov-Modelle jedoch auch weiterhin in vielen Anwendungsgebieten zu den Methoden der ersten Wahl. Aus diesem Grund werden sie im vorliegenden Unterkapitel besonders ausführlich behandelt.

2.4.1 Wahrscheinlichkeitstheoretische Grundlagen

Die Wahrscheinlichkeitstheorie stellt mathematische Modelle bereit, um Zufallsexperimente zu untersuchen. Zufallsexperimente sind Vorgänge, die mit einem bestimmten Ergebnis enden, dessen Gestalt aber aufgrund mangelnder Kontrolle der Einflussfaktoren oder mangelnder Information vorher unbekannt ist. Das klassische Beispiel ist das Würfeln. Die Menge möglicher Ergebnisse ist bekannt, doch weiß niemand vor einem Wurf, welche der sechs Augenzahlen fallen wird. Die übliche Verwendung des Würfels im Spiel beruht gerade auf der Erwartung, dass er „gerecht“ ist, also kein mögliches Ergebnis bevorzugt. Wir finden hier also einen intuitiven Begriff von Wahrscheinlichkeit: Sie stellt eine Art Tendenz des Experiments zu bestimmten Ergebnissen dar, ohne dass sich das tatsächlich

eintretende Ergebnis mit Sicherheit vorhersagen ließe. Beim Würfeln geht man davon aus, dass alle möglichen Ergebnisse die gleiche Wahrscheinlichkeit haben. Bei anderen Experimenten – und man kann fast jeden Vorgang im täglichen Leben als Zufallsexperiment betrachten – ist dies ganz anders. Aufgrund von Einschätzungen, dass die Wahrscheinlichkeit eines bestimmten Ergebnisses eine gewisse Höhe hat, treffen Menschen viele Entscheidungen. Sie schätzen Wahrscheinlichkeiten ein, dass es regnen wird, dass ein Freund pünktlich kommt oder dass sich die Autos auf der A10 am Montagmorgen stauen werden.

Eine weitere Seite unseres intuitiven Wahrscheinlichkeitsbegriffs stellt die Häufigkeitsinterpretation von Wahrscheinlichkeiten dar. Für wiederholbare Experimente erwarten wir, dass ein Ergebnis mit einer hohen Wahrscheinlichkeit auch häufig auftreten wird. Insbesondere sollte bei einer großen Anzahl von Wiederholungen die Häufigkeit eines Ergebnisses proportional zu seiner Wahrscheinlichkeit sein. Aufgrund dieser Intuition prüft man die Korrektheit eines Würfels durch eine längere Reihe von Würfeln. Von einem „gerechten“ Würfel wird erwartet, dass jede Augenzahl ungefähr in einem Sechstel der Fälle auftritt. Gleichzeitig ist es aber nicht völlig ausgeschlossen, dass auch in einer langen Reihe von Würfeln nur Einsen fallen.

Wie soll also ein mathematisches Modell der Wahrscheinlichkeit aussehen? Ein Maß für Wahrscheinlichkeiten sollte ein Spektrum zwischen „unmöglich“ und „vollkommen sicher“ abdecken. Nach der Häufigkeitsinterpretation entspricht die Wahrscheinlichkeit eines Ergebnisses dem Anteil „günstiger“ Fälle in einer Folge von Versuchen (d.h. dem Anteil von Fällen, in denen das betreffende Ergebnis eintritt). Deshalb liegt es nahe, Wahrscheinlichkeitswerte zwischen 0 („unmöglich“) und 1 („absolut sicher“) zuzulassen, die als Prozentangaben interpretiert werden können: „Mit 90-prozentiger Wahrscheinlichkeit (Wert 0.9) kommt er wieder zu spät.“

In diesem ersten Abschnitt des vorliegenden Unterkapitels soll das elementare mathematische Werkzeug vorgestellt werden, das nötig ist, um Probleme der Computerlinguistik wahrscheinlichkeitstheoretisch zu modellieren. Als Anwendungsbeispiel hierfür dient die maschinelle Spracherkennung, bei der wahrscheinlichkeitstheoretische Methoden in der Form statistischer Sprachmodelle eine wichtige Rolle spielen. Man erinnere sich aber stets daran, dass die Wahrscheinlichkeitstheorie ein universelles Modellierungswerkzeug ist. In Abschnitt 2.4.3 wurde daher bewusst mit sog. Tagging-Verfahren zur automatischen linguistischen Annotierung ein anderes Anwendungsbeispiel gewählt.

Die zentrale Aufgabe der Spracherkennung besteht darin, zu einem gegebenen Sprachsignal den Satz zu finden, dessen Äußerung das Signal produziert hat. Oder anders ausgedrückt: Gesucht ist der Satz, für den die höchste Wahrscheinlichkeit besteht, das Signal verursacht zu haben. Um dieses Problem mathematisch formulieren zu können, benötigt man den Begriff des **diskreten Wahrscheinlichkeitsraums** und den der **bedingten Wahrscheinlichkeit**, die im Folgenden eingeführt werden. Um es zu lösen, kann man sich mathematischer Sätze wie der **Bayes-Formel** bedienen, die in Abschnitt 2.4.1 bewiesen wird.

Diskrete Wahrscheinlichkeitsräume

Ziel dieses Abschnitts ist die Einführung des mathematischen Begriffs „Wahrscheinlichkeitsraum“ (hier beschränkt auf diskrete Wahrscheinlichkeitsräume) als Formalisierung des im letzten Abschnitt vorgestellten intuitiven Wahrscheinlichkeitsbegriffs.

Ereignisse und Mengenlehre: Ein Zufallsexperiment wird formal über die Menge der möglichen **Ergebnisse** definiert, die im Allgemeinen mit Ω bezeichnet wird.

Es lassen sich leicht Beispiele für endliche, unendlich abzählbare und überabzählbare Ergebnismengen finden: Das Werfen eines Würfels ist ein Zufallsexperiment mit der endlichen Ergebnismenge $\Omega = \{1, 2, 3, 4, 5, 6\}$. Ein Computerprogramm, das auf eine natürlichsprachliche Tastatureingabe wartet, befindet sich in einem Experiment mit abzählbar unendlicher Ergebnismenge $\Omega = \Sigma^*$, wobei Σ alle Zeichen der Tastatur umfasst. Die damit gemachte Annahme, dass die Eingabe beliebig lang sein darf, ist in praktischen Anwendungen zumindest fragwürdig. Sie darf jedoch als akzeptable Näherung betrachtet werden, sofern alle unrealistisch langen Eingabewörter einen sehr kleinen Wahrscheinlichkeitswert haben. Ein Computerprogramm, das auf eine gesprochene Eingabe wartet, müsste mit einer überabzählbar großen Ergebnismenge in Form von beliebig genauen Messwerten fertig werden, wenn da nicht technische und physikalische Beschränkungen der Genauigkeit wären.

Soll an die Ergebnismenge nun ein Wahrscheinlichkeitsmaß angelegt werden, so stellt man schnell fest, dass man nicht immer an der Wahrscheinlichkeit einzelner Ergebnisse (wie „Augenzahl 6“ oder „Eingabe ist der Satz *Jetzt aber schleunigst Neustart*“) interessiert ist, sondern an Merkmalen, die auf mehrere Ergebnisse zutreffen, und damit an Mengen von Ergebnissen. Beispiele wären die Wahrscheinlichkeiten der Ereignisse „eine gerade Augenzahl wird geworfen“ und „das dritte Wort der Eingabe lautet *schleunigt*.“ Es ist also zu unterscheiden zwischen **Ergebnissen** und **Ereignissen**. Ereignisse bilden Teilmengen $A \subseteq \Omega$ der Menge aller möglichen Ergebnisse. Wahrscheinlichkeiten werden den Ereignissen zugewiesen. Das Wahrscheinlichkeitsmaß ist also ein Maß auf der Potenzmenge $\wp(\Omega)$ der Ergebnismenge.

Ist man doch einmal an der Wahrscheinlichkeit eines einzelnen Ergebnisses $\omega \in \Omega$ interessiert, so wählt man als Ereignis die Menge $\{\omega\}$, die nur dieses Ergebnis enthält. Solche einelementigen Mengen nennt man **atomare Ereignisse**. Hat man die Ergebnismenge Ω definiert, so zeigt sich, dass die Mittel der Mengenlehre hilfreiche Werkzeuge bilden, um sprachliche Verknüpfungen in die Ereignismenge $\wp(\Omega)$ zu übertragen.

Dies sei am Würfelexperiment demonstriert. Ereignis A bezeichne das Werfen einer geraden Augenzahl und B das Werfen einer Augenzahl, die größer als 4 ist. Dann bildet sich das Ereignis „eine Augenzahl, die gerade und größer als 4 ist“, also die sprachliche „und“-Verknüpfung, durch den Mengenschnitt $A \cap B = \{2, 4, 6\} \cap \{5, 6\} = \{6\}$. Gleches gilt für die Mengenvereinigung $A \cup B$ („Ereignis A oder Ereignis B “) und die Mengendifferenz $A \setminus B$ („Ereignis A , aber nicht

Ereignis B^c). Das Mengenkomplement $\overline{A} = \Omega \setminus A$ („Ereignis A findet nicht statt“) entspricht im Beispiel dem Ereignis, dass eine ungerade Augenzahl geworfen wird.

Eine besondere Rolle kommt den Ereignissen zu, die durch die leere Menge \emptyset (**unmögliches Ereignis**) und durch die Gesamtmenge Ω (**sicheres Ereignis**) beschrieben werden. Gilt die Teilmengenrelation $A \subseteq B$ für Ereignisse A und B , so entspricht dies der sprachlichen Umschreibung „aus A folgt B .“ Im obigen Beispiel gilt etwa für das Ereignis $C = \{3, 4, 5, 6\}$ („Augenzahl größer als 2“) die Beziehung $B \subseteq C$: „ist die Augenzahl größer als 4, so ist sie auch größer als 2.“ Schließlich bedeutet $A \cap B = \emptyset$, dass es sich bei A und B um miteinander **unvereinbare Ereignisse** handelt.

Die letzten Absätze sollten gezeigt haben, dass die Mengenlehre eine Begrifflichkeit bereitstellt, die für eine Formalisierung des intuitiven Wahrscheinlichkeitsbegriffs sehr gut geeignet ist.

Definition des diskreten Wahrscheinlichkeitsraums: Ein Wahrscheinlichkeitsraum enthält alle notwendigen Informationen, um ein Zufallsexperiment wahrscheinlichkeitstheoretisch zu beschreiben. Dabei wird völlig von den konkreten Gegebenheiten, wie z. B. den physikalischen Eigenschaften des Würfels, abstrahiert.

Definition 2.4.1

Ein **diskreter Wahrscheinlichkeitsraum** ist ein Paar (Ω, P) . Dabei ist Ω eine nicht leere, abzählbare Menge. Die Elemente von Ω sind die möglichen Ergebnisse des Zufallsexperiments. $P : \wp(\Omega) \rightarrow \mathbb{R}$ ist ein **Wahrscheinlichkeitsmaß** auf der Potenzmenge von Ω . P weist jeder Menge von Ergebnissen, also jedem möglichen Ereignis, eine Wahrscheinlichkeit zu. Sie muss folgende Bedingungen erfüllen:

Axiom 1: $P(A) \geq 0$ für alle $A \in \wp(\Omega)$;

Axiom 2: $P(\Omega) = 1$;

Axiom 3: sind die Mengen $A_n \in \wp(\Omega)$ für $n \in \mathbb{N}$ paarweise disjunkt, so gilt:

$$P\left(\bigcup_{n \in \mathbb{N}} A_n\right) = \sum_{n \in \mathbb{N}} P(A_n).$$

Axiom 3 fordert: Die Wahrscheinlichkeit der Vereinigung einer Folge von abzählbar unendlich vielen Ereignissen muss gleich der Summe der Einzelwahrscheinlichkeiten sein, wenn die Ereignisse paarweise disjunkt (also miteinander unvereinbar) sind. Die Bezeichnung P für das Wahrscheinlichkeitsmaß geht auf das englische Wort *probability* zurück. \square

Diese Definition des Wahrscheinlichkeitsraums ist in zweierlei Hinsicht eingeschränkt gegenüber der allgemeinen Definition in gängigen Lehrbüchern der

Wahrscheinlichkeitsrechnung. Mit der Beschränkung auf diskrete Wahrscheinlichkeitsräume mit höchstens abzählbar unendlichem Ω werden überabzählbare Ereignismengen ausgeklammert. Um diese zu untersuchen, werden mathematische Grundlagen benötigt, die hier nicht vorausgesetzt und aufgrund des beschränkten Platzes auch nicht bereitgestellt werden können. Die zweite Einschränkung betrifft den Definitionsbereich des Wahrscheinlichkeitsmaßes P . In Definition 2.4.1 wird gefordert, dass P auf der gesamten Potenzmenge von Ω definiert sei. Üblicherweise beschränkt man sich stattdessen auf eine Teilmenge von $\wp(\Omega)$. Da dies jedoch weitere Komplikationen mit sich bringt, wurde hier die stärkere Forderung gestellt, dass P allen Teilmengen von Ω eine Wahrscheinlichkeit zuweise.

Eigenschaften des Wahrscheinlichkeitsmaßes P : Die axiomatisch formulierten Anforderungen an das Wahrscheinlichkeitsmaß P sind sehr knapp ausgewichen. Im Folgenden soll gezeigt werden, dass nichtsdestoweniger einige Eigenschaften, die ein Wahrscheinlichkeitsmaß besitzen sollte, um unserem intuitiven Wahrscheinlichkeitsbegriff zu entsprechen, durch diese schlanke Definition garantiert werden. Damit sind folgende Eigenschaften gemeint:

1. Die Wahrscheinlichkeit für das unmögliche Ereignis sollte 0 betragen, also $P(\emptyset) = 0$.
2. Axiom 3 ist für eine unendliche Folge von Mengen formuliert. Für eine endliche Anzahl von Ereignissen sollte aber die gleiche Formel anwendbar sein. Insbesondere muss für beliebige unvereinbare Ereignisse $A, B \in \wp(\Omega)$ mit $A \cap B = \emptyset$ gelten: $P(A \cup B) = P(A) + P(B)$.
3. Das Wahrscheinlichkeitsmaß sollte garantieren: „Tertium non datur!“ Es muss gewährleistet sein, dass ein Ereignis entweder eintritt oder nicht eintritt, nichts Drittes ist möglich. Formal: $P(A) + P(\overline{A}) = 1$ für alle $A \in \wp(\Omega)$.
4. Impliziert ein Ereignis A ein Ereignis B , gilt also $A \subseteq B$, dann darf die Wahrscheinlichkeit von B nicht unter der von A liegen, also $P(A) \leq P(B)$. Genauer sollte gelten: $P(B \setminus A) = P(B) - P(A)$.
5. Allgemein muss gelten: Kein Ereignis kann eine Wahrscheinlichkeit über 1 (entsprechend dem sicheren Ereignis) haben. Dass alle Wahrscheinlichkeiten positiv oder gleich 0 sind, ist durch Axiom 1 sichergestellt.

Die ersten beiden Behauptungen lassen sich mithilfe von Axiom 3 beweisen. Für die erste setze man alle $A_n = \emptyset$, so dass $\emptyset = \bigcup_{n \in \mathbb{N}} A_n$. Da dann gilt, dass $P(\emptyset) = P(\bigcup_{n \in \mathbb{N}} A_n)$ und der Ausdruck auf der rechten Seite die Bedingungen des Axioms erfüllt, kann zu $P(\emptyset) = \sum_{n \in \mathbb{N}} P(\emptyset)$ vereinfacht werden. Für diese Gleichung gibt es nur eine Lösung $P(\emptyset) = 0$.

Um die zweite Behauptung zu beweisen, wähle man $A_1 = A, A_2 = B$ und $A_n = \emptyset$, für $n > 2$ ($n \in \mathbb{N}$), und gehe nach dem gleichen Schema vor. Dann erhält man $P(A \cup B) = P(A) + P(B) + \sum_{n \in \mathbb{N}} P(\emptyset)$. Da sich die unendliche

Summe zu 0 addiert, ist man am Ziel. Auf dem gleichen Weg lässt sich die Verallgemeinerung für beliebige endliche Folgen von Ereignissen zeigen.

Diese zweite Aussage hilft von rechts nach links gelesen beim Beweis der dritten: $P(A) + P(\overline{A}) = P(A \cup \overline{A}) = P(\Omega) = 1$.

Auch die vierte Behauptung kann man auf die zweite zurückführen: Wenn $A \subseteq B$, so kann B als $A \cup (B \setminus A)$ geschrieben werden. Dies ist die Vereinigung zweier disjunkter Mengen, also $P(B) = P(A) + P(B \setminus A)$. Durch Umstellung erhält man die gewünschte Gleichung. Insbesondere folgt $P(A) \leq P(B)$.

Die fünfte Behauptung ergibt sich direkt aus der zuletzt bewiesenen Aussage, wenn man von Axiom 2 ausgeht und sich vor Augen führt, dass für jedes Ereignis A gilt: $A \subseteq \Omega$.

Laplace-Räume: Ein Sonderfall des diskreten Wahrscheinlichkeitsraums ist der Laplace-Raum, in dem alle Ergebnisse die gleiche Wahrscheinlichkeit haben. In Laplace-Räumen gilt daher: Die Wahrscheinlichkeit eines Ereignisses ergibt sich aus seinem Anteil an der Gesamtheit der möglichen Ergebnisse.

Z.B. gilt diese Formel bei einfachen Würfelexperimenten mit einem gerechten Würfel: Die Wahrscheinlichkeit, eine 3 oder eine 4 zu würfeln, beträgt genau $\frac{1}{3}$, da zwei Augenzahlen genau ein Drittel der sechs möglichen Ergebnisse bilden. Doch schon für die Gesamtaugenzahl von zwei Würfeln gilt die Regel nicht mehr: Die Wahrscheinlichkeit, mit zwei Würfeln insgesamt eine 6 oder eine 7 zu würfeln, beträgt nicht $\frac{2}{12}$, sondern $\frac{11}{36}$ (denn 11 von 36 möglichen Paaren addieren sich zu 6 oder zu 7). Betrachtet man die zwölf möglichen Gesamtaugenzahlen als Ergebnisse, so bildet dieses Experiment keinen Laplace-Raum.

Definition 2.4.2

Ein **Laplace-Raum** ist ein diskreter Wahrscheinlichkeitsraum $\langle \Omega, P \rangle$ mit endlichem Ω und Gleichverteilung der Wahrscheinlichkeit P . Sei $N \in \mathbb{N}$ die Kardinalität von Ω , d.h. $N = |\Omega|$. Die **Gleichverteilung** auf Ω weist allen atomaren Ereignissen die gleiche Wahrscheinlichkeit zu: $P(\{\omega\}) = \frac{1}{N}$ für alle $\omega \in \Omega$. \square

In Laplace-Räumen gilt also:

$$P(A) = \sum_{n=1}^{|A|} \frac{1}{|\Omega|} = |A| \frac{1}{|\Omega|} = \frac{\text{Anzahl der günstigen Ergebnisse}}{\text{Anzahl der möglichen Ergebnisse}}.$$

Beispiel 2.4.1

Wie groß ist die Wahrscheinlichkeit $P(G)$, dass in einer Gruppe von 30 Personen mindestens zwei am gleichen Tag Geburtstag haben? Zur Vereinfachung ignorieren wir Schaltjahre sowie die saisonale Verteilung von Geburten und nehmen an, dass die Wahrscheinlichkeit, an einem bestimmten Tag im Jahr geboren zu sein, genau $\frac{1}{365}$ betrage.

Eine günstige Formalisierung des Experiments könnte folgendermaßen aussehen: Ein Ergebnis, also eine Verteilung der 30 Personen auf die 365 Tage des Jahres wird als Folge $(g_1, g_2, \dots, g_{30}) \in \{1, \dots, 365\}^{30}$ kodiert. Sie besagt, dass

Person i am g_i -ten Tag des Jahres Geburtstag hat. Da nach Annahme alle Folgen gleichwahrscheinlich sind, liegt ein Laplace-Raum vor. Die Anzahl der möglichen Ergebnisse lässt sich mithilfe einfacher Kombinatorik bestimmen: Es gibt 365^{30} verschiedene 30-elementige Folgen aus den Zahlen von 1 bis 365.

Schwieriger ist die Bestimmung der Anzahl günstiger Fälle, also der Folgen mit mindestens einer doppelt vorkommenden Zahl. Überlegt man sich jedoch, dass es einfacher ist, die Anzahl der ungünstigen Fälle zu bestimmen, kann man sich der „Tertium non datur“-Formel in der Form $P(G) = 1 - P(\bar{G})$ bedienen. Wie viele 30-elementige Folgen aus paarweise verschiedenen Zahlen aus dem Intervall $\{1 \dots 365\}$ gibt es? Die Antwort lautet:

$$\begin{aligned} |\bar{G}| &= 365 \times 364 \times \dots \times (365 - 29) \\ &= \frac{365 \times 364 \times \dots \times 2 \times 1}{(365 - 30) \times (365 - 31) \times \dots \times 2 \times 1} \\ &= \frac{365!}{(365 - 30)!}. \end{aligned}$$

Damit beträgt die Wahrscheinlichkeit $P(G) = 1 - \frac{365!}{365^{30}(365-30)!} \approx 1 - 0.29 = 0.71$. Dieser hohe Wert widerspricht der Intuition der meisten Menschen, weshalb man dieses Beispiel auch das „Geburtstagsparadoxon“ nennt. \square

Bedingte Wahrscheinlichkeiten

Wie schon einleitend bemerkt, spielt der Begriff der bedingten Wahrscheinlichkeit eine zentrale Rolle bei Anwendungen der Wahrscheinlichkeitstheorie in der Computerlinguistik. Mithilfe dieses Begriffes lässt sich mathematisch der Einfluss zusätzlich zur Verfügung stehender Information auf die Wahrscheinlichkeiten von Ereignissen modellieren. Ist bekannt, dass bei einem Würfelwurf eine Augenzahl gefallen ist, die höher als drei ist, verschieben sich die Erwartungen bezüglich aller Ereignisse deutlich. Zum einen fällt die Hälfte der möglichen Ergebnisse weg, erhält also eine Wahrscheinlichkeit von 0. Aber auch Ereignisse wie „eine gerade Augenzahl“ verändern ihre Wahrscheinlichkeit (von $\frac{1}{2}$ zu $\frac{2}{3}$). Mit der zusätzlichen Information ist dieses Ereignis wahrscheinlicher als ohne sie, da nun zwei von drei noch möglichen Ergebnissen gerade sind.

Definition 2.4.3

Sei $\langle \Omega, P \rangle$ ein diskreter Wahrscheinlichkeitsraum und $A \in \wp(\Omega)$ ein Ereignis mit $P(A) > 0$. Dann ist die durch

$$P(B|A) = \frac{P(B \cap A)}{P(A)}$$

definierte Abbildung $P(\cdot|A) : \wp(\Omega) \rightarrow \mathbb{R}$ das **durch A bedingte Wahrscheinlichkeitsmaß** auf $\wp(\Omega)$. $P(B|A)$ steht anschaulich für „die Wahrscheinlichkeit von B bedingt durch A“ oder auch „gegeben A.“ \square

Definition 2.4.3 enthält implizit die Behauptung, dass $P(\cdot|A)$ die Axiome eines Wahrscheinlichkeitsmaßes erfüllt. Dies ist jedoch keineswegs selbstverständlich.

Exemplarisch soll hier die Gültigkeit des dritten Axiomes für $P(\cdot|A)$ gezeigt werden. Die anderen beiden sind durch Einsetzen sehr leicht zu beweisen.

Axiom 3 für die bedingte Wahrscheinlichkeit fordert, dass

$$P\left(\bigcup_{n \in \mathbb{N}} B_n | A\right) = \sum_{n \in \mathbb{N}} P(B_n | A), \quad (2.54)$$

wenn die B_n eine Folge paarweise disjunkter Ereignisse sind. Setzt man in die linke Seite die Definition der bedingten Wahrscheinlichkeit ein, erhält man $\frac{P((\bigcup_{n \in \mathbb{N}} B_n) \cap A)}{P(A)}$. Das Distributivgesetz der Mengenlehre erlaubt, den Zähler in $P(\bigcup_{n \in \mathbb{N}} (B_n \cap A))$ umzuformen. Da hier disjunkte Mengen vereinigt werden und $P(\cdot)$ sicher Axiom 3 erfüllt, ist dessen Anwendung zulässig und verändert den Zähler zu $\frac{\sum_{n \in \mathbb{N}} P(B_n \cap A)}{P(A)} = \sum_{n \in \mathbb{N}} \frac{P(B_n \cap A)}{P(A)}$ (das Summenzeichen kann vor den Bruch geschrieben werden, da der Nenner $P(A)$ nicht von n abhängt). Nun muss nur noch Definition 2.4.3 ein zweites Mal angewendet werden, um die rechte Seite von Gleichung (2.54) zu erhalten.

Mithilfe der bedingten Wahrscheinlichkeit kann das in der Einleitung vorgestellte Problem der Spracherkennung formalisiert werden. Gestellt wurde die Frage: Welcher Satz w hat am wahrscheinlichsten ein gegebenes Sprachsignal x verursacht? Wahrscheinlichkeitstheoretisch formuliert lautet sie: Welches w maximiert den Wert $P(w|x)$? Wie die Werte $P(w|x)$ für alle w bei gegebenem x berechnet werden können, ist zunächst einmal unklar. Die im nächsten Abschnitt bewiesene Bayes-Formel wird dies ermöglichen.

Der letzte wahrscheinlichkeitstheoretische Begriff, der eingeführt werden soll, betrifft die **Unabhängigkeit von Ereignissen**. Im Allgemeinen kann zwischen $P(A|B)$ und $P(A)$ irgendeine der Beziehungen $<$, $=$ oder $>$ gelten. Gilt hier die Gleichheit, so heißen A und B unabhängig. Die folgende Definition drückt genau dies aus, zeigt jedoch deutlicher die Symmetrie des Begriffs und benötigt keine Voraussetzung der Art $P(B) > 0$.

Definition 2.4.4

Zwei Ereignisse A und B heißen **unabhängig**, falls $P(A \cap B) = P(A)P(B)$. \square

Beispiel 2.4.2

Im Würfelexperiment sind die Ereignisse A , dass eine gerade Zahl gewürfelt wird, und B , dass eine Zahl ≤ 2 gewürfelt wird, unabhängig. Denn

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(\{2\})}{P(\{1, 2\})} = 0.5 = P(A).$$

Für ein Gegenbeispiel wähle man \hat{B} als das Ereignis, dass genau die 2 gewürfelt wird. Dann ist

$$P(A|\hat{B}) = \frac{P(A \cap \hat{B})}{P(\hat{B})} = \frac{P(\{2\})}{P(\{2\})} = 1 \neq P(A).$$

A und \hat{B} sind nicht voneinander unabhängig. \square

Die Formel von Bayes

Untersucht werden soll folgendes Beispiel: Bernhard erhält einen Strafzettel wegen Fahrens mit überhöhter Geschwindigkeit. Wie hoch ist die Wahrscheinlichkeit, dass er tatsächlich zu schnell gefahren ist?

Der Erhalt des Strafzettels sei als Ereignis B und das zu schnelle Fahren als Ereignis A bezeichnet. Gesucht ist also $P(A|B)$. Die Bayes-Formel, die im Folgenden entwickelt werden wird, ermöglicht es, $P(A|B)$ zu berechnen, wenn folgende Informationen zur Verfügung stehen: das Ausmaß und die Zuverlässigkeit der Verkehrsüberwachung (wie viele RaserInnen kommen ungestraft davon, und wie viele Unschuldige werden zur Kasse gebeten?) und Bernhards Fahrverhalten (wie wahrscheinlich ist bei ihm eine Geschwindigkeitsübertretung?).

Als erster Schritt in diese Richtung soll Gleichung (2.56) entwickelt werden. Die Definition der bedingten Wahrscheinlichkeiten $P(A|B)$ und $P(B|A)$ kann umgeformt werden zu

$$P(A \cap B) = P(A|B)P(B) \text{ und } P(B \cap A) = P(B|A)P(A). \quad (2.55)$$

Da der Mengenschnitt kommutativ ist, ergibt sich $P(A|B)P(B) = P(B|A)P(A)$. Dann ist nur noch eine kleine Umformung nötig, um folgende Gleichung zu erhalten:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \quad (2.56)$$

Es sei angenommen, dass RaserInnen es infolge einer strengen Politik schwer haben und fast alle Geschwindigkeitsübertretungen registriert werden: $P(B|A) = 0.8$. Bernhard sei ein äußerst verantwortungsbewusster Fahrer: $P(A) = 0.01$. Dann bleibt als letzte Unbekannte auf der rechten Seite von Formel (2.56) der Wert von $P(B)$, also die allgemeine Wahrscheinlichkeit, dass Bernhard einen Strafzettel erhält (man beachte dass es nicht möglich ist, diesen Wert direkt zu bestimmen, falls Bernhard noch nie einen Strafzettel erhalten hat). Das Ziel soll sein, $P(B)$ und damit $P(A|B)$ mit Kenntnis nur einer weiteren Information berechnen zu können: Wie hoch ist das Risiko für Nicht-RaserInnen, Ziel einer Geldforderung zu werden? Formal ist das der Wert $P(B|\bar{A})$, der hier 0.01 betragen soll (d.h. 1% aller unschuldigen FahrerInnen erhalten einen Strafzettel).

Da auch $P(\bar{A}) = 1 - P(A) = 0.99$ bekannt ist, kann nun nach Gleichung (2.55) die Wahrscheinlichkeit $P(\bar{A} \cap B)$, dass Bernhard nicht zu schnell gefahren ist und trotzdem einen Strafzettel erhält, berechnet werden: $P(\bar{A} \cap B) = 0.01 \cdot 0.99 = 0.0099$. Auf gleichem Wege erhält man $P(A \cap B) = 0.8 \cdot 0.01 = 0.008$. Dieser Wert ist die Wahrscheinlichkeit, dass Bernhard zu schnell gefahren ist und einen Strafzettel erhält. Da Bernhard nur entweder zu schnell gefahren sein kann oder nicht, muss sich die Wahrscheinlichkeit $P(B)$, dass er einen Strafzettel erhält, aus den beiden zuletzt berechneten Werten aufaddieren:

$$P(B) = P(A \cap B) + P(\bar{A} \cap B). \quad (2.57)$$

Für die Beispielwerte ergibt sich $P(B) = 0.008 + 0.0099 = 0.0179$.

Schließlich kann mit Gleichung (2.56) berechnet werden, wie wahrscheinlich es ist, dass Bernhard zu Recht zur Kasse gebeten wird:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{0.8 \cdot 0.01}{0.0179} \approx 0.45.$$

Dieses Ergebnis sollte Bernhard zum Einspruch gegen die Forderung ermutigen. Wer keine ungerechtfertigten Strafzettel erhalten will, sollte schneller fahren: Für $P(A) = 0.5$ erhält man $P(A|B) \approx 0.99$.

In Gleichung (2.57) wurde $P(B)$ aus den Wahrscheinlichkeiten der Alternativen $A \cap B$ und $\bar{A} \cap B$ zusammengesetzt. Dies ist zulässig, da diese Aufteilung in „Teilereignisse“ disjunkt ist und B vollständig abdeckt. Die Bayes-Formel wird noch allgemeiner formuliert: B darf in beliebig viele, sogar unendlich viele Teilereignisse aufgeteilt werden. Denn es gilt für jedes Ereignis $B \in \wp(\Omega)$

$$P(B) = \sum_i P(B|A_i)P(A_i), \quad (2.58)$$

wenn $(A_i)_{i \in \mathbb{N}}$ eine endliche oder abzählbar unendliche Folge von paarweise disjunkten Ereignissen $A_i \in \wp(\Omega)$ mit $P(A_i) > 0$ für alle $i \in \mathbb{N}$ ist, die eine Zerlegung von Ω bildet, also $\Omega = \bigcup_i A_i$.

Um Gleichung (2.58) zu beweisen, ist in einem ersten Schritt die Zerlegung von Ω in eine Aufteilung des Ereignisses B umzuwandeln. Nach Voraussetzung lässt sich Ω schreiben als $\Omega = \bigcup_i A_i$. Dann muss aber auch $\Omega \cap B = (\bigcup_i A_i) \cap B$ gelten. Auf der linken Seite steht einfach B , die rechte kann wegen Distributivität in $\bigcup_i (A_i \cap B)$ umgeformt werden. Dies ist eine disjunkte Vereinigung von Ereignissen, weshalb nach Axiom 3 gilt: $P(B) = \sum_i P(B \cap A_i)$. Jetzt hilft wieder Formel (2.55) beim letzten Schritt, dem Umformen der rechten Seite zu $\sum_i P(B|A_i)P(A_i)$.

Abschließend soll die Bayes-Formel als Verschmelzung der Gleichungen (2.58) und (2.56) formuliert werden: Sei $(A_i)_{i \in \mathbb{N}}$ eine endliche oder abzählbar unendliche Folge von paarweise disjunkten Ereignissen $A_i \in \wp(\Omega)$ mit $P(A_i) > 0$ für alle $i \in \mathbb{N}$, die eine Zerlegung von Ω bildet, also $\Omega = \bigcup_i A_i$. Dann gilt für alle Ereignisse $A, B \in \wp(\Omega)$ mit $P(B) > 0$ die **Bayes-Formel**:

$$P(A|B) = \frac{P(B|A)P(A)}{\sum_i P(B|A_i)P(A_i)}. \quad (2.59)$$

Die Bayes-Formel und das Spracherkennungsproblem: Wie trägt die Bayes-Formel zur Lösung des Spracherkennungs-Problems bei? Gesucht ist diejenige Wortfolge w , welche die durch ein gemessenes Sprachsignal x bedingte Wahrscheinlichkeit $P(w|x)$ maximiert. Mit Gleichung (2.56) lässt sich diese Wahrscheinlichkeit umschreiben in

$$P(w|x) = \frac{P(x|w)P(w)}{P(x)}.$$

Dieser Wert soll bei feststehendem x maximiert werden, der Nenner kann also ignoriert werden. Übrig bleibt die Suche nach der Wortfolge w , die den Wert $P(x|w) \cdot P(w)$ maximiert.

$P(x|w)$ entspricht dabei der Frage, wie gut das Sprachsignal zu den verschiedenen Wortfolgen passt. Sie kann durch ein akustisches Modell beantwortet werden. Die **Hidden-Markov-Modelle**, die im nächsten Abschnitt behandelt werden, sind ein Beispiel eines solchen akustischen Modells.

$P(w)$ ist die allgemeine Wahrscheinlichkeit, dass eine Wortfolge w geäußert wird, verlangt also ein statistisches Sprachmodell. Die Wahrscheinlichkeiten aller möglichen Wortfolgen experimentell zu ermitteln und zu speichern, wird schwierlich möglich sein. Um Abhilfe zu schaffen, bietet sich ein weiterer wahrscheinlichkeitstheoretischer Satz an. Er erlaubt es, die Wahrscheinlichkeit $P(w)$ in viele bedingte Wahrscheinlichkeiten aufzuspalten. Dies geschieht anhand der Einteilung $w = w_1 w_2 \dots w_K$ in K kleinere Einheiten, z. B. Worte.

Zuerst soll nun der betreffende Satz allgemein hergeleitet und anschließend seine Anwendung auf $P(w)$ demonstriert werden. Seine Formulierung wird einfacher durch Verwendung der verbreiteten Konvention, das Zeichen für den Mengenschnitt wegzulassen: $P(AB|CDE)$ steht also für $P(A \cap B | C \cap D \cap E)$.

Es seien $A_1, A_2 \in \wp(\Omega)$ Ereignisse, so dass $P(A_1 \cap A_2) > 0$, und A_3 ein beliebiges Ereignis aus $\wp(\Omega)$. Wird Definition 2.4.3 auf $P(A_3|A_1 A_2)$ angewendet, so ergibt sich nach Umstellung $P(A_1 A_2 A_3) = P(A_3|A_1 A_2)P(A_1 A_2)$. Auf gleichem Wege erhält man $P(A_1 A_2) = P(A_2|A_1)P(A_1)$. Durch Einsetzen der zweiten Formel in die erste entsteht

$$P(A_1 A_2 A_3) = P(A_3|A_1 A_2)P(A_2|A_1)P(A_1).$$

Allgemein gilt für $n \geq 2$ und eine Folge von Ereignissen A_1, A_2, \dots, A_n mit $P(A_1 \dots A_{n-1}) > 0$ die **Kettenformel**:

$$\begin{aligned} P(A_1 \dots A_n) &= \\ &P(A_n|A_1 \dots A_{n-1})P(A_{n-1}|A_1 \dots A_{n-2}) \cdots P(A_2|A_1)P(A_1). \end{aligned} \quad (2.60)$$

Mit Formel (2.60) kann die Wahrscheinlichkeit eines Satzes $w = w_1 w_2 \dots w_K$ folgendermaßen aufgespalten werden:

$$\begin{aligned} P(w_1 \dots w_K) &= \\ &P(w_1)P(w_2|w_1) \cdots P(w_{K-1}|w_1 \dots w_{K-2})P(w_K|w_1 \dots w_{K-1}). \end{aligned} \quad (2.61)$$

Dies ist eine sehr komprimierte Schreibweise. $P(w_1)$ steht für die Wahrscheinlichkeit, dass an erster Stelle eines Satzes das konkrete Wort w_1 steht. Auf die Definition der Ereignisse als Mengen von Ergebnissen übertragen bezeichnet w_1 also die Menge aller Sätze, in denen das Wort w_1 an erster Stelle steht.

Beispiel 2.4.3

Die Wahrscheinlichkeit $P(w)$ des Satzes $w = Stefan \text{ trinkt Kaffee}$ wird gesucht. Formel (2.61) liefert folgende Aufspaltung:

$$\begin{aligned} P(Stefan \text{ trinkt Kaffee}) &= P(w_1 = Stefan) \\ &\cdot P(w_2 = \text{trinkt} \mid w_1 = Stefan) \\ &\cdot P(w_3 = \text{Kaffee} \mid w_1 = Stefan, w_2 = \text{trinkt}). \end{aligned}$$

□

Diese Aufspaltung hat die Komplexität des Problems nicht reduziert. Sie setzt zur Berechnung von $P(w)$ die Bekanntheit aller $P(w_i \mid w_1 \dots w_{i-1})$ für $i =$

$1, \dots, K$ voraus. Um eine tatsächliche Vereinfachung zu erzielen nimmt man an, dass das Auftreten eines Wortes nur von den $n - 1$ vorhergehenden Wörtern abhängt. Dies führt zu einem sogenannten **N-Gramm-Modell**. Die gängigsten Vertreter dieser Klasse sind die Unigramm-Modelle ($n = 1$), Bigramm-Modelle ($n = 2$) und Trigramm-Modelle ($n = 3$). In einem Trigramm-Modell nimmt man beispielsweise an, dass für alle Wortvorkommen

$$P(w_i | w_1 \dots w_{i-1}) = P(w_i | w_{i-2} w_{i-1})$$

gilt. Die einzelnen $P(w_i | w_{i-2} w_{i-1})$ lassen sich nun anhand eines Korpus bestimmen (vgl. Abschnitt 2.4.3). Bei N-Gramm-Modellen handelt es sich um die einfachste Form sog. Markov-Modelle. Eine komplexere Variante, die auf denselben Prinzipien beruht, wird in Abschnitt 2.4.2 ausführlich behandelt.

Entropie und das Maximum-Likelihood-Prinzip

Ein statistisches Modell, das wie die oben genannten N-Gramm-Modelle vereinfachende Annahmen macht, liefert eine Näherung Q für die tatsächliche Wahrscheinlichkeitsverteilung P . Ist das Modell korrekt formuliert, so erfüllt auch Q die Axiome eines Wahrscheinlichkeitsmaßes gemäß Definition 2.4.1.

Eine zentrale Frage bei der Entwicklung und Parameteroptimierung solcher Modelle ist, wie gut P durch Q angenähert wird. Wir benötigen also ein Maß, um zwei Wahrscheinlichkeitsverteilungen miteinander vergleichen zu können. Auch wenn die tatsächliche Verteilung P nicht bekannt ist, kann es von Interesse sein, unterschiedliche Modellverteilungen Q_1 und Q_2 zu vergleichen.

Die Informationstheorie (Shannon 1948; Shannon und Weaver 1949) liefert uns mit dem Begriff der Entropie einen Ausgangspunkt für die Definition eines geeigneten Vergleichsmaßes. Kurz und vereinfachend dargestellt setzt die Informationstheorie den **Informationsgehalt** eines Ereignisses mit seiner Wahrscheinlichkeit gleich. Genauer gesagt: je weniger wahrscheinlich ein Ereignis $A \in \wp(\Omega)$ ist, desto „überraschender“ und damit informativer ist es für uns. Mathematisch wird der Informationsgehalt von A definiert als $-\log_2 P(A)$ und kann als Anzahl unabhängiger Bits interpretiert werden.

Wir wollen uns diese Definition am Beispiel einer gerechten Münze veranschaulichen. Jedes mögliche Ergebnis (d.h. Kopf oder Zahl) liefert $-\log_2 \frac{1}{2} = \log_2 2 = 1$ Bit Information. Dieser Wert ist intuitiv verständlich, wenn wir Kopf und Zahl numerisch als 1 und 0 kodieren. Eine Folge von n Münzwürfen (z. B. Kopf, Zahl, Kopf, Kopf, Zahl) entspricht dann einer Binärzahl (hier 10110) der Länge n . In analoger Weise stellen wir fest, dass ein Wurf eines gerechten achtseitigen Würfels $-\log_2 \frac{1}{8} = 3$ Bits Information liefert: nämlich eine Zahl zwischen 0 und 7, die in 3 Bits binärkodiert werden kann. Der Informationsgehalt eines herkömmlichen sechsseitigen Würfels beträgt $-\log_2 \frac{1}{6} \approx 2.585$ Bits. Dieser Informationsgehalt reduziert sich, wenn eine ungleichmäßige Verteilung der Ergebnisse bekannt ist. Wissen wir etwa, dass ein Würfel fast immer eine 6 wirft (z. B. $P(6) = 0.9$), so ist dieses Ergebnis kaum überraschend ($-\log_2 P(6) \approx 0.15$) und damit nicht informativ.

Durch Mittelung über alle Ergebnisse eines diskreten Wahrscheinlichkeitsraums können wir nun den durchschnittlichen Informationsgehalt einer Wahrscheinlichkeitsverteilung bestimmen. Diese sogenannte **Entropie**

$$H[P] = - \sum_{\omega \in \Omega} P(\omega) \cdot \log_2 P(\omega) \quad (2.62)$$

ist ein Maß für die durchschnittliche „Überraschung“ bei vielen Wiederholungen des entsprechenden Zufallsexperiments. Im Beispiel eines sechsseitigen Würfels lässt sich Gleichung (2.62) folgendermaßen herleiten: In $P(1)$ Prozent aller Fälle wird eine 1 geworfen, die $-\log_2 P(1)$ Bits Information liefert. Der Beitrag dieser Einser-Würfe zur durchschnittlichen Information ist also $-P(1) \cdot \log_2 P(1)$. Gleiches gilt für Zweier, Dreier, usw., so dass wir insgesamt eine Entropie von $H[P] = -\sum_{n=1}^6 P(n) \cdot \log_2 P(n)$ erhalten.

Beispiel 2.4.4

Als konkretes Beispiel wollen wir die Entropie eines gerechten und eines nicht-gerechten Würfels berechnen, deren Wahrscheinlichkeitsverteilungen wie folgt gegeben sind:

	1	2	3	4	5	6
gerechter Würfel	1/6	1/6	1/6	1/6	1/6	1/6
ungerechter Würfel	1/4	1/16	1/16	1/16	1/16	1/2

Der ungerechte Würfel liefert also in 50% aller Würfe eine 6 und in 25% aller Würfe eine 1; die restlichen Augenzahlen sind gleichmäßig verteilt. Nach Gleichung (2.62) berechnen wir für den gerechten Würfel

$$H[P_{\text{gerecht}}] = - \sum_{n=1}^6 \frac{1}{6} \cdot \log_2 \frac{1}{6} = -\log_2 \frac{1}{6} \approx 2.585 \text{ Bits.}$$

Der durchschnittliche Informationsgehalt ist also identisch mit dem Informationsgehalt jeder einzelnen Augenzahl. Anschaulich ist klar, dass eine Gleichverteilung, wie sie beim gerechten Würfel vorliegt, eine maximale Entropie besitzt, da wir in diesem Fall keine Vorhersagen über den Ausgang des Zufallsexperiments machen können. Ist aber bekannt, dass es sich um den oben beschriebenen ungerichteten Würfel handelt, so erwarten wir, die 6 und die 1 wesentlich häufiger zu sehen als andere Augenzahlen. Im Mittel sind wir daher von den Ergebnissen des Zufallsexperiments weniger überrascht und berechnen eine niedrigere Entropie

$$H[P_{\text{ungerecht}}] = -\frac{1}{4} \log_2 \frac{1}{4} - \sum_{n=2}^5 \frac{1}{16} \log_2 \frac{1}{16} - \frac{1}{2} \log_2 \frac{1}{2} = 2 \text{ Bits.}$$

•

Dieses Beispiel zeigt, dass die Entropie ein Maß für die Gleichmäßigkeit einer Wahrscheinlichkeitsverteilung ist. Wie können wir ein solches Homogenitätsmaß nutzen, um Verteilungen zu vergleichen?

Man stelle sich dazu einen ungerechten Würfel mit tatsächlicher Wahrscheinlichkeitsverteilung P vor. Wir glauben aber, dass es sich um einen gerechten Würfel mit Gleichverteilung Q handele, sind also von jedem Ergebnis in gleichem Maße (nämlich $\log_2 6$ Bits) überrascht. Unsere durchschnittliche Überraschung bei wiederholten Würfen ist somit $\sum_{\omega \in \Omega} P(\omega) \cdot \log_2 6 = \log_2 6$. Diesen Wert bezeichnet man auch als **Kreuzentropie** $H[P, Q]$ (engl. *cross entropy*) zwischen der angenommenen Verteilung Q und der tatsächlichen Verteilung P . Für den ungerechten Würfel aus Beispiel 2.4.4 ist $H[P, Q]$ größer als die Entropie $H[P] = 2$ des Würfels. Dies ist intuitiv naheliegend: würden wir die tatsächliche Verteilung P kennen, so wäre unsere durchschnittliche Überraschung sicherlich geringer, nämlich genau $H[P, P] = H[P]$.

Ein wichtiges Ergebnis der Informationstheorie zeigt, dass für beliebige Wahrscheinlichkeitsverteilungen P und Q die Kreuzentropie $H[P, Q]$ immer größer oder gleich der tatsächlichen Entropie $H[P]$ ist (**Gibbs-Ungleichung**). Die Gleichheit tritt genau dann ein, wenn P und Q identisch sind. Wir können also die Differenz zwischen $H[P, Q]$ und $H[P]$ als ein Maß für die Ähnlichkeit von Q zu P heranziehen. Dies motiviert die folgende

Definition 2.4.5

Gegeben seien zwei diskrete Wahrscheinlichkeitsräume $\langle \Omega, P \rangle$ und $\langle \Omega, Q \rangle$ über derselben Ergebnismenge Ω . Die **Entropie** $H[P]$ der Wahrscheinlichkeitsverteilung P ist gegeben durch

$$H[P] = - \sum_{\omega \in \Omega} P(\omega) \cdot \log_2 P(\omega).$$

Die **Kreuzentropie** $H[P, Q]$ zwischen Q und P ist definiert als

$$H[P, Q] = - \sum_{\omega \in \Omega} P(\omega) \cdot \log_2 Q(\omega).$$

Die Differenz zwischen Kreuzentropie und Entropie wird als **Kullback-Leibler-Divergenz** $D(P||Q)$ bezeichnet (oder kurz als **KL-Divergenz**). Sie kann kompakt durch die folgende Formel berechnet werden:

$$D(P||Q) = H[P, Q] - H[P] = \sum_{\omega \in \Omega} P(\omega) \cdot \log_2 \frac{P(\omega)}{Q(\omega)}.$$

Für beliebige Verteilungen P und Q gilt stets $D(P||Q) \geq 0$. Gleichheit von Entropie und Kreuzentropie, also der Fall $D(P||Q) = 0$, tritt genau dann ein, wenn die Verteilungen identisch sind, also $P = Q$ gilt. \square

Beispiel 2.4.5

Zur Veranschaulichung berechnen wir die KL-Divergenz des gerechten Würfels

(mit Gleichverteilung Q) von dem ungerechten Würfel aus Beispiel 2.4.4 (Verteilung P). Die Kreuzentropie beträgt

$$H[P, Q] = -\frac{1}{4} \log_2 \frac{1}{6} - 4 \cdot \frac{1}{16} \log_2 \frac{1}{6} - \frac{1}{2} \log_2 \frac{1}{6} = \log_2 6 \approx 2.585 \text{ Bits.}$$

Mit $H[P] = 2$ Bits erhalten wir eine KL-Divergenz von

$$D(P\|Q) \approx 2.585 - 2 = 0.585 \text{ Bits.}$$

Unsere mittlere Überraschung ist also um 0.585 Bits erhöht, wenn wir irrtümlicherweise annehmen, dass es sich um einen gerechten Würfel handele. Berechnen wir umgekehrt die KL-Divergenz zwischen ungerechtem und gerechtem Würfel (dies entspricht dem Fall, dass wir irrtümlicherweise einen ungerechten Würfel vermuten), so erhalten wir zunächst

$$H[Q, P] = -\frac{1}{6} \log_2 \frac{1}{4} - 4 \cdot \frac{1}{6} \log_2 \frac{1}{16} - \frac{1}{6} \log_2 \frac{1}{2} = \frac{19}{6} \approx 3.167 \text{ Bits.}$$

Mit $H[Q] = \log_2 6$ ergibt sich daraus eine KL-Divergenz von

$$D(Q\|P) = H[Q, P] - H[Q] \approx 3.167 - 2.585 = 0.582 \text{ Bits.}$$

□

Dieses Beispiel zeigt, dass Kreuzentropie und KL-Divergenz nicht symmetrisch sind. In der Regel wird die KL-Divergenz $D(P\|Q)$ einer Modellverteilung Q von der tatsächlichen Wahrscheinlichkeitsverteilung P berechnet. Sollen zwei Modellverteilungen Q_1 und Q_2 verglichen werden, bietet es sich an, die symmetrische Divergenz $\frac{1}{2}(D(P\|Q) + D(Q\|P))$ zu verwenden.

Bei der praktischen Bestimmung der KL-Divergenz stellt sich das Problem, dass die tatsächliche Verteilung P üblicherweise nicht genau bekannt ist: das statistische Modell hat gerade den Zweck, diese unbekannte Verteilung anzunähern. Zur Lösung ziehen wir die Häufigkeitsinterpretation von Wahrscheinlichkeiten heran. Wird das zugrunde liegende Zufallsexperiment wiederholt durchgeführt, so erhalten wir eine Folge zufälliger Ergebnisse $x_1, x_2, \dots, x_M \in \Omega$. Bei einer hinreichend großen Anzahl von Wiederholungen sollten die relativen Häufigkeiten $\hat{p}(\omega) = f(\omega)/M$ der Ergebnisse $\omega \in \Omega$ eine gute Näherung für die tatsächlichen Wahrscheinlichkeiten $P(\omega)$ darstellen. In statistischen Begriffen ausgedrückt handelt es sich bei der Folge (x_1, x_2, \dots, x_M) um eine Zufallsstichprobe der Größe M aus der Verteilung P . Die Häufigkeit $f(\omega)$ eines Ergebnisses $\omega \in \Omega$ kann formal definiert werden als die Anzahl der Positionen $i \in \{1, \dots, M\}$ mit $x_i = \omega$, also

$$f(\omega) = |\{i \mid 1 \leq i \leq M, x_i = \omega\}| = \sum_{i=1}^M \chi_{[x_i=\omega]}. \quad (2.63)$$

Die charakteristische Funktion $\chi_{[x_i=\omega]}$ nimmt dabei den Wert 1 an, wenn der Ausdruck in eckigen Klammern erfüllt ist, und sonst den Wert 0.

Wir können nun in der Berechnung von Kreuzentropie und KL-Divergenz die tatsächliche Verteilung P durch die relativen Häufigkeiten \hat{p} ersetzen und erhalten damit die Näherungen $H[P, Q] \approx H[\hat{p}, Q]$ und $D(P\|Q) \approx D(\hat{p}\|Q)$. Im Beispiel eines ungerechten Würfels mit unbekannter Verteilung nehmen wir zunächst eine Gleichverteilung Q an. Um die tatsächliche Verteilung P abzuschätzen, die dem ungerechten Würfel aus Beispielen 2.4.4 und 2.4.5 entspreche, führen wir $M = 100$ Würfe aus und erhalten folgende Häufigkeitstabelle:

ω	1	2	3	4	5	6
$f(\omega)$	22	6	11	5	9	47
$\hat{p}(\omega)$	0.22	0.06	0.11	0.05	0.09	0.47

Daraus berechnen wir eine Entropie von $H[\hat{p}] \approx 2.115$, eine Kreuzentropie von $H[\hat{p}, Q] \approx 2.585$ und schließlich als Näherung für die KL-Divergenz

$$D(\hat{p}\|Q) = H[\hat{p}, Q] - H[\hat{p}] \approx 2.585 - 2.115 = 0.470.$$

Aus Beispiel 2.4.5 wissen wir, dass die korrekte KL-Divergenz etwas größer ist, nämlich $D(P\|Q) \approx 0.585$.

Bei der in Abschnitt 2.4.1 vorgestellten Anwendung statistischer Modelle auf das Spracherkennungsproblem besteht die Ergebnismenge $\Omega = \Sigma^*$ aus allen möglichen Sätzen über dem Alphabet Σ , das alle Wörter der betrachteten Sprache enthalten muss. Es handelt sich also um eine zumindest im Prinzip abzählbar unendliche Anzahl von Ergebnissen. Eine Zufallsstichprobe aus der tatsächlichen Wahrscheinlichkeitsverteilung P bildet in diesem Fall ein **Korpus** zufällig ausgewählter Sätze x_1, x_2, \dots, x_M . Solche Korpora spielen eine wichtige Rolle in der modernen Computerlinguistik und sind für viele Sprachen im Umfang von mehreren Millionen Sätzen verfügbar (also $M > 10^6$). Trotz der enormen Größe dieser Korpora stellen die relativen Häufigkeiten $\hat{p}(\omega)$ nur eine grobe Näherung für die tatsächlichen Wahrscheinlichkeiten $P(\omega)$ dar. Insbesondere wird es aufgrund des **Zipfschen Gesetzes** (Zipf 1949; Leopold 2002) sehr viele (plausible) Sätze geben, die in dem gewählten Korpus überhaupt nicht oder nur ein einziges Mal vorkommen.

Die praktische Berechnung von $D(\hat{p}\|Q)$ über eine Auszählung der Häufigkeiten aller möglichen Sätze $\omega \in \Sigma^*$ ist sehr umständlich. Wir wollen nun versuchen, durch geeignete Umformungen den Rechenaufwand zu reduzieren. Zunächst stellen wir fest, dass ein Ergebnis ω mit $\hat{p}(\omega) = 0$ keinen Beitrag zur Kreuzentropie leistet, da in diesem Fall $-\hat{p}(\omega) \cdot \log_2 Q(\omega) = 0 \cdot \log_2 Q(\omega) = 0$ ist. Gleichermaßen gilt für die Entropie $H[\hat{p}]$, wenn wir $0 \cdot \log_2 0 = 0$ definieren (als stetige Fortsetzung der Funktion $x \cdot \log_2 x$ für $x \rightarrow 0$). Es genügt also, bei der Berechnung von $H[\hat{p}]$ und $H[\hat{p}, Q]$ nur über diejenigen Sätze ω zu summieren, welche tatsächlich in dem gewählten Korpus vorkommen.

Eine weitere Vereinfachung ergibt sich dadurch, dass es für den Vergleich und die Optimierung statistischer Modelle nicht zwingend erforderlich ist, die KL-Divergenz $D(\hat{p}\|Q)$ oder die Entropie $H[\hat{p}]$ des Korpus zu berechnen. Da $H[\hat{p}]$ für alle Modelle Q gleich ist, genügt es, den Wert der Kreuzentropie $H[\hat{p}, Q]$ zu vergleichen. Je kleiner diese ist, desto besser ist Q an das Korpus angepasst. Nach

diesen Vorüberlegungen formen wir $H[\hat{p}, Q]$ nun so um, dass auf die explizite Bestimmung von Satzhäufigkeiten $f(\omega)$ verzichtet werden kann.

$$\begin{aligned} H[\hat{p}, Q] &= - \sum_{\omega \in \Omega} \hat{p}(\omega) \cdot \log_2 Q(\omega) = - \sum_{\omega \in \Omega} \frac{f(\omega)}{M} \cdot \log_2 Q(\omega) \\ &= - \frac{1}{M} \sum_{\omega \in \Omega} \sum_{i=1}^M \chi_{[x_i=\omega]} \cdot \log_2 Q(\omega) \end{aligned}$$

Im letzten Schritt haben wir die formale Definition (2.63) von $f(\omega)$ eingesetzt. Wir können jetzt die beiden Summenzeichen vertauschen. Die einzelnen Terme sind aufgrund der charakteristischen Funktion nur dann von 0 verschieden, wenn $x_i = \omega$ gilt, so dass wir $Q(\omega)$ durch $Q(x_i)$ ersetzen können. Damit erhalten wir:

$$\begin{aligned} H[\hat{p}, Q] &= - \frac{1}{M} \sum_{i=1}^M \sum_{\omega \in \Omega} \chi_{[x_i=\omega]} \cdot \log_2 Q(x_i) \\ &= - \frac{1}{M} \sum_{i=1}^M \log_2 Q(x_i) \cdot \sum_{\omega \in \Omega} \chi_{[x_i=\omega]} \end{aligned}$$

Dabei ist $\sum_{\omega \in \Omega} \chi_{[x_i=\omega]} = 1$, weil die Bedingung $x_i = \omega$ für genau ein ω wahr wird. Dieser Term kann also weggelassen werden und wir erhalten schließlich

$$H[\hat{p}, Q] = - \frac{1}{M} \sum_{i=1}^M \log_2 Q(x_i) = - \frac{1}{M} \log_2 \prod_{i=1}^M Q(x_i). \quad (2.64)$$

$\prod_{i=1}^M Q(x_i) = Q(x_1) \cdot Q(x_2) \cdot \dots \cdot Q(x_M)$ ist nichts anderes als die **Modellwahrscheinlichkeit** (engl. *likelihood*) des Korpus. Unter der Annahme, dass die einzelnen Sätze x_i voneinander unabhängig sind, ergibt sie sich als Produkt der Satzwahrscheinlichkeiten $Q(x_i)$. Die Kreuzentropie $H[\hat{p}, Q]$ ist also umso niedriger, und die Anpassung des Modells an das Korpus umso besser, je höher die Modellwahrscheinlichkeit des Korpus ist. Mit anderen Worten: dasjenige Modell Q ist am besten, welches dem Korpus die höchste Wahrscheinlichkeit zuweist. Diese grundlegende Einsicht ist als **Maximum-Likelihood-Prinzip** bekannt. Aus ihm können u.a. Methoden zur Parameterschätzung statistischer Modelle abgeleitet werden (siehe Abschnitte 2.4.2 und 2.4.3).

2.4.2 Hidden-Markov-Modelle

Hidden-Markov-Modelle, im Folgenden mit **HMM** abgekürzt, wurden in einer Reihe von klassischen Aufsätzen von Baum und seinen Kollegen Ende der 60er und Anfang der 70er Jahre beschrieben. Die grundlegende Theorie wurde von Baum Ende der 60er Jahre entwickelt (Baum und Petrie 1966; Baum und Eagon 1967). Die ersten Implementierungen in den 70er Jahren stammen von Baker an der CMU (Baker 1975) und Jelinek bei IBM (Jelinek 1976). In der Praxis erreichte das Verfahren durch die Arbeiten von Rabiner zur Spracherkennung

(Rabiner 1989; Rabiner und Juang 1993) sowie Church und anderen zur Wortartenannotierung (Church 1988; Schmid 1995; Brants 2000b) große Bedeutung. In nahezu allen heute kommerziell vertriebenen Spracherkennern werden HMMs zur Mustererkennung verwendet.

Motivation: HMMs in der Spracherkennung

Vor der Verwendung eines HMM in einem Spracherkennungssystem wird das mit einem Mikrofon aufgenommene Sprachsignal in eine Folge von **Merkmalsvektoren** transformiert: In kurzen Zeitabständen wird das analoge Signal abgetastet, z. B. mit 22 kHz, also 22000 mal in der Sekunde. Zu jedem Zeitpunkt wird ein diskreter Signalwert mit ausreichend hoher (typisch: 16 Bit) Genauigkeit gespeichert. Diese Werte werden in der Regel in gleichmäßig langen Zeitfenstern zusammengefasst, die sich auch überlappen können. Zu jedem Fenster werden bestimmte Merkmale berechnet, die den Signalabschnitt innerhalb des Fensters möglichst gut repräsentieren sollen. Unter anderem werden hier üblicherweise die Fourierkoeffizienten des Signalabschnitts herangezogen. Bei der Erstellung der Merkmalsvektoren ist insbesondere darauf zu achten, dass dieser Vorverarbeitungsprozess möglichst robust gegen Hintergrundgeräusche, Stimmveränderungen etc. ist, also nur die für die Spracherkennung tatsächlich relevanten Parameter liefert. Einen guten Überblick über die verschiedenen Merkmalsextraktionsverfahren bietet das Standardwerk von Schukat-Talamazzini (1995, 45–74).

Das eigentliche **Problem der Spracherkennung** lautet wie folgt: Wie kann aus dem Signal respektive der Merkmalsvektorfolge auf die tatsächlich gesprochene Wortfolge geschlossen werden? Wenn wir für alle möglichen Wortfolgen die Wahrscheinlichkeiten $P(\text{Wortfolge} \mid \text{Sprachsignal})$ bestimmen könnten, dann würden wir natürlich die Wortfolge mit der größten Wahrscheinlichkeit auswählen. Die direkte Berechnung aller Wahrscheinlichkeiten ist aber viel zu umfangreich, da es unendlich viele Wortfolgen gibt. Wir nähern uns dem Problem, in dem wir einige Umformungen vornehmen. Laut der Bayes-Formel gilt folgender Zusammenhang:

$$P(\text{Wortfolge} \mid \text{Sprachsignal}) = \frac{P(\text{Sprachsignal} \mid \text{Wortfolge}) \cdot P(\text{Wortfolge})}{P(\text{Sprachsignal})}$$

Dabei ist $P(\text{Sprachsignal})$ ein konstanter Skalierungsfaktor, da das Sprachsignal nicht davon abhängt, welche Wortfolge gerade zum Vergleich herangezogen wird. Für die Auswahl der besten Wortfolge bei gegebenem Sprachsignal kann dieser Faktor also ignoriert werden.

$P(\text{Wortfolge})$ kann als statistisches Maß für die syntaktische und semantische Plausibilität der Satzkonstruktion im Rahmen eines vorgegebenen Anwendungsbereiches aufgefasst werden. Diese Wahrscheinlichkeitsverteilung repräsentiert also das linguistische **Sprachmodell**. Eine relativ gute Näherung kann mittels der in Unterkapitel 3.2 beschriebenen N-Gramm-Modelle gewonnen werden, die auf der Auszählung von Wortfolgen in geeigneten Texten basieren.

$P(\text{Sprachsignal} \mid \text{Wortfolge})$ bzw. $P(\text{Merkmalsfolge} \mid \text{Wortfolge})$ ist also der letzte noch fehlende Baustein in unserer Berechnung. An dieser Stelle kommen nun HMMs ins Spiel: In einem automatischen Trainingsverfahren kann zu jedem Wort aus vielen Beispielen ein adäquates HMM erstellt werden, dessen Übereinstimmung mit einer zu testenden Äußerung effizient überprüft werden kann. Die beste Wortfolge kann auf einfache Art durch hintereinander geschaltete Wort-HMMs bestimmt werden. Eine Wortfolge kann aber auch durch die Verkettung von HMMs für Phoneme, Halbsilben, Silben etc. erstellt werden. Die spezielle Art der Verwendung spielt bei den folgenden allgemeinen Definitionen von HMMs und Algorithmen keine Rolle.

Allgemeine Definitionen für ein HMM

Ein Hidden-Markov-Modell besitzt N Zustände. Der Startzustand wird zufällig „gelöst“, hängt also von einer Wahrscheinlichkeitsverteilung ab. In jedem Zustand gibt das Modell ein Symbol aus. Danach wechselt es in einen neuen Zustand (auch derselbe Zustand ist erlaubt) und gibt wieder ein Symbol aus. Die Auswahl des neuen Zustands folgt einer Wahrscheinlichkeitsverteilung, die nur von dem aktuellen Zustand abhängt. Es ist also für die Auswahl egal, über welche anderen Zustände das Modell in diesen Zustand gekommen ist. Einen solchen „gedächtnislosen“ Prozess nennt man auch **Markov-Prozess**. Die Symbolausgabe folgt ebenfalls einer Wahrscheinlichkeitsverteilung, die lediglich vom aktuellen Zustand abhängt. Ein außenstehender Betrachter sieht nur die Folge der Symbole, weiß aber nicht, in welchem Zustand das Modell ein bestimmtes Symbol ausgibt. Die Zustandsfolge ist also verborgen (engl. *hidden*), die ausgegebene Symbolfolge sichtbar (engl. *observed*). Das Modell kann demnach wie folgt definiert werden:

- Verborgener Teil des Modells:
 - Es gibt N verborgene Zustände s_1, \dots, s_N .
 - Die zeitliche Abfolge der Zustände wird in der Folge $\mathbf{q} = q_1 \dots q_T$ festgehalten.
 - π_i bezeichnet die Wahrscheinlichkeit, zum Zeitpunkt $t = 1$ im Zustand i zu starten, d.h. für $1 \leq i \leq N$ sei

$$\pi_i = P(q_1 = s_i).$$

Die **Startwahrscheinlichkeiten** π_i bilden eine Wahrscheinlichkeitsverteilung, müssen also die **Normierungsbedingung** $\sum_{i=1}^N \pi_i = 1$ erfüllen. Wir speichern sie im N -dimensionalen Vektor

$$\boldsymbol{\pi} = (\pi_1, \dots, \pi_N).$$

- a_{ij} sei die Wahrscheinlichkeit, zu einem beliebigen Zeitpunkt t vom Zustand i in den Zustand j zum Zeitpunkt $t + 1$ zu wechseln, d.h. für $1 \leq i, j \leq N, 1 \leq t < T$ sei

$$a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i).$$

Die **Übergangswahrscheinlichkeiten** a_{ij} bilden für jeden Ausgangszustand i eine Wahrscheinlichkeitsverteilung, müssen also für jedes $1 \leq i \leq N$ die Normierungsbedingung $\sum_{j=1}^N a_{ij} = 1$ erfüllen. Wir speichern die Übergangswahrscheinlichkeiten in der $N \times N$ -Matrix

$$\mathbf{A} = [a_{ij}]_{N \times N}.$$

- Sichtbarer Teil des Modells:

- Es gibt K sichtbare Symbole v_1, \dots, v_K .
- Die zeitliche Abfolge der sichtbaren Symbole wird in der Beobachtungssequenz $\mathbf{O} = o_1 \dots o_T$ aufgeschrieben.
- b_{jk} bezeichnet die Wahrscheinlichkeit, zu einem beliebigen Zeitpunkt t im Zustand j das Symbol v_k auszugeben, d.h. für $1 \leq j \leq N$, $1 \leq k \leq K$ und $1 \leq t \leq T$ sei

$$b_{jk} = b_j(v_k) = P(o_t = v_k \mid q_t = j).$$

Die **Ausgabewahrscheinlichkeiten** b_{jk} bilden für jeden verborgenen Zustand j eine Wahrscheinlichkeitsverteilung mit der Normierungsbedingung $\sum_{k=1}^K b_{jk} = 1$ für jedes $1 \leq j \leq N$. Wir speichern sie in der $N \times K$ -Matrix

$$\mathbf{B} = [b_{jk}]_{N \times K}.$$

Die HMM-Spezifikation fassen wir zusammen in das Tripel

$$\lambda = (\mathbf{A}, \mathbf{B}, \pi).$$

HMM für ein gesprochenes Wort

Die Funktionsweise eines HMM wird nun anhand eines konkreten Beispiels erläutert. Es sollen verschiedene Ausspracheverarianten des Wortes „haben“ mit Verzögerungen und Verschleifungen modelliert werden. Wir wählen für jedes darin vorkommende Phonem einen Zustand, also $N = 5$ Zustände, die wir hier in SAMPA-Notation (Fourcin et al. 1989, 141–159) bezeichnen: $s_1 = /h/$, $s_2 = /a/$, $s_3 = /b/$, $s_4 = /@/$ und $s_5 = /n/$. Die tatsächlich zu hörenden Laute werden mit $K = 7$ Symbolen modelliert: $v_1 = [h]$, $v_2 = [a]$, $v_3 = [O]$, $v_4 = [b]$, $v_5 = [@]$, $v_6 = [n]$ und $v_7 = [m]$. Das Modell wird von links nach rechts durchlaufen, wobei es mit bestimmten Wahrscheinlichkeiten über mehrere Zeitschritte in einem Zustand verharren oder auch einen Zustand überspringen kann.

Abbildung 2.17 zeigt eine graphische Darstellung dieses HMM, wobei die Wahrscheinlichkeiten für Zustandsübergänge neben den entsprechenden Pfeilen eingetragen wurden. Der Startzustand ist nur $/h/$, hat also eine Startwahrscheinlichkeit $\pi_1 = 1$. Für Übergänge mit Wahrscheinlichkeit $a_{ij} = 0$ wurden keine Pfeile eingezeichnet. Unterhalb eines Zustandes stehen die möglichen Ausgabe-symbole mit den zugehörigen Wahrscheinlichkeiten, wobei wiederum alle Symbole mit Ausgabewahrscheinlichkeit $b_{jk} = 0$ weggelassen wurden. Die Ähnlichkeit

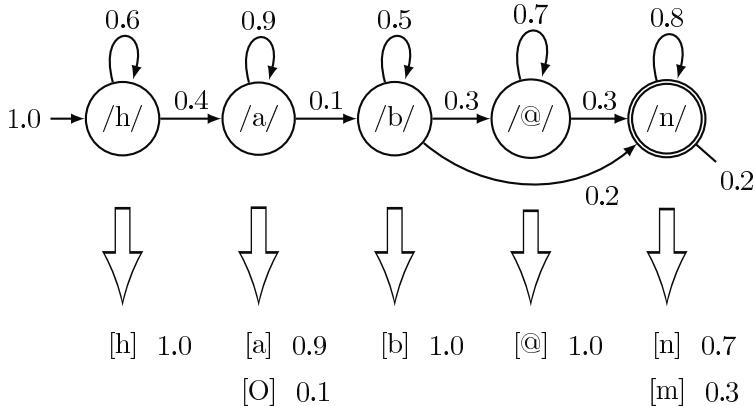


Abbildung 2.17: HMM für das Wort „haben“

von Abbildung 2.17 zu einem endlichen Automaten (vgl. Unterkapitel 2.2) ist kein Zufall: Markov-Modelle können als eine stochastische Erweiterung solcher Automaten betrachtet werden.

Wir wollen uns die Funktionsweise des HMM für das Wort „haben“ mit einem Beispiel veranschaulichen: Das Modell muss im Zustand /h/ starten. Dort gibt es auch nur die Möglichkeit, das Symbol [h] auszugeben. Nun wird gelost: Mit 60% Wahrscheinlichkeit bleiben wir dabei im Zustand /h/, mit 40% Wahrscheinlichkeit wird Zustand /a/ ausgelost. Wir nehmen an, das Los fiel auf den Wechsel zu Zustand /a/. Dort wird nun gelöst, ob Symbol [a] oder [O] emittiert wird und zwar mit den Wahrscheinlichkeiten 90% bzw. 10%. In diesem Fall sei es das [a]. Wieder wird gelöst, ob wir in /a/ bleiben (90% Wahrscheinlichkeit) oder nach /b/ übergehen (10% Wahrscheinlichkeit). Nehmen wir an, das Modell bleibt in /a/,lost wiederum [a] aus und geht danach nach /b/ über. Dort wird mit Sicherheit [b] ausgegeben. Anschließend gibt es drei Möglichkeiten: Mit 50% Wahrscheinlichkeit bleibt das Modell in /b/, mit 30% Wahrscheinlichkeit gibt es einen Übergang zu @/ und mit 20% Wahrscheinlichkeit zu /n/. Der letzte Fall sei eingetreten. Nun wird entschieden, ob [n] (70% Wahrscheinlichkeit) oder [m] (30% Wahrscheinlichkeit) ausgegeben wird. Hier sei es ein [n]. Anschließend wird gelöst, ob wir im Zustand /n/ bleiben (80%) oder der Prozess beendet wird (20%). Letzteres sei der Fall, also wurde insgesamt die Symbolfolge [h][a][a][b][n] produziert. Die Gesamtwahrscheinlichkeit, dass genau diese Symbolfolge erzeugt wird, ergibt sich durch Multiplikation aller gelosten Übergangs- und Ausgabe-wahrscheinlichkeiten, hier also $0.4 \cdot 0.9 \cdot 0.9 \cdot 0.9 \cdot 0.1 \cdot 0.2 \cdot 0.7 \cdot 0.2 = 0.00081648$ oder ca. 0.08%. Die auf den ersten Blick sehr niedrige Wahrscheinlichkeit erklärt sich durch die große Anzahl möglicher (und plausibler) phonetischer Realisierungen.

Die komplette formale Spezifikation des oben dargestellten Modells lautet:

$$\boldsymbol{\pi} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{A} = \begin{pmatrix} 0.6 & 0.4 & 0 & 0 & 0 \\ 0 & 0.9 & 0.1 & 0 & 0 \\ 0 & 0 & 0.5 & 0.3 & 0.2 \\ 0 & 0 & 0 & 0.7 & 0.3 \\ 0 & 0 & 0 & 0 & 0.8 \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.9 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.7 & 0.3 \end{pmatrix}$$

In jeder Zeile der Matrizen \mathbf{A} und \mathbf{B} müssen sich aufgrund der Normierungsbedingungen die Wahrscheinlichkeiten zu 1 ergänzen; gleiches gilt für den Spaltenvektor $\boldsymbol{\pi}$. In der letzten Zeile von \mathbf{A} wird diese Regel scheinbar verletzt. Der Grund hierfür ist, dass das HMM im Zustand $/n/$ mit Wahrscheinlichkeit 0.2 terminiert, also keine weiteren Laute mehr ausgibt. In Abbildung 2.17 wird dies durch einen Doppelkreis angedeutet, der zusätzlich mit der Terminierungswahrscheinlichkeit 0.2 annotiert ist. Alternativ könnte man einen zusätzlichen Endzustand s_E einführen, der mit Wahrscheinlichkeit 1 in s_E verbleibt und dabei jeweils ein spezielles „Ende“-Symbol emittiert. Die hier verwendete Notation ist aber kürzer und erlaubt eine einfache Einbettung des HMM als Zustand in einem übergeordneten HMM (z. B. ein Aussprachemodell für beliebige Sätze).

Bei dem in diesem Abschnitt vorgestellten HMM handelt es sich um ein sogenanntes **Links-Rechts-Modell**, kurz L-R-Modell, bei dem nur Übergänge von links nach rechts vorkommen (also zu Zuständen mit gleichen oder höheren Ordnungsnummern). Die linke untere Hälfte der Matrix \mathbf{A} enthält deshalb nur Nullen. Ein Spezialfall davon ist das **lineare Modell**, bei dem nur Übergänge von einem Zustand zu sich selbst oder zum rechten Nachbarn erlaubt sind. Beim **Bakis-Modell** sind außerdem Übergänge zum übernächsten rechten Nachbarn erlaubt. Unser Beispiel ist also auch ein Bakis-Modell. Werden nur solche Modelle eingesetzt, so können die im Folgenden behandelten Algorithmen deutlich vereinfacht werden.

Grundprobleme bei der Bestimmung von HMMs

Im obigen Beispiel wurden die Wahrscheinlichkeiten „von Hand“ bestimmt. Wir werden nun sehen, wie wir diese Parameter automatisch ermitteln können, wenn verschiedene Aussprachen eines Wortes in ein HMM integriert werden sollen. Auf dem Weg dorthin müssen wir drei Grundprobleme lösen, die in den drei folgenden Abschnitten ausführlich beschrieben werden.

- Das **Beobachtungswahrscheinlichkeitsproblem**: Wir betrachten ein festes Modell λ . Wie gut passt eine Beobachtungssequenz $\mathbf{O} = o_1 \dots o_T$ zu diesem Modell? Gesucht ist also die Wahrscheinlichkeit $P(\mathbf{O}|\lambda)$. Für das

obige Beispiel entspricht das der Frage, wie wahrscheinlich es ist, dass das „haben“-HMM z. B. die Sequenz [h][a][a][b][n] produziert.

- Das **Problem der optimalen Zustandsfolge**: Welches ist die wahrscheinlichste Sequenz eingenommener Zustände bei gegebener Beobachtungssequenz? Im Beispiel: Welche Zustandsfolge war für die Beobachtung [h][a][a][b][n] die wahrscheinlichste? Es geht also darum, den verborgenen Teil des Modells offen zu legen.
- Das **Parameteroptimierungsproblem**: Wie optimiert man für gegebene Sprachsignale die Parameter von Modell λ , um $P(\mathbf{O}|\lambda)$ zu maximieren? Im Beispiel entspricht das der folgenden Problemstellung: Zu vielen gesammelten Aussprüchen des Wortes „haben“, z. B. [h][a][a][b][m], [h][O][b][n], [h][a][b][@][n], [h][a][a][b][m], usw. sollen die Parameter \mathbf{A} , \mathbf{B} und π so angepasst werden, dass das Modell die gleichen typischen Aussprüchen produziert.

Es sei darauf hingewiesen, dass in unserem Beispiel die verborgene Zustandsfolge eindeutig aus der Beobachtungssequenz \mathbf{O} abgelesen werden kann, da es keine zwei Zustände gibt, die denselben Laut emittieren. In diesem Fall lässt sich das Problem der optimalen Zustandsfolge also trivial lösen; im allgemeinen sind jedoch die in Abschnitt 2.4.2 behandelten Methoden erforderlich.

Beobachtungswahrscheinlichkeit

Wir wollen jetzt die Wahrscheinlichkeit der Beobachtungssequenz $\mathbf{O} = o_1 \dots o_T$ für das Modell λ berechnen, also $P(\mathbf{O}|\lambda)$. Gäbe es nur eine mögliche Zustandsfolge $\mathbf{q} = q_1 \dots q_T$, dann ließe sich die gesuchte Wahrscheinlichkeit einfach aus dem Produkt der Start-, Ausgabe- und Übergangswahrscheinlichkeiten ermitteln. Mathematisch gesprochen handelt es sich hierbei um die gemeinsame Wahrscheinlichkeit von Zustandsfolge und Beobachtungssequenz:

$$\begin{aligned} P(\mathbf{q}, \mathbf{O}|\lambda) &= \pi_{q_1} \cdot b_{q_1}(o_1) \cdot a_{q_1 q_2} \cdot b_{q_2}(o_2) \cdot \dots \cdot a_{q_{T-1} q_T} \cdot b_{q_T}(o_T) \\ &= \pi_{q_1} \cdot b_{q_1}(o_1) \cdot \prod_{t=1}^{T-1} a_{q_t q_{t+1}} \cdot b_{q_{t+1}}(o_{t+1}) \end{aligned} \quad (2.65)$$

Diese Gleichung ist folgendermaßen zu interpretieren: In der ersten Zeiteinheit ($t = 1$) befinden wir uns im Zustand q_1 mit Wahrscheinlichkeit π_{q_1} und generieren das Symbol o_1 mit Wahrscheinlichkeit $b_{q_1}(o_1)$. Der Zeitpunkt wechselt von t nach $t + 1$, wir gehen vom Zustand q_t in den Zustand q_{t+1} mit Wahrscheinlichkeit $a_{q_t q_{t+1}}$ über und generieren dort das Symbol o_{t+1} mit Wahrscheinlichkeit $b_{q_{t+1}}(o_{t+1})$. Weil die Übergänge und die Ausgaben unabhängig sind, dürfen wir die Wahrscheinlichkeiten multiplizieren. Dieser Schritt ist nun bis zum letzten Übergang (von $T - 1$ nach T) zu wiederholen.

Um die Gesamtwahrscheinlichkeit einer Ausgabe \mathbf{O} zu bestimmen, müssen die Wahrscheinlichkeiten $P(\mathbf{q}, \mathbf{O}|\lambda)$ für alle möglichen Zustandsfolgen der Länge T

aufsummiert werden, die wir in der Menge Q_T zusammenfassen:

$$P(\mathbf{O}|\boldsymbol{\lambda}) = \sum_{\mathbf{q} \in Q_T} P(\mathbf{q}, \mathbf{O}|\boldsymbol{\lambda}) = \sum_{\mathbf{q} \in Q_T} \pi_{q_1} \cdot b_{q_1}(o_1) \cdot \prod_{t=1}^{T-1} a_{q_t q_{t+1}} \cdot b_{q_{t+1}}(o_{t+1}). \quad (2.66)$$

Die Menge Q_T wird in Abbildung 2.18 in Form eines Gitters veranschaulicht. Zur Bestimmung des Rechenaufwands nummerieren wir die Elemente von Q_T durch, müssen also alle möglichen Zustandsfolgen $\mathbf{q} = q_1 \dots q_T$ der Länge T aufzählen. Da zu jedem Zeitpunkt N verschiedene Zustände möglich sind, gibt es insgesamt N^T solcher Folgen. Der Rechenaufwand für eine Folge beträgt $2T - 1$ Multiplikationen, für alle Folgen also $(2T - 1) \cdot N^T$ Multiplikationen sowie $N^T - 1$ Additionen. Insgesamt benötigen wir für Gleichung (2.66) also $2TN^T - 1$ Rechenschritte. Wenn wir z. B. mit 5 Zuständen und 100 Beobachtungen arbeiten, dann wären bereits $2 \cdot 100 \cdot 5^{100} - 1$, also etwa 10^{72} Berechnungen durchzuführen. Es gibt aber einen wesentlich effizienteren Algorithmus, der sich aus dem Prinzip der dynamischen Programmierung herleitet.

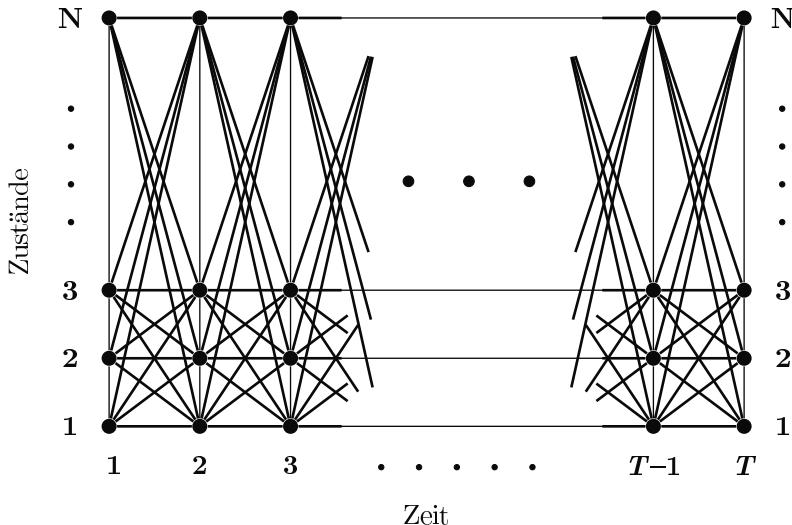


Abbildung 2.18: Gitter zur Berechnung der Beobachtungswahrscheinlichkeit

Vorwärtsprozedur: Die Idee der dynamischen Programmierung ist es, Teilsequenzen von Beobachtungen zu berechnen und möglichst oft wiederzuverwenden. Größere Teilsequenzen lassen sich aus kleineren Teilsequenzen zusammensetzen. Dazu definieren wir die sogenannte Vorwärtsvariable

$$\alpha_t(j) = P(o_1 \dots o_t, q_t = s_j | \boldsymbol{\lambda}).$$

$\alpha_t(j)$ gibt für ein gegebenes Modell $\boldsymbol{\lambda}$ die Wahrscheinlichkeit an, die Beobachtungsfolge $o_1 \dots o_t$ zu sehen und dabei im Zustand s_j zu enden. Wir können $\alpha_t(j)$ iterativ berechnen:

1. Initialisierung: Für $1 \leq j \leq N$ sei

$$\alpha_1(j) = \pi_j \cdot b_j(o_1) \quad (2.67)$$

2. Iteration: Für $1 \leq t < T$, $1 \leq j \leq N$ sei

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^N \alpha_t(i) \cdot a_{ij} \right) \cdot b_j(o_{t+1}) \quad (2.68)$$

3. Terminierung:

$$P(\mathbf{O}|\boldsymbol{\lambda}) = \sum_{j=1}^N \alpha_T(j) \quad (2.69)$$

Die Initialisierung (2.67) ergibt sich durch einfaches Einsetzen: $\alpha_1(j) = P(o_1, q_1 = s_j | \boldsymbol{\lambda})$ ist die Wahrscheinlichkeit, direkt im Zustand s_j zu starten und dort das Symbol o_1 auszugeben, also $\pi_j \cdot b_j(o_1)$. Der entscheidende Iterationsschritt (2.68) kann anhand von Abbildung 2.19 nachvollzogen werden. Nach

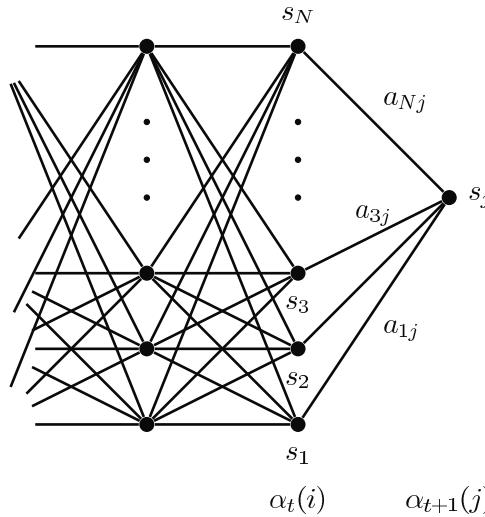


Abbildung 2.19: Iterationsschritt bei der Vorwärtsprozedur

Schritt t sind die Vorwärtsvariablen $\alpha_t(i)$ für alle Zustände s_1 bis s_N bekannt. Sie summieren jeweils die Wahrscheinlichkeiten aller Pfade, die zum Zeitpunkt t im entsprechenden Zustand s_i enden. Ausgehend von einem dieser Zustände gehen wir mit Wahrscheinlichkeit a_{ij} zum Zustand s_j über. Die Wahrscheinlichkeit, die Symbole $o_1 \dots o_t$ gesehen zu haben und zum Zeitpunkt $t + 1$ im Zustand s_j zu landen, beträgt also $\sum_{i=1}^N \alpha_t(i) \cdot a_{ij}$. Im Anschluss daran wird das Symbol o_{t+1} mit Wahrscheinlichkeit $b_j(o_{t+1})$ ausgegeben. Durch Multiplizieren ergibt sich der Wert der Vorwärtsvariablen $\alpha_{t+1}(j)$ gemäß Gleichung (2.68). Bei

der Berechnung greifen wir auf die bereits bekannten Vorwärtsvariablen zurück, wodurch der Rechenaufwand wesentlich geringer wird.

Wenn wir bei der Iteration in Zeitschritt T angekommen sind, kennen wir mit den Vorwärtsvariablen $\alpha_T(1)$ bis $\alpha_T(N)$ die Wahrscheinlichkeiten, die gesamte Symbolfolge $\mathbf{O} = o_1 \dots o_T$ gesehen zu haben und in einem der Zustände s_1 bis s_N zu landen. Bei der Berechnung von $P(\mathbf{O}|\boldsymbol{\lambda})$ spielt es keine Rolle, in welchem der Endzustände wir landen. Wir erhalten also durch Summation das gewünschte Resultat gemäß Gleichung (2.69).

Die Initialisierung beansprucht N Multiplikationen, jeder Iterationsschritt $N(N + 1)$ Multiplikationen und $N(N - 1)$ Additionen. Da wir $T - 1$ solcher Iterationsschritte und im letzten Schritt noch N Additionen benötigen, ergeben sich also $N + N(N + 1)(T - 1)$ Multiplikationen und $N(N - 1)(T - 1) + N$ Additionen. Für $N = 5, T = 100$ benötigen wir nur noch 4960 Operationen statt etwa 10^{72} vorher. Wenn man die relevanten Faktoren bei der Vorwärtsprozedur mit der sogenannten \mathcal{O} -Notation (Bronstein und Semendjajew 1987) abschätzt, ergibt sich ein Aufwand von $\mathcal{O}(N^2T)$ Operationen, gegenüber $\mathcal{O}(N^T)$ beim naiven Algorithmus.

Rückwärtsprozedur als Spiegelung: Wir können $P(\mathbf{O}|\boldsymbol{\lambda})$ bei gleichem Rechenaufwand auch mit der sogenannten Rückwärtsprozedur berechnen, die man als Spiegelung der Vorwärtsprozedur betrachten kann. Beide Algorithmen werden für die Lösung der weiteren Probleme benötigt. Wir definieren also in analoger Weise zur Vorwärtsvariablen $\alpha_t(j)$ die sogenannte Rückwärtsvariable

$$\beta_t(i) = P(o_{t+1} \dots o_T, q_t = s_i | \boldsymbol{\lambda}).$$

Wir gehen also davon aus, dass sich das Modell $\boldsymbol{\lambda}$ zum Zeitpunkt t im Zustand s_i befindet. Dann bezeichnet $\beta_t(i)$ die Wahrscheinlichkeit, dass die Symbolfolge $o_{t+1} \dots o_T$ ausgegeben wird. Die Rückwärtsvariablen können wir wieder iterativ berechnen:

1. Initialisierung: Für $1 \leq i \leq N$ sei

$$\beta_T(i) = 1 \tag{2.70}$$

2. Iteration: Für $t = T - 1, T - 2, \dots, 1$ und $1 \leq i \leq N$ sei

$$\beta_t(i) = \sum_{j=1}^N a_{ij} \cdot b_j(o_{t+1}) \cdot \beta_{t+1}(j) \tag{2.71}$$

3. Terminierung:

$$P(\mathbf{O}|\boldsymbol{\lambda}) = \sum_{j=1}^N \pi_j \cdot b_j(o_1) \cdot \beta_1(j) \tag{2.72}$$

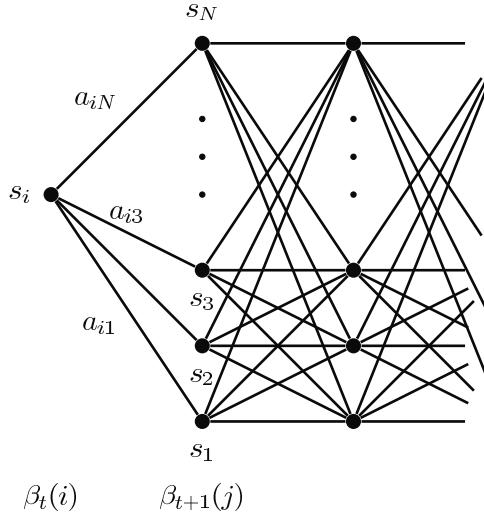


Abbildung 2.20: Iterationsschritt bei der Rückwärtsprozedur

Nach Zeitpunkt T wird sicher kein Symbol mehr ausgegeben, egal in welchem Zustand wir uns befinden. Für die Initialisierung (2.70) gilt also $\beta_T(i) = 1$. Der Iterationsschritt (2.71) wird in Abbildung 2.20 veranschaulicht. Die Iteration stützt sich wieder auf die vorher berechneten Werte, bei der Terminierung (2.72) müssen zusätzlich noch die Startwahrscheinlichkeiten berücksichtigt werden. Die Berechnung von $P(\mathbf{O}|\boldsymbol{\lambda})$ über die Rückwärtsvariablen $\beta_t(i)$ beansprucht wie bei der Vorwärtsprozedur $\mathcal{O}(N^2T)$ Operationen.

Optimale Zustandsfolge

Welches ist die optimale Folge von Zuständen im Modell $\boldsymbol{\lambda}$ bei gegebener Beobachtung $\mathbf{O} = o_1 \dots o_T$? Wir wollen den verborgenen Teil des Modells aufdecken. Die Frage ist aber nicht eindeutig zu klären, weil es verschiedene Bewertungsmöglichkeiten dafür gibt, was man unter „optimal“ versteht. Ein mögliches Kriterium wäre es, die Sequenz mit den wahrscheinlichsten *Einzelzuständen* auszuwählen. Hierbei kann es aber zu Problemen kommen, weil die wahrscheinlichsten Zustände möglicherweise gar nicht untereinander erreichbar sind, diese Folge also in Wirklichkeit nicht auftreten kann. Um das Problem zu umgehen, könnte man an Stelle der besten Einzelzustände die besten Paare von Zuständen in die Sequenz aufnehmen. Auch die wahrscheinlichsten Tripel oder n -Tupel von Zuständen wären denkbar.

Der wohl nächstliegende Ansatz ist es aber, die wahrscheinlichste *Gesamtfolge* von Zuständen auszuwählen. Der Rechenaufwand ist zwar höher als bei den vorgenannten Kriterien, aber es gibt mit dem **Viterbi-Algorithmus** (Viterbi 1967;

Forney 1973) eine elegante und effiziente Lösung. Wir suchen also bei gegebenem Modell λ und gegebener Beobachtung $\mathbf{O} = o_1 \dots o_T$ nach der wahrscheinlichsten Gesamtsequenz \mathbf{q}^* , d.h.

$$\mathbf{q}^* = \arg \max_{\mathbf{q} \in Q_T} P(\mathbf{q} | \lambda, \mathbf{O}).$$

Wie bei der Berechnung von Beobachtungswahrscheinlichkeiten könnte man wieder einfach alle Zustandsfolgen durchprobieren und die beste heraussuchen. Aber auch hier ist es wesentlich günstiger, den Ansatz der dynamischen Programmierung zu verwenden. Wir bestimmen den wahrscheinlichsten Gesamtpfad auf ähnliche Weise wie bei der Vorwärtsprozedur, indem wir auf bereits berechnete Teilstufen zurückgreifen. Außerdem nutzen wir aus, dass nach der Formel von Bayes folgender Zusammenhang gilt:

$$P(\mathbf{q} | \lambda, \mathbf{O}) = \frac{P(\mathbf{O} | \lambda, \mathbf{q}) \cdot P(\mathbf{q} | \lambda)}{P(\mathbf{O} | \lambda)} = \frac{P(\mathbf{q}, \mathbf{O} | \lambda)}{P(\mathbf{O} | \lambda)}.$$

Da der Nenner $P(\mathbf{O} | \lambda)$ unabhängig von \mathbf{q} ist, können wir an Stelle von $P(\mathbf{q} | \lambda, \mathbf{O})$ auch die gemeinsame Wahrscheinlichkeit $P(\mathbf{q}, \mathbf{O} | \lambda)$ bezüglich \mathbf{q} maximieren, um den besten Pfad \mathbf{q}^* zu bestimmen:

$$\mathbf{q}^* = \arg \max_{\mathbf{q} \in Q_T} P(\mathbf{q}, \mathbf{O} | \lambda). \quad (2.73)$$

Bei gegebenem Modell λ sei $\delta_t(j)$ die höchste Wahrscheinlichkeit, mit einem Teilstufenpfad $q_1 \dots q_t$, der im Zustand $q_t = s_j$ endet, die Symbole $o_1 \dots o_t$ zu erzeugen. $\delta_t(j)$ wird also durch Maximierung über alle geeigneten Pfade $q_1 \dots q_t$ berechnet:

$$\delta_t(j) = \max_{q_1 \dots q_t} P(q_1 \dots q_t, q_t = s_j, o_1 \dots o_t | \lambda)$$

Wir verwenden außerdem eine sogenannte **Rückverfolgungsmatrix**

$$\psi = [\psi_t(j)]_{(T-1) \times N},$$

um den jeweils letzten Schritt des besten Teilstufenpfads $q_1 \dots q_t$ zu speichern. Diese Matrix hilft uns schließlich, den wahrscheinlichsten Gesamtpfad $\mathbf{q}^* = q_1^* \dots q_T^*$ zu bestimmen. Der Algorithmus läuft nun wie folgt ab:

1. Initialisierung: Für $1 \leq j \leq N$ sei

$$\delta_1(j) = \pi_j \cdot b_j(o_1) \quad (2.74)$$

2. Iteration: Für $1 \leq t < T$ und $1 \leq j \leq N$ sei

$$\delta_{t+1}(j) = \left(\max_i \delta_t(i) \cdot a_{ij} \right) \cdot b_j(o_{t+1}) \quad (2.75a)$$

$$\psi_t(j) = \arg \max_i \delta_t(i) \cdot a_{ij} \quad (2.75b)$$

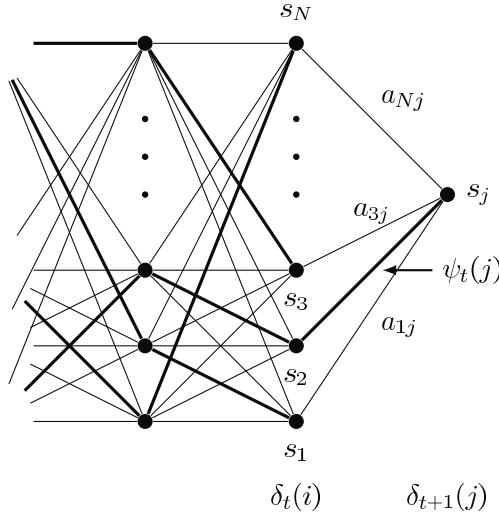


Abbildung 2.21: Iterationsschritt beim Viterbi-Algorithmus. Die fett eingezeichneten Linien zeigen den jeweils „besten“ Pfad, dessen Wahrscheinlichkeit in $\delta_t(i)$ bzw. $\delta_{t+1}(j)$ gespeichert ist und der sich aus der Rückverfolgungsmatrix $\psi_t(j)$ rekonstruieren lässt.

3. Terminierung der Iteration:

$$P^*(\mathbf{O}|\boldsymbol{\lambda}) = \max_i \delta_T(i) \quad \text{und} \quad q_T^* = \arg \max_i \delta_T(i) \quad (2.76)$$

4. Pfadrückverfolgung: Für $t = T - 1, T - 2, \dots, 1$ sei

$$q_t^* = \psi_t(q_{t+1}^*) \quad (2.77)$$

Der erste Schritt (2.74) ergibt sich wieder durch einfaches Einsetzen: $\delta_1(j) = \max_{q_1} P(q_1, q_1 = s_j, o_1 | \boldsymbol{\lambda})$ ist die höchste Wahrscheinlichkeit, mit einem Teilpfad der Länge 1, der in Zustand s_j endet, das Symbol o_1 zu erzeugen. Da nur ein solcher Teilpfad existiert (nämlich $q_1 = s_j$), ergibt sich für $\delta_1(j)$ sofort die Wahrscheinlichkeit $\pi_j \cdot b_j(o_1)$, genau wie bei der Vorwärtsprozedur.

Der entscheidende Iterationsschritt (2.75) ist in Abbildung 2.21 nachvollziehbar. Die Werte $\delta_t(i)$ sind für alle Zustände s_1 bis s_N bekannt und geben die höchste Wahrscheinlichkeit eines Teilpfads an, der zum Zeitpunkt t in s_i endet und dabei die Symbolfolge $o_1 \dots o_t$ erzeugt. Die entsprechenden Teilpfade sind in Abbildung 2.21 fett eingezeichnet. Ausgehend von einem dieser Zustände gehen wir mit Wahrscheinlichkeit a_{ij} zum Zustand s_j über. Die größte Wahrscheinlichkeit, die Symbolfolge $o_1 \dots o_t$ gesehen zu haben und zum Zeitpunkt $t + 1$ im Zustand s_j zu landen, ist also $\max_i \delta_t(i) \cdot a_{ij}$. Im Anschluss daran wird das

Symbol o_{t+1} mit Wahrscheinlichkeit $b_j(o_{t+1})$ ausgegeben. Durch Multiplizieren ergibt sich somit der Wert von $\delta_{t+1}(j)$ gemäß Gleichung (2.75a). In $\psi_t(j)$ wird für den Zustand s_j und Zeitpunkt $t + 1$ genau der Vorgängerzustand gespeichert, für den in (2.75a) das Maximum angenommen wird, der also das letzte Teilstück des besten Pfades nach s_j bildet. Dies reicht aus, um am Schluss durch Pfadrückverfolgung den besten Gesamtpfad zu rekonstruieren.

Wenn wir bei der Iteration in Zeitschritt T angekommen sind, kennen wir mit den Werten $\delta_T(1)$ bis $\delta_T(N)$ die jeweils größte der Wahrscheinlichkeiten, beim Gesamtpfad $q_1 \dots q_T$ die Gesamtsequenz $\mathbf{O} = o_1 \dots o_T$ gesehen zu haben und in einem der Zustände s_1 bis s_N zu landen. Wir speichern die größte Wahrscheinlichkeit nach Gleichung (2.76) in $P^*(\mathbf{O}|\boldsymbol{\lambda})$. Den entsprechenden Zustandsindex merken wir uns in q_T^* . Mittels Pfadrückverfolgung (2.77) ist dann die gesuchte Zustandsfolge $\mathbf{q}^* = q_1^* \dots q_T^*$ einfach zu bestimmen. Der Gesamtaufwand für die Ermittlung des optimalen Pfades beträgt $\mathcal{O}(N^2T)$ Operationen.

Für $P(\mathbf{O}|\boldsymbol{\lambda})$, der summierten Wahrscheinlichkeit aller Pfade, wurde im letzten Abschnitt schon ein effizientes Verfahren vorgestellt. $P^*(\mathbf{O}|\boldsymbol{\lambda})$ ist üblicherweise stark mit diesem Wert korreliert und kann, wie soeben dargestellt, ebenfalls effizient berechnet werden. In Spracherkennern wird häufig $P^*(\mathbf{O}|\boldsymbol{\lambda})$ statt $P(\mathbf{O}|\boldsymbol{\lambda})$ zur Bestimmung einer Maßzahl verwendet, die angibt, wie gut eine Äußerung \mathbf{O} zu einem Modell $\boldsymbol{\lambda}$ passt. Der Viterbi-Algorithmus ist praktisch deckungsgleich mit dem **DTW-Verfahren** (Schukat-Talamazzini 1995, 133), dessen Name sich von engl. *dynamic time warping* (dynamische Zeitverzerrung) ableitet. Beim DTW-Verfahren wird allerdings mit nur einer einzigen Referenzbeobachtung verglichen, beim Viterbi-Algorithmus dagegen mit einem trainierbaren HMM, das viele Beobachtungen repräsentiert. Der HMM-Ansatz ist also flexibler. Ein Trainingsverfahren, das ein HMM-Modell an eine Menge von Beobachtungsfolgen anpasst, wird im folgenden Abschnitt beschrieben.

Parameter-Optimierung

Das letzte und schwierigste Problem beschäftigt sich mit der Anpassung der Modellparameter (Start-, Übergangs- und Ausgabewahrscheinlichkeiten) eines Hidden-Markov-Modells an eine gegebene Beobachtungssequenz. Dieser Vorgang wird analog zur Terminologie bei künstlichen Neuronalen Netzen und anderen maschinellen Lernverfahren auch „Training“ des Modells genannt. Tatsächlich kann sogar ein HMM als Neuronales Netz aufgefasst und als solches trainiert werden (Bridle 1990). Das Training eines HMM lässt sich also als die Konstruktion eines Modells interpretieren, das am besten die modellierte natürliche Signalquelle simuliert.

Nach dem Maximum-Likelihood-Prinzip gilt es, das folgende Problem zu lösen: „Bestimme $\boldsymbol{\lambda} = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$ so, dass $P(\mathbf{O}|\boldsymbol{\lambda})$ maximal wird.“ Die zu optimierende Größe $P(\mathbf{O}|\boldsymbol{\lambda})$ wird hier als eine Funktion des Parameters $\boldsymbol{\lambda}$ aufgefasst und als **Likelihood-Funktion** bezeichnet.

In vielen Anwendungen ist allerdings eine einzelne Beobachtungssequenz \mathbf{O} nicht ausreichend. Vielmehr muss das HMM an eine Folge $\mathbf{O}^{(1)}, \dots, \mathbf{O}^{(M)}$ von Beobachtungen angepasst werden, z. B. Aussprachevarianten des Wortes „ha-

ben“. Da wir davon ausgehen können, dass die einzelnen Beobachtungen $\mathbf{O}^{(m)}$ voneinander unabhängig sind, besteht die Likelihood-Funktion aus dem Produkt ihrer Wahrscheinlichkeiten. Wir erhalten damit als optimales Modell

$$\boldsymbol{\lambda}^* = \arg \max_{\boldsymbol{\lambda}} \prod_{m=1}^M P(\mathbf{O}^{(m)} | \boldsymbol{\lambda}). \quad (2.78)$$

Wir beschreiben hier das einfache Trainingsverfahren für eine einzelne Beobachtungssequenz \mathbf{O} . Es kann problemlos auf eine Folge von Beobachtungen, d.h. die in Gleichung (2.78) ausgedrückte Situation, erweitert werden.

Für die Parameter-Optimierung durch Maximierung der Likelihood-Funktion $P(\mathbf{O} | \boldsymbol{\lambda})$ existiert keine analytische Lösung. Stattdessen können allgemeine Optimierungsverfahren wie stochastische Optimierung (*simulated annealing*, genetische Algorithmen usw.) oder lokale Optimierungsverfahren (z. B. Gradientenabstiegsverfahren, *gradient descent*) angewendet werden.

Im Folgenden soll der **Baum-Welch-Algorithmus** (Baum und Petie 1966; Baum und Eagon 1967) beschrieben werden, ein speziell an HMM-Modelle angepasstes lokales Optimierungsverfahren, das auch als Gradientenabstiegsverfahren verstanden werden kann (Baum und Sell 1968). Das Verfahren verbessert das Modell $\boldsymbol{\lambda}$ sukzessive und erzeugt so eine Folge von Modellen $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \dots$; laut (Baum, Petie, Soules und Weiss 1972) existiert eine konvergente Teilfolge, sofern die Likelihood-Funktion nur endlich viele lokale Optima besitzt. Da der Algorithmus auf die oben beschriebenen Vorwärts- und Rückwärtsprozeduren zurückgreift, wird er in der Literatur oft auch **Forward-Backward-Algorithmus** genannt.

Sei $\xi_t(i, j) = P(q_t = s_i, q_{t+1} = s_j | \mathbf{O}, \boldsymbol{\lambda})$ die Wahrscheinlichkeit, dass sich das HMM bei gegebener Beobachtungssequenz \mathbf{O} zur Zeit t im Zustand s_i und zur Zeit $t + 1$ im Zustand s_j befindet. Um diese Wahrscheinlichkeit über die Vorwärts- bzw. Rückwärtsvariablen auszudrücken, formen wir zunächst um:

$$\begin{aligned} \xi_t(i, j) &= P(q_t = s_i, q_{t+1} = s_j | \mathbf{O}, \boldsymbol{\lambda}) \\ &= \frac{P(q_t = s_i, q_{t+1} = s_j, \mathbf{O} | \boldsymbol{\lambda})}{P(\mathbf{O} | \boldsymbol{\lambda})} \\ &= \frac{P(q_t = s_i, q_{t+1} = s_j, \mathbf{O} | \boldsymbol{\lambda})}{\sum_{i=1}^N \sum_{j=1}^N P(q_t = s_i, q_{t+1} = s_j, \mathbf{O} | \boldsymbol{\lambda})} \end{aligned}$$

Beim letzten Umformungsschritt beachte man, dass der Nenner äquivalent zu einer Summierung über alle möglichen Zustandsfolgen $q_1 \dots q_T$ ist.

Die Vorwärtsvariable $\alpha_t(t)$ ist laut Definition die Wahrscheinlichkeit, die Symbolfolge $o_1 \dots o_t$ auszugeben und dabei im Zustand s_i zu enden. Die entsprechende Rückwärtsvariable $\beta_t(i)$ ist die Wahrscheinlichkeit, dass vom Zustand s_i zum Zeitpunkt t ausgehend die Symbolfolge $o_{t+1} \dots o_T$ produziert wird.

Damit das in $P(q_t = s_i, q_{t+1} = s_j, \mathbf{O} | \boldsymbol{\lambda})$ beschriebene Ereignis eintritt, muss die Symbolfolge $o_1 \dots o_t$ ausgegeben werden und der Prozess zunächst in Zustand s_i enden (mit Wahrscheinlichkeit $\alpha_t(i)$), dann der Übergang von Zustand s_i nach s_j erfolgen (mit Wahrscheinlichkeit a_{ij}), dort das Symbol o_{t+1} ausgegeben

werden (mit Wahrscheinlichkeit $b_j(o_{t+1})$) und abschließend noch die Symbolfolge $o_{t+2} \dots o_T$ produziert werden (mit Wahrscheinlichkeit $\beta_{t+1}(j)$):

$$P(q_t = s_i, q_{t+1} = s_j, \mathbf{O} | \boldsymbol{\lambda}) = \alpha_t(i) \cdot a_{ij} \cdot b_j(o_{t+1}) \cdot \beta_{t+1}(j).$$

Wir können damit $\xi_t(i, j)$ wie folgt schreiben:

$$\xi_t(i, j) = \frac{\alpha_t(i) \cdot a_{ij} \cdot b_j(o_{t+1}) \cdot \beta_{t+1}(j)}{\sum_{i=1}^N \sum_{j=1}^N \alpha_t(i) \cdot a_{ij} \cdot b_j(o_{t+1}) \cdot \beta_{t+1}(j)}.$$

Man nennt $\xi_t(i, j)$ auch „Kombinationsereignis“. Die Summe $\sum_{t=1}^{T-1} \xi_t(i, j)$ ist dann die erwartete Anzahl von Transitionen von s_i nach s_j , da wir uns durch die Summation vom Zeitpunkt des Übergangs unabhängig machen.

Sei ferner $\gamma_t(i) = P(q_t = s_i | \mathbf{O}, \boldsymbol{\lambda})$ die Wahrscheinlichkeit, zur Zeit t im Zustand s_i zu sein. Es gilt offensichtlich der Zusammenhang

$$\gamma_t(i) = \sum_{j=1}^N \xi_t(i, j)$$

und $\sum_{t=1}^{T-1} \gamma_t(i)$ ist die erwartete Anzahl von Transitionen ausgehend von s_i . Nun können wir eine iterative Methode zur Verbesserung der Modellparameter $\boldsymbol{\lambda} = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}) \rightsquigarrow \bar{\boldsymbol{\lambda}} = (\bar{\mathbf{A}}, \bar{\mathbf{B}}, \bar{\boldsymbol{\pi}})$ angeben (Baum und Petrie 1966; Baum und Eagon 1967):

- $\bar{\pi}_i$ wird abgeschätzt als die erwartete Aufenthaltswahrscheinlichkeit im Zustand s_i zu Beginn ($t = 1$), also

$$\bar{\pi}_i = \gamma_1(i) = \sum_{j=1}^N \xi_1(i, j)$$

- \bar{a}_{ij} wird neu bestimmt als die erwartete Anzahl von Transitionen von s_i nach s_j in Relation zu allen erwarteten Transitionen ausgehend von s_i :

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{j=1}^N \sum_{t=1}^{T-1} \xi_t(i, j)} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

- $\bar{b}_j(k)$ schätzt man ab als die erwartete Anzahl von Transitionen ausgehend von s_j mit Ausgabe v_k in Relation zu allen erwarteten Transitionen ausgehend von s_j , d.h.

$$\bar{b}_j(k) = \frac{\sum_{t=1}^T \gamma_t(j) \cdot \chi_{[o_t=v_k]}}{\sum_{t=1}^T \gamma_t(j)}$$

Die charakteristische Funktion $\chi_{[\cdot]}$ in der letzten Formel nimmt immer dann den Wert 1 an, wenn der Ausdruck in eckigen Klammern erfüllt ist, ansonsten 0. Es

kann gezeigt werden, dass man bei iterativer Anwendung der obigen Prozedur immer einen Fixpunkt erreicht, sofern nur endlich viele Fixpunkte existieren (Baum et al. 1972). Erstmals bewiesen wurde dies in (Baum und Sell 1968).

In der Regel wird ein **lokales Optimum** gefunden, das nicht gleich dem gesuchten **globalen Optimum** ist. Die Likelihood-Funktion eines HMM ist üblicherweise so komplex, dass viele lokale Optima existieren. Eine Abhilfe besteht darin, mehrere unabhängige Trainingsläufe mit zufälligen Startwerten der Modellparameter durchzuführen und das beste Modell λ aus allen Trainingsläufen auszuwählen.

Der Baum-Welch-Algorithmus überführt in jedem Iterationsschritt das aktuelle Modell $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ in ein verbessertes Modell $\bar{\lambda} = (\bar{\mathbf{A}}, \bar{\mathbf{B}}, \bar{\pi})$. Es handelt sich bei diesem Verfahren um einen Spezialfall des allgemeineren **EM-Algorithmus** (von engl. *expectation maximization*), der erstmals von Dempster, Laird und Rubin (1977) beschrieben wurde. In Rabiner und Juang (1993) oder Schukat-Talamazzini (1995) finden sich sehr gute Herleitungen der Baum-Welch-Schätzformeln aus dem EM-Prinzip.

Der Rechenaufwand eines Optimierungsschritts nach Baum-Welch beträgt $\mathcal{O}(N^2T)$ Operationen für die Bestimmung von $\xi_t(i, j)$, sowie $\mathcal{O}(N)$, $\mathcal{O}(N^2T)$ und $\mathcal{O}(NKT)$ Operationen für die verbesserten Parameterschätzwerte. Ein schnelleres Verfahren für dieselbe Aufgabe ist das **Viterbi-Training**, das auf dem Viterbi-Algorithmus aufbaut. Die damit erzielten Ergebnisse reichen an die des Baum-Welch-Trainings heran, sofern ausreichend Trainingsmaterial zur Verfügung steht (Merhav und Ephraim 1991). In jedem Schritt verbessert es das Modell λ bezüglich der Viterbi-Bewertung $P^*(\mathbf{O}|\lambda)$ anstelle der Likelihood-Funktion $P(\mathbf{O}|\lambda)$, d.h. es wird nur der beste Pfad des Modells bei der Verbesserung betrachtet. Dazu müssen gegenüber dem Baum-Welch-Algorithmus lediglich folgende Parameter anders definiert werden:

$$\xi_t(i, j) = \chi_{[q_t^* = s_i, q_{t+1}^* = s_j]} \quad \text{und} \quad \gamma_t(i) = \chi_{[q_t^* = s_i]}.$$

Statt $\xi_t(i, j) = P(q_t = s_i, q_{t+1} = s_j | \mathbf{O}, \lambda)$, also der Wahrscheinlichkeit, zur Zeit t im Zustand s_i und zur Zeit $t+1$ im Zustand s_j zu sein, wird nun genau dann der Wert 1 verwendet, wenn s_i und s_j zum Zeitpunkt t bzw. $t+1$ auf dem optimalen Pfad liegen, ansonsten 0. Analog wird mit $\gamma_t(i) = P(q_t = s_i | \mathbf{O}, \lambda)$ verfahren, also der Wahrscheinlichkeit, zur Zeit t im Zustand s_i zu sein. Man setzt $\gamma_t(i)$ genau dann auf 1, wenn s_i zum Zeitpunkt t auf dem besten Pfad liegt, ansonsten auf 0.

Das Viterbi-Training kann auch als eine entscheidungsüberwachte Variante des EM-Algorithmus für HMMs aufgefasst werden. Die Rechengeschwindigkeit des Verfahrens wird durch die kleinen Modifikationen des Algorithmus erheblich gesteigert, weshalb es sehr oft in aktuellen Spracherkennern verwendet wird. Wir geben das Viterbi-Trainingsverfahren hier in kurzer Form wieder.

Nach Festlegung der K Ausgabesymbole und der Anzahl der Zustände N der zu trainierenden HMMs werden alle Modelle mit geeigneten Parametern \mathbf{A} , \mathbf{B} und π initialisiert (z.B. Gleichverteilungen). Nun wird eine Beobachtungssequenz $\mathbf{O} = o_1 \dots o_T$ aus dem Lernmaterial gewählt und das zugehörige HMM verbessert, indem folgende Schleife wiederholt wird:

1. Bestimme den optimalen Pfad \mathbf{q}^* mit dem Viterbi-Algorithmus gemäß

$$\mathbf{q}^* = \arg \max_{\mathbf{q} \in Q_T} P(\mathbf{q}, \mathbf{O} | \boldsymbol{\lambda}).$$

2. Berechne zu \mathbf{q}^* die Start-, Übergangs- und Ausgabehäufigkeiten

$$\widehat{\pi}_i = \chi_{[q_1=s_i]}, \quad \widehat{a}_{ij} = \sum_{t=1}^{T-1} \chi_{[q_t^*=s_i, q_{t+1}^*=s_j]} \quad \text{und} \quad \widehat{b}_j(k) = \sum_{t=1}^T \chi_{[q_t^*=s_j, o_t=v_k]}.$$

3. Normiere diese zu neuen Parameterschätzwerten:

$$\bar{\pi}_i = \frac{\widehat{\pi}_i}{\sum_i \widehat{\pi}_i}, \quad \bar{a}_{ij} = \frac{\widehat{a}_{ij}}{\sum_j \widehat{a}_{ij}} \quad \text{und} \quad \bar{b}_j(k) = \frac{\widehat{b}_j(k)}{\sum_k \widehat{b}_j(k)}.$$

4. Setze $\boldsymbol{\lambda} = (\bar{\mathbf{A}}, \bar{\mathbf{B}}, \bar{\boldsymbol{\pi}})$.

Der Aufwand der einzelnen Berechnungen beträgt $\mathcal{O}(N^2T)$ Operationen für den optimalen Pfad, sowie $\mathcal{O}(N)$, $\mathcal{O}(N^2+T)$ und $\mathcal{O}(NK+T)$ Operationen für die verbesserten Parameterschätzwerte. Das Viterbi-Training hat also im wesentlichen die gleiche quadratische Komplexität wie das Baum-Welch-Training, allerdings bei einem wesentlich geringeren konstanten Faktor.

Die oben beschriebene Schleife kann beendet werden, wenn sich das Modell $\boldsymbol{\lambda}$ kaum noch verbessert, oder nachdem eine feste Anzahl von Wiederholungen ausgeführt wurde. In der Regel wird dasselbe Modell $\boldsymbol{\lambda}$ mit verschiedenen Beobachtungssequenzen $\mathbf{O} = o_1 \dots o_T$ abwechselnd trainiert und der Erfolg des Trainings an einer Testmenge überprüft. Die Erkennung der Testmenge sollte sich zunächst verbessern. Wenn das Modell $\boldsymbol{\lambda}$ zu sehr an die Trainingsmenge angepasst wurde, dann wird der Fehler auf der Testmenge ansteigen. Das ist der Zeitpunkt, an dem das Gesamttraining beendet wird, weil das Auswendiglernen der Trainingsmenge die Generalisierungsfähigkeit des Modells beeinträchtigt.

2.4.3 Evaluation und Optimierung statistischer Modelle

Die im vorigen Abschnitt eingeführten HMMs können als stochastische Prozesse aufgefasst werden, die verborgene Variablen \mathbf{q} und sichtbare Variablen \mathbf{O} sukzessive mit zufällig ausgewählten Werten belegen, wobei die Wahrscheinlichkeitsverteilung in jedem Schritt durch die bereits zugewiesenen Werte und die Modellparameter bestimmt ist. Man bezeichnet statistische Modelle dieser Form daher als **generative Modelle**, zu denen u.a. auch probabilistische kontextfreie Grammatiken (PCFG, siehe Unterkapitel 3.5) und ein als Naive Bayes-Klassifikation bekanntes maschinelles Lernverfahren gehören. Bei den HMMs bildet \mathbf{q} eine zu der Beobachtungssequenz \mathbf{O} parallele Folge verborgener Zustände; bei PCFGs formen die verborgenen Variablen \mathbf{q} einen Strukturbau; und bei dem Naive Bayes-Klassifikationsverfahren steht eine einzelne verborgene Variable q für die unbekannte Kategorie eines Objekts, während \mathbf{O} eine ungeordnete Menge von beobachtbaren Merkmalen darstellt.

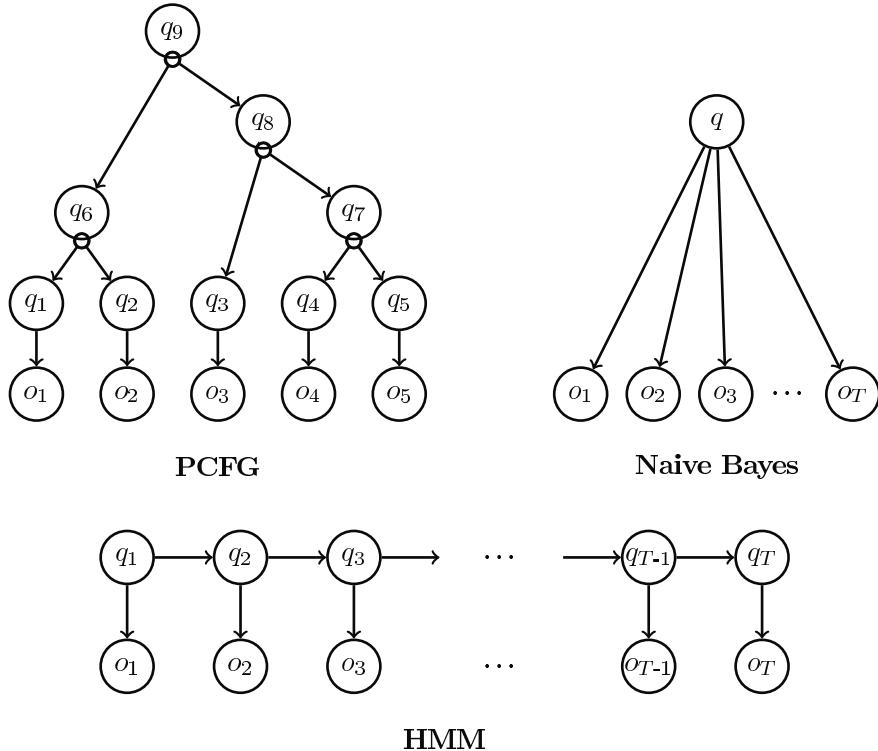


Abbildung 2.22: Diagramm-Darstellung ausgewählter generativer Modelle

Generative Modelle können sehr anschaulich in Form von Diagrammen wiedergegeben werden, die Abhängigkeitsbeziehungen zwischen den verborgenen und sichtbaren Variablen beschreiben (Abbildung 2.22). Der einem Modell λ zugrunde liegende stochastische Prozess induziert eine gemeinsame Wahrscheinlichkeitsverteilung $P(\mathbf{q}, \mathbf{O}|\lambda)$ der verborgenen und sichtbaren Variablen. Aus Abbildung 2.22 kann unmittelbar die spezielle Form dieser Modellverteilung abgelesen werden:

$$\begin{aligned}
P(\mathbf{q}, \mathbf{O} | \boldsymbol{\lambda}_{\text{PCFG}}) &= P(q_9) \cdot P(q_6, q_8 | q_9) \cdot P(q_1, q_2 | q_6) \\
&\quad \cdot P(q_3, q_7 | q_8) \cdot P(q_4, q_5 | q_7) \cdot \prod_{t=1}^5 P(o_t | q_t) \\
P(\mathbf{q}, \mathbf{O} | \boldsymbol{\lambda}_{\text{NB}}) &= P(q) \cdot \prod_{t=1}^T P(o_t | q) \\
P(\mathbf{q}, \mathbf{O} | \boldsymbol{\lambda}_{\text{HMM}}) &= P(q_1) \cdot P(o_1 | q_1) \cdot \prod_{t=1}^{T-1} P(q_{t+1} | q_t) \cdot P(o_{t+1} | q_{t+1}) \\
&= \pi_{q_1} \cdot b_{q_1}(o_1) \cdot \prod_{t=1}^{T-1} a_{q_t q_{t+1}} \cdot b_{q_{t+1}}(o_{t+1})
\end{aligned}$$

Die letzte Zeile zeigt, dass die aus Abbildung 2.22 gewonnene Formel für die HMM-Wahrscheinlichkeit $P(\mathbf{q}, \mathbf{O} | \boldsymbol{\lambda}_{\text{HMM}})$ zu der in Abschnitt 2.4.2 hergeleiteten Formel (2.65) äquivalent ist.

Generative statistische Modelle werden in der Computerlinguistik häufig zur automatischen Annotierung von Texten mit linguistischen Merkmalen herangezogen. Bekannte Beispiele sind die **Textklassifikation** (Naive Bayes-Verfahren, u.a. bei Spam-Filtern verbreitet), die Annotierung auf Wortebene oder **Tagging** (HMM, siehe Unterkapitel 3.4) und die syntaktische Annotierung oder **Parsing** (PCFG, siehe Unterkapitel 3.5). Bei diesen Anwendungen steht die Beobachtungssequenz \mathbf{O} jeweils für einen Eingabesatz oder ein Textdokument, während die verborgenen Variablen \mathbf{q} die zu bestimmenden linguistischen Merkmale repräsentieren. Zur Annotierung gilt es, die optimale Belegung

$$\mathbf{q}^* = \arg \max_{\mathbf{q} \in Q_T} P(\mathbf{q} | \boldsymbol{\lambda}, \mathbf{O})$$

der verborgenen Variablen zu bestimmen. Für ein HMM geschieht dies z. B. mit dem Viterbi-Algorithmus aus Abschnitt 2.4.2.

Bei der Anwendung von HMMs in der Spracherkennung kann ein **unüberwachtes Training** ohne Kenntnis der verborgenen Zustände durchgeführt werden, da hier nur die Beobachtungswahrscheinlichkeit $P(\mathbf{O} | \boldsymbol{\lambda})$ von Interesse ist (siehe Abschnitt 2.4.2). Für die automatische Annotierung ist hingegen eine genaue Modellierung der bedingten Wahrscheinlichkeit $P(\mathbf{q} | \boldsymbol{\lambda}, \mathbf{O})$ entscheidend. Nur durch **überwachtes Training** auf manuell annotierten Texten kann dem Computermodell die notwendige Information über die gewünschten linguistischen Merkmale vermittelt werden.

Die folgenden Unterabschnitte behandeln das überwachte Training, die Evaluation und die Optimierung von generativen Modellen. Als Beispiel dient uns hierfür die Wortartenannotierung mit Hilfe von HMMs. Die eingeführten Methoden und Begriffe lassen sich aber unmittelbar auf weitere Tagging-Anwendungen und andere Typen generativer Modelle übertragen.

Manuelle Annotierung und überwachtes Training

Ziel der automatischen **Wortartenannotierung** (auch **POS-Tagging**, von engl. *part of speech*) ist es herauszufinden, ob z. B. die Wortform *gleichen* als Adjektiv (*die gleichen Beispiele*) oder als Verb (*Kaninchen gleichen Hasen*) gebraucht wird. Tatsächlich trifft man meist wesentlich feinere Unterscheidungen (z. B. Infinitiv vs. Indikativ vs. Imperativ bei Verben), die durch ein sogenanntes **Tagset** festgelegt werden. Das bekannteste Tagset für deutsche Texte ist das Stuttgart-Tübingen Tagset oder **STTS** (Schiller, Teufel und Stückert 1999), das wir in den folgenden Beispielen zugrunde legen. Abbildung 2.23 zeigt den Beispielsatz *Gut schmeckt der Kohl heuer nicht*, wobei für jedes Wort alle möglichen Wortarten in Form von STTS-Tags angegeben sind. Aufgrund der Großschreibung

					ADJA
					ADJD
					ADV
		ART			VVFIN
ADJD	VVFIN	PDS	NE	VVIMP	
NN	VVIMP	PRELS	NN	...	PTKNEG
<i>Gut</i>	<i>schmeckt</i>	<i>der</i>	<i>Kohl</i>	<i>heuer</i>	<i>nicht</i>

Abbildung 2.23: Beispiel für die automatische Wortartenannotierung

bung am Satzanfang kann es sich bei *Gut* sowohl um ein Substantiv (NN) als auch um ein prädikativ gebrauchtes Adjektiv (ADJD) handeln; *schmeckt* kann Indikativ (VVFIN) oder Imperativ (*schmeckt!*, VVIMP) sein; die Wortform *der* kann als Artikel (ART), Demonstrativpronomen (PDS) oder Relativpronomen (PRELS) verwendet werden; *Kohl* kann Substantiv (NN) oder Eigenname (NE) sein; nur bei *nicht* handelt es sich eindeutig um eine Negationspartikel (PTKNEG). Der süddeutsche Ausdruck *heuer* ist dem Computer nicht bekannt, daher muss jede mögliche Wortart in Betracht gezogen werden. Ausgenommen bleiben lediglich Tags für Funktionswörter wie Artikel, Pronomina und Präpositionen: man geht davon aus, dass diese vollständig im Lexikon des Taggers aufgeführt sind. Außerdem können anhand der Kleinschreibung die Tags NN und NE ausgeschlossen werden. Die korrekten Wortarten sind in Abbildung 2.23 fett gedruckt. Sie werden von gängigen HMM-Taggern (Schmid 1995; Brants 2000b) fehlerfrei annotiert.

Ausgangspunkt für das überwachte Training ist ein manuell mit Wortarten annotiertes **Referenzkorpus** (engl. *gold standard*). Formal handelt es sich dabei um eine Folge von Paaren $(\mathbf{O}^{(m)}, \mathbf{q}^{(m)})$ für $m = 1, \dots, M$, wobei $\mathbf{O}^{(m)}$ jeweils einen Satz, d.h. eine Beobachtungssequenz von Wörtern, und $\mathbf{q}^{(m)}$ die zugehörige Folge manuell zugewiesener POS-Tags darstellt. Die verborgenen Zustände des HMM entsprechen in unserem Beispiel also den 54 Wortarten des STTS-Tagsets ($N = 54$), die Ausgabesymbole der im Prinzip unbeschränkten Menge aller deutscher Wortformen (d.h. für K lässt sich kein fester Wert angeben).

Geeignete Referenzkorpora in der Größenordnung von einer Million Wörtern Text sind u.a. für Englisch mit der Penn Treebank (Marcus et al. 1993) und für Deutsch mit der TIGER-Baumbank (Brants et al. 2002) verfügbar, siehe Unterkapitel 4.2. Sofern ein umfangreiches Lexikon vorliegt, können bei geeigneten Optimierungen aber bereits mit nur 20000 Wörtern manuell annotiertem Text als Referenzkorpus sehr gute Ergebnisse erzielt werden (Schmid 1995).

Zur Bestimmung der Modellparameter wenden wir nun wiederum das Maximum-Likelihood-Prinzip an. Die Trainingsprozedur kann erheblich vereinfacht werden, indem wir nicht die bedingte Wahrscheinlichkeit $P(\mathbf{q}|\boldsymbol{\lambda}, \mathbf{O})$ optimieren sondern die gemeinsame Wahrscheinlichkeit $P(\mathbf{q}, \mathbf{O}|\boldsymbol{\lambda})$, die ja für die Bestimmung der optimalen Belegung \mathbf{q}^* ausreichend war. Wie in Abschnitt 2.4.2 nehmen wir an, dass die zum Training verwendeten Sätze voneinander unabhängig sind. Die optimalen Modellparameter ergeben sich somit aus der Gleichung

$$\boldsymbol{\lambda}^* = \arg \max_{\boldsymbol{\lambda}} \prod_{m=1}^M P(\mathbf{q}^{(m)}, \mathbf{O}^{(m)} | \boldsymbol{\lambda}). \quad (2.79)$$

Nach einigen Umformungen, auf die wir hier nicht näher eingehen können, erhält man (wieder unter Zuhilfenahme der Gibbs-Ungleichung) geschlossene Formeln für die optimalen Modellparameter, die als **MLE-Schätzwerte** (für engl. *maximum likelihood estimate*) bekannt sind:

$$\pi_i = \frac{f(q_1 = s_i)}{M}, \quad a_{ij} = \frac{f(s_i s_j)}{\sum_j f(s_i s_j)} \quad \text{und} \quad b_j(k) = \frac{f(s_j, v_k)}{\sum_k f(s_j, v_k)}. \quad (2.80)$$

π_i ist dabei der Anteil von Sätzen, die mit Wortart s_i beginnen, an allen M Sätzen des Referenzkorpus; $f(s_i s_j)$ die Anzahl der Übergänge von s_i nach s_j , also die Häufigkeit des Wortarten-Bigramms $s_i s_j$; der Nenner $\sum_j f(s_i s_j)$ entspricht der gesamten Anzahl aller Übergänge von s_i aus, d.h. der Häufigkeit der Wortart s_i im Referenzkorpus; $f(s_j, v_k)$ ist die Anzahl der Vorkommen von Wortform v_k , die mit der Wortart s_j getaggt sind; und $\sum_k f(s_j, v_k)$ gibt erneut die Gesamthäufigkeit der Wortart s_j an (in Kombination mit einer beliebigen Wortform). Wir setzen hier nicht einfach $f(s_i)$ bzw. $f(s_j)$ in den Nenner ein, um sicherzustellen, dass die HMM-Parameter alle erforderlichen Normierungsbedingungen erfüllen. In der Tat berechnen die beiden Summenformeln leicht unterschiedliche Wortartenhäufigkeiten: in der Formel für a_{ij} wird das jeweils letzte Wort eines Satzes, von dem kein Zustandsübergang mehr stattfindet, nicht mitgezählt.

Das überwachte Training erfordert im Gegensatz zum unüberwachten Training kein iteratives Verfahren (vgl. Abschnitt 2.4.2). Es genügt ein einfaches Auszählen der Korpushäufigkeiten, aus denen direkt die optimalen Modellparameter nach Gleichung (2.80) berechnet werden können. Diese Effizienz wurde in erster Linie durch den Trick erzielt, in Gleichung (2.79) die gemeinsame und nicht die bedingte Wahrscheinlichkeit zu maximieren.

Es hat sich bald gezeigt (Church 1988; Schmid 1994), dass die einfachen HMM-Modelle aus Abschnitt 2.4.2, die bei der Berechnung der Übergangswahrscheinlichkeiten nur den jeweils letzten Zustand berücksichtigen, nicht ausreichen, um

lokale syntaktische Muster wie z. B. die typische Abfolge von Wortarten in einer Präpositionalphrase (Präposition, optionaler Artikel, optionale Adjektive, Substantiv) zu modellieren. Solche einfachen HMMs werden auch als Bigramm-HMMs bezeichnet, da ihre Übergangswahrscheinlichkeiten nach Gleichung (2.80) aus Bigramm-Häufigkeiten geschätzt werden.

Stattdessen werden bei aktuellen POS-Taggern **Trigramm-HMMs** eingesetzt, bei denen Übergangswahrscheinlichkeiten jeweils durch die letzten zwei Zustände bedingt sind. Statt der $N \times N$ -Matrix $\mathbf{A} = [a_{ij}]$ erhalten wir also eine dreidimensionale $N \times N \times N$ -Struktur von Übergangswahrscheinlichkeiten

$$a_{hij} = P(q_{t+2} = s_j \mid q_t = s_h, q_{t+1} = s_i).$$

Darüber hinaus müssen die Startwahrscheinlichkeiten auf Kombinationen der ersten beiden Zustände erweitert werden,

$$\pi_{ij} = P(q_1 = s_i, q_2 = s_j),$$

so dass die Formel für die gemeinsame Wahrscheinlichkeit des Trigramm-HMM (analog zu Gleichung (2.65) für ein Bigramm-HMM) folgendermaßen lautet:

$$P(\mathbf{q}, \mathbf{O} | \boldsymbol{\lambda}) = \pi_{q_1 q_2} \cdot b_{q_1}(o_1) \cdot b_{q_2}(o_2) \cdot \prod_{t=1}^{T-2} a_{q_t q_{t+1} q_{t+2}} \cdot b_{q_{t+2}}(o_{t+2}). \quad (2.81)$$

Die MLE-Schätzwerte für die neuen Parameter des Trigramm-HMM sind

$$\pi_{ij} = \frac{f(q_1 = s_i, q_2 = s_j)}{M} \quad \text{und} \quad a_{hij} = \frac{f(s_h s_i s_j)}{\sum_j f(s_h s_i s_j)}, \quad (2.82)$$

die Formel für $b_j(k)$ bleibt unverändert. Vorwärtsprozedur, Viterbi-Algorithmus und andere in Abschnitt 2.4.2 beschriebene Algorithmen können ohne Schwierigkeiten an die Trigramm-HMMs angepasst werden.

Wir haben bislang eine wichtige Frage ausgeklammert: Wie zuverlässig ist die manuelle Annotierung des Referenzkorpus? Wir wollen schließlich vermeiden, dass das HMM nur lernt, Fehler der Annotatoren zu reproduzieren. Da wir nicht wissen können, was die tatsächlich korrekten POS-Tags wären (dies kann ja nur durch einen menschlichen Annotator festgelegt werden), müssen wir die Frage nach der Zuverlässigkeit auf einem indirekten Weg beantworten. Dazu wird das Referenzkorpus (oder eine Zufallsstichprobe daraus) von zwei (oder mehr) Annotatoren unabhängig voneinander bearbeitet und im Anschluss die **Übereinstimmung** (oft **IAA**, für engl. *inter-annotator agreement*) zwischen den Annotatoren berechnet. Bei einer guten Übereinstimmung geht man davon aus, dass die Annotationen auch korrekt sind und somit die erforderliche Zuverlässigkeit gewährleistet ist.

Entscheidend für eine hohe Zuverlässigkeit sind eine genaue Beschreibung des Tagsets sowie detaillierte **Richtlinien** (engl. *annotation guidelines*), die Beispiele diskutieren und Entscheidungshilfen für Zweifelsfälle geben. So konnte bei der Wortartenannotierung der TIGER-Baumbank beispielsweise eine Übereinstimmung von 98.57% zwischen zwei unabhängigen Annotatoren erzielt werden

(Brants 2000a). Dieser Wert gehört zu den besten Übereinstimmungen bei der manuellen Annotierung linguistischer Information. Für syntaktische Analysen in der TIGER-Baumbank liegt die Übereinstimmung nur bei ca. 93%, für subjektive Entscheidungen (u.a. in der Semantik) sogar deutlich unter 90%.

Grundsätzlich muss bei der Interpretation solcher IAA-Studien zwischen **zufälligen** und **systematischen** Annotierungsfehlern unterschieden werden. Zufällige Fehler entstehen z. B. durch Unaufmerksamkeit eines Annotators. Sie führen in der Regel zu Diskrepanzen zwischen zwei unabhängigen Annotatoren, werden also bei der Berechnung der Übereinstimmung mit erfasst. Bei bestimmten Annotierungsaufgaben kann es allerdings häufig vorkommen, dass beide Annotatoren zufällig den gleichen Fehler machen, der somit unbemerkt bleibt.

Beispiel 2.4.6

Man kann sich diese Situation anhand eines einfachen Beispiels veranschaulichen. Dazu nehmen wir an, dass die Annotatoren lediglich eine Unterscheidung zwischen zwei Möglichkeiten treffen sollen, z. B. ob es sich bei einer großgeschriebenen Wortform um ein Substantiv (NN) oder einen Eigennamen (NE) handelt. Weisen nun beide Annotatoren rein zufällige Tags zu, ohne sich die Wortformen und ihre Kontexte anzusehen, so werden sie dennoch bei 50% aller Wörter übereinstimmen: in $0.5 \cdot 0.5 = 25\%$ der Fälle entscheiden sich zufällig beide für NN, in weiteren 25% beide für NE. Die zufällige Übereinstimmung kann noch wesentlich höher ausfallen, wenn die Kategorien ungleichmäßig verteilt sind. Markieren etwa beide Annotatoren nur 5% aller Wortformen als Eigennamen (treffen dabei aber weiterhin in jedem Einzelfall eine rein zufällige Entscheidung), so steigt die zufällige Übereinstimmung auf 90.5% (lediglich $0.05 \cdot 0.05 = 0.25\%$ für NE stehen $0.95 \cdot 0.95 = 90.25\%$ für NN gegenüber). Hier wäre also eine Übereinstimmung von 93% sicher kein Zeichen für eine zuverlässige Annotierung. □

Es gibt verschiedene Ansätze, die berechnete Übereinstimmung um solche Zufallseffekte zu bereinigen. In der Computerlinguistik ist vor allem der **Kappa-Koeffizient** (Cohen 1960; Carletta 1996) gebräuchlich, der die Differenz zwischen tatsächlicher Übereinstimmung p_o (engl. *observed agreement*) und zufälliger Übereinstimmung p_c (engl. *chance agreement*) folgendermaßen normiert:

$$\kappa = \frac{p_o - p_c}{1 - p_c}.$$

Mit dieser Definition entspricht $\kappa = 1$ stets einer perfekten Übereinstimmung, und $\kappa = 0$ rein zufälligen Entscheidungen der beiden Annotatoren. Im obigen Beispiel würde die auf den ersten Blick sehr gute Übereinstimmung von $p_o = 93\%$ bei $p_c = 90.5\%$ nur einem Kappa-Koeffizienten von $\kappa \approx 0.26$ entsprechen. Eine gute Zuverlässigkeit der Annotierung liegt aber erst bei $\kappa \geq 0.8$ vor.

Aus Platzgründen ist es hier nicht möglich, näher auf die Berechnung des Kappa-Koeffizienten und seine Interpretation einzugehen. Eine detaillierte Anleitung für verschiedene Varianten von Kappa und ein hervorragender Überblick über die einschlägige Literatur finden sich bei Artstein und Poesio (2008).

Schwerer als zufällige Fehler wiegen die **systematischen Annotierungsfehler**, die u.a. auf eine ungenaue Beschreibung des Tagsets oder eine unterschiedli-

che Interpretation der Richtlinien durch die beiden Annotatoren zurückzuführen sind. Eine häufige Fehlerquelle bei der Wortartenannotierung ist beispielsweise die Abgrenzung zwischen Substantiven (NN) und Eigennamen (NE), die in der Studie von Brants (2000a) für 21.5% der beobachteten Diskrepanzen verantwortlich ist.

Neuere Untersuchungen deuten darauf hin, dass zufällige Fehler nur geringe Auswirkungen auf das überwachte Training statistischer Modelle und maschinelner Lernverfahren haben, während systematische Fehler zu einer erheblichen Verschlechterung führen (Reidsma und Carletta 2008). Im Idealfall sollte ein Referenzkorpus daher vollständig von zwei oder mehr unabhängigen Annotatoren bearbeitet werden. Anschließend diskutieren die Annotatoren jede aufgetretene Diskrepanz und treffen (falls möglich) eine einvernehmliche Entscheidung. Im Rahmen dieses Arbeitsprozesses werden viele Quellen systematischer Fehler aufgedeckt und können durch Verbesserung der Richtlinien vermieden werden.

Evaluation und Optimierung

Im Anschluss an das überwachte Training kann im Prinzip die erreichte Anpassung des Modells an das Referenzkorpus durch Berechnung der KL-Divergenz nach Definition 2.4.5 bestimmt werden. Dieser Wert lässt aber nur indirekt Rückschlüsse auf die im praktischen Einsatz erzielte Tagging-Qualität zu. Daher ist eine empirische **Evaluation** des trainierten POS-Taggers unerlässlich. Für diesen Zweck wird der Tagger auf eine Teilmenge des Referenzkorpus angewendet. Durch Vergleich der automatisch zugewiesenen Wortarten mit der manuellen Annotation kann die **Genauigkeit** (engl. *accuracy*) des Verfahrens berechnet werden. Gängige HMM-Tagger erzielen dabei Werte zwischen 96.7% für Englisch (Brants 2000b) und 97.5% für Deutsch (Schmid 1995).

Bei einer solchen Evaluation ist darauf zu achten, dass die zur Berechnung der Genauigkeit herangezogenen Korpusteile nicht zum Training des Taggers eingesetzt werden. Ansonsten könnte ein Lernverfahren einfach alle im Training gesehenen Sätze mit ihren korrekten Wortarten abspeichern und später reproduzieren. Ein solcher Tagger würde bei der Evaluation eine perfekte Genauigkeit von 100% erzielen, bei der Anwendung auf neuen Text aber sehr schlechte oder überhaupt keine Resultate liefern. Man spricht in einem solchen Fall von einer **Überanpassung** (engl. *overtraining*) des statistischen Modells.

Eine korrekte Evaluation teilt daher das Referenzkorpus zunächst in ein **Trainingskorpus** und ein **Testkorpus** auf. Die Parameter des statistischen Modells werden anhand des Trainingskorpus bestimmt. Anschließend wird das trainierte Modell durch Berechnung der auf dem Testkorpus erzielten Genauigkeit evaluiert. Bisweilen wird auch die entsprechende Genauigkeit für das Trainingskorpus berechnet. Die Differenz der beiden Werte dient dann als Maß für die Überanpassung des Modells.

Um sämtliche Daten für die Evaluation nutzen zu können, wird in der Regel eine **Kreuzvalidierung** (engl. *cross validation*) durchgeführt. Dabei wird das Referenzkorpus in meist 10 gleich große Teile zerlegt. Nun kann das statistische Modell auf jedem Korpusteil evaluiert werden, wobei jeweils die restlichen neun

Korpusteile als Trainingskorpus dienen. Zum Schluss werden die berechneten Genauigkeitswerte für alle 10 Teile gemittelt. Gleichzeitig bildet die Standardabweichung ein Maß für die Stabilität der Evaluationsergebnisse. Die Aufteilung des Referenzkorpus kann entweder zufällig erfolgen, oder durch Zerlegung in 10 zusammenhängende Abschnitte. Letzteres Vorgehen liefert eine realistischere Einschätzung der tatsächlichen Tagging-Qualität, insbesondere wenn das Referenzkorpus aus unterschiedlichen Textsorten besteht.

Oft ist ein Globalwert für die Tagging-Genauigkeit nicht ausreichend: man möchte auch wissen, wie zuverlässig eine bestimmte Wortart (z. B. Eigennamen) erkannt wird. Hierzu können drei gängige Evaluationsmaße aus dem Information Retrieval herangezogen werden. Der **Recall** R gibt an, wie oft z. B. ein Eigename vom Tagger als solcher erkannt wird. Die **Precision** P sagt aus, wie häufig ein vom Tagger als Eigename identifiziertes Wort tatsächlich ein Eigename ist. Zur Berechnung dieser Maße benötigen wir drei Zahlenwerte: die Anzahl der Eigennamen, die vom Tagger korrekt erkannt wurden (TP , für *true positives*); die Anzahl der Eigennamen, die nicht vom Tagger erkannt wurden (FN , für *false negatives*); und die Anzahl der Wörter, die vom Tagger fälschlicherweise für Eigennamen gehalten wurden (FP , für *false positives*). Dann gilt

$$P = \frac{TP}{TP + FP} \quad \text{und} \quad R = \frac{TP}{TP + FN}.$$

Precision und Recall können stark voneinander abweichen. Als einheitliches Gütemaß wird daher oft das als **F-Maß** (engl. *F-score*) bekannte harmonische Mittel angegeben:

$$F = \frac{2PR}{P + R}.$$

Bei einer korrekten Evaluation wird man oft feststellen, dass ein mit den Formeln in (2.80) bzw. (2.82) trainierter Tagger eine sehr schlechte Genauigkeit erzielt. Der Grund hierfür ist im Zipfschen Gesetz zu suchen: viele plausible Wortarten-Trigramme kommen selbst in einem großen Trainingskorpus nicht vor. Dies bedeutet aber, dass die entsprechenden MLE-Parameter a_{hij} gleich 0 sind und damit jede Zustandsfolge \mathbf{q} , welche ein solches Trigramm enthält, eine Modellwahrscheinlichkeit von $P(\mathbf{q}, \mathbf{O} | \boldsymbol{\lambda}) = 0$ zugewiesen bekommt. Entsprechende Sätze im Testkorpus können also vom Tagger nicht korrekt annotiert werden.

Um solche Probleme zu vermeiden, muss sichergestellt werden, dass für alle Übergangswahrscheinlichkeiten $a_{hij} > 0$ gilt. Diesen Prozess bezeichnet man als Glättung oder **Smoothing** der Wahrscheinlichkeiten. Es genügt hierbei nicht, die Parameter $a_{hij} = 0$ auf einen kleinen Wert $\epsilon > 0$ anzuheben. Damit die Normierungsbedingung erfüllt bleibt, müssen die Werte der anderen Übergangswahrscheinlichkeiten etwas verringert werden. Ein einfaches Verfahren ist das **Laplace-** oder **Add-One-Smoothing** (Lidstone 1920), bei dem alle Korpushäufigkeiten um 1 erhöht werden. Dies führt auf die folgenden Schätzwerte für ein Trigramm-HMM:

$$\pi'_{ij} = \frac{f(q_1 = s_i, q_2 = s_j) + 1}{M + N^2} \quad \text{und} \quad a'_{hij} = \frac{f(s_h s_i s_j) + 1}{\sum_j f(s_h s_i s_j) + N}.$$

Diese einfache Methode resultiert oft in einer übermäßig starken Glättung der Parameterwerte. Beim **Add- λ -Smoothing** wird daher ein Wert $0 \leq \lambda \leq 1$ zu den Korpushäufigkeiten addiert:

$$\pi'_{ij} = \frac{f(q_1 = s_i, q_2 = s_j) + \lambda}{M + \lambda N^2} \quad \text{und} \quad a'_{hij} = \frac{f(s_h s_i s_j) + \lambda}{\sum_j f(s_h s_i s_j) + \lambda N}. \quad (2.83)$$

Für $\lambda = 0$ erhält man die MLE-Schätzwerte, für $\lambda = 1$ das relativ starke Laplace-Smoothing. Andere Werte interpolieren zwischen diesen beiden Extremen.

Es gibt zahlreiche weitere Smoothing-Verfahren, die oft mathematisch wesentlich komplexer sind. Neben **Good-Turing Smoothing** (Good 1953) sind für HMM-Tagger insbesondere **Interpolations-** und **Back-Off-Verfahren** interessant, die auf Bigramm- und Unigramm-Schätzwerte zurückgreifen, wenn für ein Trigramm nicht ausreichend viele Daten vorliegen. Einen guten und leicht verständlichen Überblick über diese Verfahren geben Jurafsky und Martin (2009, 83–122) sowie Manning und Schütze (2003, 191–228).

Herkömmliche Smoothing-Verfahren lassen sich in der Regel nicht auf die Ausgabewahrscheinlichkeiten $b_j(k)$ anwenden. Z. B. würde die zu (2.83) analoge Gleichung für Add- λ -Smoothing lauten:

$$b'_j(k) = \frac{f(s_j, v_k) + \lambda}{\sum_k f(s_j, v_k) + \lambda K}.$$

Für die Anzahl K der Ausgabesymbole lässt sich aber kein endlicher Wert angeben, da sie das gesamte Wortformeninventar der jeweiligen Sprache umfassen. Aus diesem Grund kommt die **OOV-Methode** zum Einsatz, bei der unbekannte Wortformen durch ein spezielles Symbol OOV (für engl. *out of vocabulary*) ersetzt werden. Unser Beispielsatz *Gut schmeckt der Kohl heuer nicht* würde für den Tagger, der das Wort *heuer* nicht kennt, also so aussehen: *Gut schmeckt der Kohl OOV nicht*. Um Wahrscheinlichkeiten zu schätzen, werden im Trainingskorpus alle seltenen Wortformen (die nur ein- oder zweimal vorkommen) ebenfalls durch OOV ersetzt. Hochwertige Tagger setzen ausgeklügelte Verfahren ein, die bei unbekannten Wörtern versuchen, anhand von Endung, Groß- oder Kleinschreibung und anderen Merkmalen die korrekte Wortart zu raten.

Viele Smoothing-Verfahren besitzen sogenannte **Meta-Parameter**, z. B. die Konstante λ in (2.83), für die ebenfalls optimale Werte bestimmt werden müssen. Dazu wird die Schätzung der Modellparameter wiederholt für verschiedene Werte der Meta-Parameter durchgeführt. Zum Schluss wählt man diejenigen Parameterwerte, welche das beste Evaluationsergebnis erzielt haben. Um Überanpassung zu vermeiden, ist hier eine Dreiteilung des Referenzkorpus erforderlich. Im Beispiel der Kreuzevaluation werden jeweils 8 von 10 Teilen als Trainingskorpus verwendet. Ein Teil dient als **Entwicklungskorpus** (engl. *development set*), mit dessen Hilfe die Meta-Parameter optimiert werden. Auf dem verbleibenden Teil findet eine abschließende Evaluation statt, um die Güte des resultierenden Modells zu messen.

Die in der Literatur berichteten Tagging-Genauigkeiten lassen sich nur durch ausgereifte Smoothing-Verfahren und umfangreiche Optimierung aller Meta-

Parameter erzielen. Dieser Prozess wird von Schmid (1995) und Brants (2000b) ausführlich beschrieben.

2.4.4 Literaturhinweise

Eine übersichtliche und anwendungsorientierte Einführung in die Wahrscheinlichkeitstheorie bietet Scheid (1992). Durch die Beschränkung auf größtenteils endliche Ergebnismengen kann auf die Behandlung anspruchsvollerer Grundlagen wie der Maßtheorie verzichtet werden. Stattdessen finden sich viele Beispiele aus Statistik und Kombinatorik, um die Anwendung der Wahrscheinlichkeitstheorie zu üben.

Oberhofer (1979) steigt tiefer in die Materie ein, ist jedoch an Wirtschaftswissenschaftler gerichtet, weshalb keine fortgeschrittenen mathematischen Grundlagen vorausgesetzt werden. Die gesamte erste Hälfte des Buches behandelt ausführlich diskrete Wahrscheinlichkeitsräume.

Foata und Fuchs (1999) und Pfanzagl (1988) sind an Studierende der Mathematik gerichtet, also teilweise anspruchsvoller bezüglich mathematischer Voraussetzungen und weiterführender in der Theorie. Beide zeichnen sich aber durch einen klaren Aufbau, eine übersichtliche Notation und den Verzicht auf eine Einführung in die Maßtheorie aus.

Richter (1966) ist ein klassisches mathematisches Lehrbuch der Wahrscheinlichkeitstheorie mit Einführung in die Maßtheorie, Vorstellung auch weiterführender Ergebnisse der Theorie und recht komplizierter Notation. Allerdings findet sich hier eine interessante Auseinandersetzung mit den begrifflichen Grundlagen der Wahrscheinlichkeitsrechnung, wo unter anderem zwischen intuitivem, naturwissenschaftlichem und mathematischem Wahrscheinlichkeitsbegriff differenziert wird.

Im englischsprachigen Bereich ist besonders das Lehrbuch von DeGroot und Schervish (2002) empfehlenswert. Es gibt eine sehr klare Einführung in die mathematische Theorie der Wahrscheinlichkeitsrechnung, bleibt dabei aber auch ohne umfangreiche Vorkenntnisse zugänglich. Ein Schwerpunkt dieses Buches liegt auf der Anwendung wahrscheinlichkeitstheoretischer Methoden in der mathematischen und angewandten Statistik.

Die ersten Arbeiten über HMMs stammen von Baum und seinen Kollegen (Baum und Petrie 1966; Baum und Eagon 1967; Baum und Sell 1968; Baum et al. 1972), Baker (1975) und Jelinek (1976). Sehr bekannt wurde das Verfahren durch Rabiner (1989). Einen guten Überblick über das Gebiet der Spracherkennung mit einer ausführlichen Beschreibung von HMMs bietet das weit verbreitete Buch von Rabiner und Juang (1993) und auch das deutschsprachige Standardwerk von Schukat-Talamazzini (1995). Eine kompakte Darstellung mit Schwerpunkt auf Tagging-Anwendungen findet sich in den Lehrbüchern von Jurafsky und Martin (2009, 173–192) sowie Manning und Schütze (2003, 317–360).

N-Gramm-Modelle, Smoothing-Techniken und probabilistische kontextfreie Grammatiken (PCFG) werden von Jurafsky und Martin (2009, 83–122 und 459–480) sowie Manning und Schütze (2003, 191–228 und 381–405) ausführlich behandelt.

Ein Standardwerk für Naive Bayes und andere maschinelle Lernverfahren ist das Lehrbuch von Bishop (2006), das aber z.T. mathematisch recht anspruchsvoll ist. Maschinelle Lernverfahren spielen heutzutage eine wichtige Rolle in der Computerlinguistik, da viele Anwendungen statistischer Modelle als Klassifikationsprobleme betrachtet werden können. Ein bekanntes und sehr vielseitiges Verfahren sind die Support Vector Machines (Vapnik 1995), die für POS-Tagging und zahlreiche andere computerlinguistische Aufgaben eingesetzt werden. Gute Darstellungen der zugrunde liegenden Theorie finden sich bei Bishop (2006, 291–358) und ausführlicher bei Schölkopf und Smola (2002).

In den letzten Jahren werden generative statistische Modelle zunehmend durch sog. diskriminative Modelle ersetzt, welche direkt die bedingte Verteilung $P(\mathbf{q}|\boldsymbol{\lambda}, \mathbf{O})$ modellieren und damit besser für die automatische Annotierung optimiert werden können. Ein weiterer Vorteil diskriminativer Modelle liegt in der besseren Integration vieler Informationsquellen (z. B. Wortendungen, Groß- und Kleinschreibung sowie Lexikoninformation bei der Bestimmung möglicher Wortarten für bekannte und unbekannte Wörter), auch wenn diese stark miteinander korreliert sind. Bekannte Vertreter sind Maximum-Entropy-Modelle (Berger et al. 1996) und Conditional Random Fields (Sutton und McCallum 2006). Eine sehr schöne und leicht verständliche Einführung geben Jurafsky und Martin (2009, S. 193–212).

2.5 Texttechnologische Grundlagen

Georg Rehm

Die Texttechnologie stellt ein noch junges Forschungsfeld dar, das sich mit der linguistisch motivierten Informationsanreicherung und Verarbeitung digital verfügbarer Texte mittels standardisierter Auszeichnungssprachen beschäftigt. Eine zielgerichtete Definition ist aufgrund der zahlreichen, in konkreten Anwendungen potentiell beteiligten Disziplinen – u. a. Computer- und Korpuslinguistik, Text- und Hypertext-Theorie, Text-Mining – nicht ohne weiteres möglich, d. h. die Texttechnologie umfasst nicht eine eindeutig bestimmbarer Menge aufeinander aufbauender Methoden oder Theorien, sie ist vielmehr „wissenschaftlich begründete Praxis“ (Lobin und Lemnitzer 2004a, S. 1). Als ‚kleinster gemeinsamer Nenner‘ wird in allen texttechnologischen Anwendungen die Metasprache **XML** (*Extensible Markup Language*, Bray et al. 2000) eingesetzt, die die Definition beliebiger Auszeichnungssprachen (die auch als **Markup-Sprachen** oder **XML-Anwendungen** bezeichnet werden) erlaubt. XML ist also eine formale Sprache zur Spezifizierung konkreter Markup-Sprachen, die wiederum zur *Auszeichnung* (auch: *Annotation*) arbiträrer Informationseinheiten in textuellen Daten eingesetzt werden können. In computerlinguistischen Anwendungen geht es hierbei u. a. um die Auszeichnung linguistischer Informationen in Texten, die Verwendung von XML als Datenaustauschformat zur einheitlichen Repräsentation von Ressourcen und den Einsatz einer Datengrundlage in unterschiedlichen Applikationskontexten.

Die Wurzeln der Texttechnologie liegen im Bereich der medien- und plattform-unabhängigen Textauszeichnung, die erstmals mit der Standard Generalized Markup Language (SGML, ISO 8879 1986) eine breite Verwendung gefunden hat. Das zentrale Merkmal der SGML- bzw. XML-basierten Informationsmodellierung (Lobin 2000) ist die strikte **Trennung von Form und Struktur** durch die Einführung einer Abstraktionsebene, in der ein Textfragment von einem deklarativen Etikett (etwa *ueberschrift*, *paragraph* oder *beispielsatz*) umschlossen wird, ohne dabei jedoch Textsatz- oder Layout-Anweisungen zu kodieren. Im Vordergrund steht also nicht die typographische Gestaltung einer Wortfolge, sondern die eindeutige Markierung ihrer logischen Funktion bzw. ihrer Semantik innerhalb eines spezifischen Dokumenttyps. Die *Auszeichnungselemente* – häufig schlicht *Elemente* oder *Tags* genannt – einer Markup-Sprache sind nicht in beliebiger Form kombinierbar: Eine **Dokumentgrammatik** legt explizit die *hierarchischen Kombinationsmöglichkeiten* hinsichtlich der Strukturierung von Elementen fest, weshalb die in einem annotierten Dokument enthaltenen Tags im graphentheoretischen Sinn einen Baum aufspannen (vgl. Abb. 2.24 auf S. 162). Die Aufbereitung eines XML-annotierten Textes (auch: *Dokumentinstanz*) zu einem formatierten und publizierbaren Dokument erfolgt mit XSL/XSLT (Extensible Stylesheet Language, XSL Transformations, Clark 1999). Die Auslagerung der Abbildung einzelner Tags auf korrespondierende Layout-Anweisungen legte den Grundstein für das **Cross Media** bzw. **Single Source Publishing**. Hierun-

ter versteht man die Möglichkeit, aus ein- und derselben annotierten Textquelle mit Hilfe unterschiedlicher *Style Sheets* z. B. eine für den Einsatz im WWW aufbereitete HTML-Version und eine für den Druck optimierte PDF-Version generieren zu können.

Abschnitt 2.5.1 thematisiert zunächst HTML, die *Lingua Franca* des World Wide Web, woraufhin Abschnitt 2.5.2 auf XML eingeht. Abschnitt 2.5.3 stellt unterschiedliche Verarbeitungsmethoden von XML-Instanzen vor, woraufhin Standards dargestellt werden, die XML flankieren und zusätzliche Funktionalität bereit stellen. Der sehr komplexe Standard SGML wurde mittlerweile fast vollständig von XML verdrängt, das eine Teilmenge des 1998 verabschiedeten WebSGML darstellt, wobei das Kernmerkmal von SGML – die Möglichkeit der Definition beliebiger Markup-Sprachen zur hierarchischen Strukturierung arbiträrer Informationen – beibehalten wurde. Diejenigen Eigenschaften, die die Implementierung von SGML-Prozessoren und die Verarbeitung von Instanzen unnötig komplex werden lassen, wurden aus Gründen der Vereinfachung bei der Spezifizierung von XML nicht berücksichtigt.

2.5.1 HTML – Hypertext Markup Language

Webdokumente werden mit Hilfe der **Hypertext Markup Language** (Raggett et al. 1999) ausgezeichnet, die zugleich die bekannteste Auszeichnungssprache darstellt (Unterkapitel 4.7, *Das World Wide Web*, thematisiert die Nutzung von HTML-Dokumenten in sprachtechnologischen Anwendungen). HTML 4.01 spezifiziert 93 verschiedene Elemente, so markiert z. B. das Tag `<p>` (*paragraph*) einen Absatz, und die Struktur einer Tabelle wird durch das Element `<table>` (bestehend aus *table rows*, `<tr>`, die wiederum `<td>`-Elemente, *table data cell*, enthalten) modelliert. Weitere Elemente erlauben z. B. die Auszeichnung von Überschriften, Listen und vor allem die Integration von Hyperlinks mittels des Tags `<a>` (*anchor*), wobei das Attribut `href` die URL des Dokuments enthält, auf das verwiesen wird, z. B. `JLU Gießen`. Ein wichtiger Aspekt betrifft die *Mischung unterschiedlicher Auszeichnungsebenen* (Walker 1999): HTML definiert Auszeichnungselemente für strukturelles (z. B. `<h1>`, eine Überschrift erster Stufe), logisches (``, ``) zur Markierung wichtiger bzw. sehr wichtiger Textteile), präsentationsorientiertes (`<i>`, `` für Kursiv- bzw. Fettdruck), referentielles (`<a>`) und funktionales Markup (`<object>`, zur Einbettung externer Programme).

Die Dokumenttyp-Definitionen (DTDs), d. h. die regelbasierten, formalen Definitionen, die die Namen und das Zusammenspiel von Elementen und Attributen spezifizieren, wurden bis zu HTML 4.01 mit Hilfe von SGML definiert. Im Jahr 2000 wurde erstmals eine Neuformulierung von HTML auf der Grundlage von XML vorgenommen, um die formale Kompatibilität mit dem XML-Paradigma zu gewährleisten. **XHTML** 1.0 (Pemberton 2002) ist dabei nur der erste Schritt, denn mit *Modularization of XHTML* steht mittlerweile ein Inventar zur Verfügung, das die Anwendung speziell angepasster XHTML-Vokabularen erlaubt, wie z. B. *XHTML Basic*, das für mobile Endgeräte gedacht ist, oder XHTML 1.1,

das als Grundlage für zukünftige modularisierte Versionen von XHTML dienen soll.

2.5.2 XML – Extensible Markup Language

Bereits 1994 wurde angeregt, das fest vorgeschriften, statische Inventar von HTML-Elementen durch einen flexibleren Mechanismus zu ergänzen, der die Definition beliebiger Auszeichnungssprachen und ihren Einsatz im Web-Umfeld erlaubt. Gerade die mangelnde Erweiterbarkeit von HTML war für das **World Wide Web Consortium (W3C, <http://www.w3.org>)**, das Web-bezogene Standards konzipiert und verabschiedet, der Anlass, die **Extensible Markup Language (XML, Bray et al. 2000)** zu spezifizieren, die eben diese Explizierung arbiträrer Informationen ermöglicht, indem eine strikte Trennung von Inhalt und Struktur vorgenommen wird. Dieser Strukturaspekt bezieht sich dabei in vielen XML-basierten Markup-Sprachen – im Übrigen auch in den meisten XML-Einführungstexten – vornehmlich auf logische Texteinheiten, die auf der Makroebene anzusiedeln sind, z. B. *Tabelle*, *Überschrift*, *Absatz* oder *Liste* oder einzelne Binnenstrukturen der Mikroebene, etwa die Aufspaltung einer *Postanschrift* in *Empfänger* (bestehend aus *Vorname*, *Nachname*) und *Anschrift* (*Straße*, *Hausnummer*, *Postleitzahl*, *Stadt*, *Land*). Tatsächlich sind mit XML-basierten Markup-Sprachen jedoch *beliebige Strukturen* annotierbar, im linguistischen Bereich z. B. die rhetorische Struktur eines Textes auf der Grundlage der Rhetorical Structure Theory (RST, vgl. etwa Lobin 1999a und Rehm 1999 sowie Abschnitt 3.7.1) oder die Phrasenstruktur einzelner Sätze. Abb. 2.24 zeigt mit einer **XML-Dokumentinstanz** ein derartiges Beispiel (nach Witt 1999) sowie die von den XML-Elementen aufgespannte Baumstruktur, die der syntaktischen Struktur des Satzes entspricht. Die zugehörige **Dokumenttyp-Definition (DTD)** folgt einer speziellen Syntax, die im XML-Standard festgelegt ist, und enthält im Wesentlichen die Namen von Elementen und Attributen sowie die Kombinationsmöglichkeiten von Elementen, die durch sog. *Inhaltsmodelle* spezifiziert werden.

```
<!ELEMENT s  (np, vp)          <!ELEMENT v   (#PCDATA)>
<!ELEMENT np (det, n)>        <!ELEMENT n   (#PCDATA)>
<!ELEMENT vp (v, np*)>       <!ELEMENT det (#PCDATA)>

<!ATTLIST np kasus (nom|akk|dat|gen) #REQUIRED>
<!ATTLIST v  agr   (1s|2s|3s|1p|2p|3p) #REQUIRED>
```

Die ersten drei Zeilen der DTD deklarieren, eingeleitet durch das Schlüsselwort ELEMENT, sechs **Auszeichnungselemente** sowie die jeweiligen Inhaltsmodelle; hierbei ist zu beachten, dass **s** das *Wurzelement* der DTD ist, also das äußerste Element einer Instanz sein muss. Das Element **s** muss laut Inhaltsmodell in einer Dokumentinstanz zunächst das Element **np** gefolgt von dem Element **vp** enthalten. Diese Sequenz wird durch den *Konnektor* , erzwungen; als weiterer Konnektor ist das Zeichen | erlaubt, das eine *entweder oder*-Beziehung zwischen Elementen spezifiziert. Die Anzahl der Vorkommen einzelner Elemente wird – ähnlich wie in regulären Ausdrücken – durch *Okkurrenzindikatoren* festgelegt.

Im Inhaltsmodell von **s** befinden sich keine Okkurrenzindikatoren, weshalb in einer Instanz, die nach dieser DTD annotiert wird, jeweils genau ein **np**- und ein **vp**-Element enthalten sein müssen. In der Deklaration des Elements **vp** hingegen ist durch ***** markiert, dass dem Tag **v** $0..n$ **np**-Elemente folgen dürfen (zur groben Modellierung intransitiver, transitiver und ditransitiver Verben). Die beiden weiteren Okkurrenzindikatoren sind **?** und **+**, mit denen $0..1$ bzw. $1..n$ Vorkommen spezifiziert werden können. Mit Hilfe der Klammerung einzelner Teile eines Inhaltsmodells sowie der Angabe von Okkurrenzindikatoren an diesen Untermodellen sind *komplexe Inhaltsmodelle* realisierbar.

```
<?xml version="1.0" encoding="UTF-8"?>
<s>
    <np kasus="nom">
        <det>Der</det>
        <n>Mann</n>
    </np>
    <vp>
        <v agr="3s">tritt</v>
        <np kasus="akk">
            <det>den</det>
            <n>Ball</n>
        </np>
    </vp>
</s>
```

Abbildung 2.24: Eine XML-Dokumentinstanz am Beispiel der Annotation der Phrasenstruktur des Satzes „Der Mann tritt den Ball“

Die Elemente **s**, **np** und **vp** werden auch als *Containerelemente* bezeichnet, weil sie weitere Elemente aufnehmen, also noch keine konkreten Daten enthalten. Die drei *Datenelemente* **v**, **n** und **det** enthalten hingegen die Angabe des Schlüsselworts #PCDATA (*parsed character data*), d. h. sie dürfen nur konkrete Inhalte und keine Markup-Elemente enthalten. Mit Hilfe von **Attributen** können innerhalb eines Elements zusätzliche Informationen hinterlegt werden – hierbei gilt die Faustregel, dass sich Attribute nur auf **Metadaten**, d. h. Informationen über die annotierten Daten, beziehen sollten (Maler und Andaloussi 1996). Schmidt (2003) diskutiert die wesentlichen Aspekte und Standards bei der Auszeichnung von Metadaten im World Wide Web. Neben RDF/RDFS (vgl. Abschnitt 2.5.3) geht sie auf HTML und Dublin Core ein (<http://purl.org/dc/>). In Attributlistendeklarationen (**ATTLIST**) wird zunächst festgelegt, auf welches Element sich ein Attribut bezieht. In der Beispiel-DTD werden die beiden Attribute **kasus** und **agr** definiert, die für **np** bzw. **v** gelten. Es existieren unterschiedliche Typen von Attributen, die in einer Instanz u. a. die Eingabe eines beliebigen Textes als Wert erlauben. Das Beispiel zeigt jedoch den Aufzählungstyp, bei dem innerhalb der Deklaration die erlaubten Werte spezifiziert werden. Dabei gibt **#REQUIRED**

an, dass ein Attribut obligatorisch ist (Optionalität wird durch #IMPLIED ausgedrückt). Neben den im Beispiel aufgeführten Syntaxelementen einer DTD existieren noch Entitäten, die als Platzhalter für textuelle Inhalte eingesetzt werden können. Weiterhin sind bei der DTD-Erstellung verschiedene Ausnahmen zu beachten, z. B. sind mehrdeutige Inhaltsmodelle wie ((a, c) | (a, d)) verboten, können jedoch in den meisten Fällen umformuliert werden: (a, (c | d)).

Schemasprachen

Das syntaktische Inventar zur Spezifizierung von DTDs ist relativ eingeschränkt, weshalb mit **XML Schema** eine sehr komplexe Spezifikation standardisiert wurde, die präzisere Restriktionen in Instanzen ermöglicht. XML Schema gestattet z. B. die freie Definition von Datentypen, die Typisierung von Elementen sowie die Möglichkeit, die Anzahl der Vorkommen eines Elements innerhalb eines Inhaltsmodells genauer spezifizieren zu können als dies durch die Angabe von Okkurrenzindikatoren in einer DTD geschehen kann. Ein weiterer Vorteil von XML Schema und parallel entwickelten, alternativen Schemasprachen wie Relax NG oder Schematron besteht darin, dass Schemabeschreibungen ebenfalls *in einer XML-Notation kodiert* werden, d. h. die proprietäre Syntax, die innerhalb von DTDs benutzt wird, wurde aufgegeben, so dass Schemabeschreibungen – ebenso wie XML-Instanzen – mit beliebigen XML-Werkzeugen verarbeitet werden können.

Formale Eigenschaften von XML

Die Syntax einer formalen bzw. natürlichen Sprache wird durch ein Startsymbol, Produktionsregeln, ein terminales sowie ein non-terminales Vokabular definiert (vgl. Unterkapitel 2.2). Analog zu dieser Terminologie kann das Wurzelement einer DTD als Startsymbol und die einzelnen Elementdeklarationen als Syntaxregeln bezeichnet werden; Containerelemente stellen das non-terminale Vokabular dar, Datenelemente umfassen die terminalen Symbole. Diese Analogie zu kontextfreien Grammatiken lässt sich jedoch nur in terminologischen Aspekten finden, denn DTDs und die verschiedenen Schema-Sprachen erzeugen keine linearen Ketten, sondern *hierarchisch angeordnete Baumstrukturen*, weshalb sie **reguläre Baumgrammatiken** genannt werden (Lobin 2003, Mönnich und Morawietz 2003). DTDs erlauben hierbei nicht die Benutzung abstrakter Symbole in Regeln, weshalb sie als **lokale Baumgrammatiken** bezeichnet werden.

2.5.3 Verarbeitung XML-annotierter Daten

Der wichtigste Aspekt im Umgang mit XML-Instanzen betrifft deren Verarbeitung mit einem **XML-Parser** (z. B. expat, Xerces oder MS XML), der nach dem gleichen Prinzip arbeitet wie ein Parser für eine natürliche Sprache: Die DTD fungiert als formale Grammatik, gegen die eine Dokumentinstanz als Eingabekette überprüft werden kann. Falls dies fehlerfrei abläuft, nennt man eine Instanz **valide**. Durch das Parsing können zahlreiche Fehler aufgedeckt werden,

z. B. unbekannte Element- oder Attributnamen bzw. -werte oder ungültige Elementschachtelungen. XML-Instanzen muss jedoch nicht zwangsläufig eine DTD zu Grunde liegen, sie können von einem Parser auch isoliert verarbeitet werden, wobei dann lediglich überprüft werden kann, ob das Dokument der im Standard definierten Basissyntax entspricht (öffnende Elemente beginnen mit dem Zeichen < und enden mit >, Elementschachtelungen dürfen sich nicht überlappen etc.), weshalb man von der Überprüfung auf **Wohlgeformtheit** spricht.

Betrachtung und Transformation

Die Betrachtung von XML-Instanzen ist mittlerweile mit den verbreiteten Browsern und zahlreichen frei erhältlichen Werkzeugen möglich, so bieten etwa die aktuellen Versionen von Mozilla und des Internet ExplorersTM Möglichkeiten der *Visualisierung der Baumstruktur* sowie der *integrierten XSLT-Transformation* einer Instanz nach XHTML oder der Zuordnung eines *Cascading Style Sheets* (CSS), um einzelne Elemente kontextabhängig formatieren zu können.

XSLT-Transformationen basieren auf dem Prinzip der regelgesteuerten Überführung eines XML-Dokuments, etwa in ein Präsentationsformat, zur Umstrukturierung oder, dies stellt die häufigste Anwendung dar, zur Konvertierung einer Instanz der DTD x in eine Instanz der DTD y (z. B. DocBook → XHTML, vgl. <http://www.oasis-open.org/docbook/> sowie <http://www.docbook.org>). Insbesondere für die Printausgabe wird XSL-FO (*Formatting Objects*) benutzt, das gemeinsam mit **XSLT** (Clark 1999) und **XPath** (Clark und DeRose 1999) die *Extensible Stylesheet Language* (**XSL**, Adler et al. 2001) bildet. XPath übernimmt dabei die Rolle eines Hilfsstandards, der die Navigation in einem Dokument, genauer gesagt, in dessen Baumrepräsentation, und die Auswahl abstrakter Knoten (Elemente, Attribute, Text) ermöglicht. Mit Hilfe von XPath-Ausdrücken ist es in einem **XSLT-Stylesheet** möglich, für einzelne Elemente einer Dokumentinstanz korrespondierende **Templates** (Transformationsregeln) zu implementieren. XSLT wiederum ist als eine Markup-Sprache realisiert, d. h. XSLT-Stylesheets sind zugleich XML-Instanzen und können somit – wie Schemabeschreibungen – selbst zum Gegenstand XML-basierter Verarbeitungsprozesse werden. Zur Anwendung eines Stylesheets ist ein XML-Parser nicht ausreichend, da dieser lediglich Auskunft über die Validität bzw. Wohlgeformtheit einer Instanz geben kann, weshalb ein dezidierter **XSLT-Prozessor** wie z. B. Xalan, Saxon oder Sablotron benötigt wird.

Datenstrom- vs. baumbasierte Verarbeitung

Neben XSLT werden häufig zwei weitere Paradigmen eingesetzt, um in Programmiersprachen wie JavaTM oder Perl die Verarbeitung XML-annotierter Daten zu ermöglichen: **SAX** (Simple API for XML, <http://www.saxproject.org>) erlaubt die *Verarbeitung als Datenstrom*, wobei das Auftreten eines öffnenden oder schließenden Tags oder von Fließtext spezielle Ereignisse auslösen (*event-based processing*), die durch Funktionen abgefangen werden können. Da die Instanz nicht vollständig in den Speicher eingelesen werden muss, eignet sich SAX ins-

besondere zur Verarbeitung extrem großer Datenmengen sowie für einfache Filterapplikationen. Im Gegensatz dazu erzeugt ein **DOM**-Prozessor (Document Object Model, Hors et al. 2000) eine *interne Baumrepräsentation* (*tree-based processing*). Der DOM-Standard definiert zahlreiche Methoden zur Navigation innerhalb des Baums und zur Manipulation von Knoten. Dies ermöglicht u. a. die Implementierung rekursiver Funktionen, weshalb DOM meist dann benutzt wird, wenn die Funktionalität von XSLT nicht mehr ausreicht (z. B. zur Aktualisierung einer Instanz mit Informationen, die aus einer Datenbank eingelesen werden).

Datenhaltung von XML-Instanzen

Zur Datenhaltung existieren verschiedene Möglichkeiten, von denen die häufigste die Pflege der Instanzen innerhalb des *Dateisystems* darstellt. Alternativ können *relationale Datenbanken* eingesetzt werden, wobei XML-Dokumente entweder vollständig in einer Tabelle gespeichert oder dynamisch aus mehreren Datensätzen ausgelesen, in eine Dokumentschablone integriert und anschließend exportiert werden. Die interessanteste Möglichkeit betrifft den Einsatz nativer *XML-Datenbanken*. Dieser neuartige Datenbanktyp, der eben nicht auf dem relationalen oder objektorientierten Paradigma basiert, speichert Instanzen in internen Datenstrukturen ab und unterstützt neben der Validierung zahlreiche Zugriffsmöglichkeiten, die üblicherweise auf XPath und XML Query basieren.

Flankierende XML-Standards

Um XML gruppieren sich zahlreiche, teils verabschiedete, teils noch in der Entwurfsphase befindliche W3C-Standards, die Schwerpunkte in einzelnen Anwendungsbereichen setzen. **Namespaces** erlauben z. B. die gleichzeitige Verwendung *mehrerer* Auszeichnungssprachen in *einer* XML-Instanz. Hierzu werden im Wurzelement die korrespondierenden Namensräume sowie jeweils ein Präfix deklariert, das in Elementen benutzt wird, um ein spezielles Schema zu referenzieren (z. B. <`myns:doc`>). Die XML Linking Language (**XLink**) wurde zur Verknüpfung von Instanzen, aber auch zur Verbindung von Ressourcen im WWW entworfen. Die von XLink definierten Funktionen gehen weit über die einfachen Links hinaus, die HTML realisiert, so werden z. B. *bidirektionale Links* und *Typisierungen* ermöglicht. Mit **SVG** steht eine XML-basierte Markup-Sprache zur Verfügung, mit deren Hilfe Vektorgraphiken beschrieben werden können. SVG bietet insbesondere zur Visualisierung XML-annotierter Texte interessante Möglichkeiten, so sind XSLT-gesteuerte Transformationen von Textinstanzen in SVG-Instanzen möglich, die auch mit dynamischen Navigationselementen versehen werden können. Der Bereich der **Web Services**, die die Web-gestützte Kommunikation von Applikationen mittels **SOAP** (Simple Object Access Protocol) und **WSDL** (Web Services Description Language) spezifizieren, wird in Wolff (2003) aus computerlinguistischer Perspektive dargestellt.

In sprach- bzw. texttechnologischen Anwendungen werden häufig die Standards **RDF (Resource Description Framework**, Lassila und Swick 1999) und XML Topic Maps (XTM, Pepper und Moore 2001) zur Modellierung von *semantischen Netzen* und *Ontologien* eingesetzt (siehe Unterkapitel 4.6). RDF, konzipiert zur Annotation von Metadaten, arbeitet mit den Objekttypen *Resources* (die Entität, über die eine Aussage getroffen wird), *Properties* (die spezifische Eigenschaft der zu beschreibenden Entität) und *Statements*. Ein Statement umfasst dabei eine Ressource (Subjekt), die Eigenschaft (Prädikat) sowie den annotierten Wert (Objekt). Häufig genannte Beispiele für Metadaten – Daten über Daten – sind der Autor einer Informationsressource, deren Titel oder die assoziierte Organisation. Zur Spezifizierung einer RDF-Beschreibung kann man ein Metadatum als natürlichsprachlichen Satz formulieren, z. B. „Ora Lassila ist der Autor der Ressource <http://www.w3.org/Home/Lassila>“. Subjekt, Prädikat und Objekt dieses Satzes entsprechen dabei nicht notwendigerweise den korrespondierenden Bestandteilen eines Statements, so wird in diesem Beispiel eine Aussage über die *Resource* <http://www.w3.org/Home/Lassila> (das Subjekt der RDF-Beschreibung) getroffen, wobei die *Property* (das Prädikat) **Autor** mit dem Wert **Ora Lassila** (das Objekt) belegt wird. Abb. 2.25 zeigt dieses Beispiel in erweiterter Form. RDF spezifiziert lediglich eine Syntax zur Beschreibung benannter Eigenschaften von Ressourcen sowie assoziierter Werte. Die eigentliche Mächtigkeit wird erst durch den derzeit in der Konzeptionierung befindlichen Standard **RDF Schema (RDFS**, Brickley und Guha 2003) erreicht. RDFS gestattet die Spezifizierung von RDF-Vokabularen durch die Definition typisierter Relationen und Eigenschaften, wobei auch Vererbungen von Eigenschaften modellierbar sind, so dass letztlich semantische Netze oder Ontologien beschrieben werden können. Diese Anwendung war die eigentliche Motivation des für SGML entwickelten Standards **Topic Maps**, der mit **XML Topic Maps (XTM**, Pepper und Moore 2001) auch in einer XML-Version vorliegt. In XTM werden Knoten (*Topics*) mit Kanten (*Associations*) verbunden und u. U. mit einem Typensystem angereichert. Dieses abstrakte Wissen kann anschließend mit konkreten Instanzen (*Occurrences*) verbunden werden.

```

<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>
      <rdf:Description about="http://www.w3.org/staffId/85740">
        <v:Name>Ora Lassila</v:Name>
        <v:Email>lassila@w3.org </v:Email>
      </rdf:Description>
    </s:Creator>
  </rdf:Description>
</rdf:RDF>
```

Abbildung 2.25: Eine RDF-Beschreibung in XML-Notation

2.5.4 Texttechnologie und Computerlinguistik

Innerhalb sprachverarbeitender Anwendungen wird XML in verschiedenen Bereichen eingesetzt – die **Annotation von Korpora** ist das prominenteste Beispiel (vgl. Unterkapitel 4.1). Hierzu werden häufig die von der **Text Encoding Initiative (TEI)**, <http://www.tei-c.org>, Sperberg-McQueen und Burnard 2002) entwickelten DTDs benutzt, die die Annotation unterschiedlichster Textsorten (Gedichte, Dramen, historische Materialien, Lexika) auf verschiedenen Ebenen erlauben: Zunächst kann man verschiedene Metadaten von Dokumenten erfassen. Die zweite Ebene beinhaltet die Auszeichnung textueller Einheiten wie *Band*, *Kapitel* oder *Abschnitt*. Auf der dritten Ebene werden mit *Sätzen* oder *Wörtern* Strukturen innerhalb von Abschnitten markiert. Die vierte Ebene umfasst schließlich die Markierung *syntaktischer oder morphologischer Einheiten* (Ide und Véronis 1994, Witt 2003). Ein keinesfalls trivialer Aspekt betrifft die **Zeichensatzkodierung**: XML basiert auf **Unicode**, wodurch beliebige Alphabete repräsentierbar sind (Sasaki und Witt 2003). Generell konzentriert sich die Anwendung von XML in sprachverarbeitenden Systemen auf den Bereich der textuellen Datenbanken (Lobin 1999a) und eine Auswertung und Aufbereitung der Daten auf den angesprochenen Ebenen. Dabei geht es um die **manuelle oder automatische Annotation von Texten** (McKelvie et al. 1997, Ule und Hinrichs 2003, Ule und Müller 2003) und die **Manipulation** sowie die **Verwendung des annotierten Materials** in konkreten Anwendungsszenarien (Lobin und Lemnitzer 2004b, Mehler und Lobin 2003).

Die automatische Annotation von Daten kann in den unterschiedlichsten Anwendungsszenarien und prinzipiell mit beliebigen computerlinguistischen Methoden erfolgen (vgl. Kapitel 3). Ein denkbare Szenario betrifft das in Abb. 2.24 gezeigte Beispiel der Annotation eines Satzes mit Phrasenstruktur-Elementen. Ein syntaktischer Parser für ein Fragment des Deutschen könnte derartige Annotationen aus der internen Syntaxrepräsentation des Eingabesatzes erzeugen, die daraufhin automatisch mit annotierten Testsätzen verglichen werden können (vgl. Volk 1998). In diesem speziellen Beispiel kodiert die DTD Syntaxregeln (z. B. $VP \rightarrow V\ NP^*$), was z. B. bei der Implementierung und Evaluation von Chunk-Parsern ausgenutzt werden kann (vgl. Ule und Müller 2003, siehe auch Abschnitt 3.4.3), die auf Eingabesätzen operieren, die aus Webseiten gewonnen wurden. Zu diesem Zweck muss zunächst ein Korpus aufgebaut werden (vgl. Unterkapitel 4.7), woraufhin die Dokumente tokenisiert und Satzgrenzen ermittelt werden müssen. Ein zentraler Aspekt texttechnologischer Verfahren, die XML-Instanzen maschinell erzeugen, gründet sich in deren Validierbarkeit gegen eine DTD. Beim Parsing einer automatisch erzeugten Instanz gegen die von der DTD erlaubten Strukturen resultieren fehlerhafte Annotationsalgorithmen unmittelbar in ungültigem Markup, was wiederum die Ausgabe einer Fehlermeldung des XML-Parsers bewirkt (Rehm 1999), die zur Evaluation des Systems benutzt oder auch automatisch ausgewertet werden kann, um dynamisch neue Annotationsregeln zu generieren.

2.5.5 Literaturhinweise

Die Spezifikationen von XML sowie den beteiligten Standards pflegt das W3C (<http://www.w3.org>). <http://www.edition-w3c.de> bietet deutsche Übersetzungen an, die – mit ergänzenden Kommentaren und Beispielen versehen – auch in Buchform erhältlich sind (Mintert 2002). Wichtige Ressourcen sind <http://xml.coverpages.org> und <http://www.xml.com>. Licht in den von zahlreichen Abkürzungen gezeichneten Dschungel der Text- und Web-Technologien bringen <http://www.xml-acronym-demystifier.org> und <http://wildeesweb.com/glossary/>. Die Beiträge in Lobin (1999b) und Mehler und Lobin (2003) stellen Verknüpfungen von texttechnologischen und computerlinguistischen Methoden dar. Mit Lobin und Lemnitzer (2004b) liegt erstmals ein Band vor, der in die unterschiedlichen Facetten der Texttechnologie einführt.

3 Methoden

Kapitelherausgeber: Christian Ebert und Cornelia Ebert

In diesem Kapitel zu den Methoden der Computerlinguistik und Sprachtechnologie werden die Grundbegriffe und wichtigsten Ansätze der Computerlinguistik eingeführt, welche sich mit den großen Gebieten der theoretischen Linguistik decken. Weiter werden Techniken vorgestellt, die aus den speziellen Erfordernissen der Verarbeitung natürlicher Sprache erwachsen sind. Dabei wird zum einen von den theoretischen Grundlagen des vorangegangenen Kapitels reger Gebrauch gemacht werden, und zum anderen ein Ausblick auf Ressourcen und Anwendungen der beiden nachfolgenden Kapitel gegeben werden, bei denen die hier beschriebenen Methoden in der Praxis eingesetzt werden.

Unterkapitel 3.1 behandelt zunächst das Gebiet der **Phonetik und Phono-
logie**, jenen Teil der Sprachwissenschaft, der sich mit den Lauten der Sprache beschäftigt. Hier steht die Frage im Vordergrund, welche Laute in einer betreffenden Sprache unterscheidbar sind. Neben den Grundbegriffen und den wichtigsten Ansätzen dieses linguistischen Teilgebiets liegt ein Augenmerk auf der automatischen Analyse gesprochener Sprache im Rahmen der **Computerphonologie**. Das daran anschließende Unterkapitel 3.2 beschäftigt sich mit den Methoden der **Verarbeitung gesprochener Sprache** aus Sicht der Spracherkennung und Sprachsynthese und liefert damit die Grundlagen für die entsprechenden Anwendungskapitel.

Die **Morphologie** beschäftigt sich mit den Wörtern bzw. Wortformen der Sprache. Dieser Teilbereich der Sprachwissenschaft untersucht die Regeln, nach denen Wortformen gebildet werden können. Auch hier werden in Unterkapitel 3.3 zunächst die wichtigsten Begriffe eingeführt und schließlich Modellierungen, insbesondere mittels endlicher Automaten, diskutiert.

Mit der Satzverarbeitung beschäftigen sich Unterkapitel 3.4 und 3.5. Dabei gehen tiefgreifende computerlinguistische Analysen oft mit großem Verarbeitungsaufwand einher, weshalb sich Methoden zur **flachen Satzverarbeitung** – d.h. oberflächlich arbeitende Verfahren – als fruchtbar erwiesen haben. In Unterkapitel 3.4 werden Methoden u.a. zur Wortarterkennung (POS-Tagging) und zur Analyse von syntaktischen Teilstrukturen (Chunk-Parsing) vorgestellt. Das darauftreffende Unterkapitel **Syntax und Parsing** behandelt tiefergehende Methoden der Satzanalyse. Nach einer Einführung in die linguistischen Grundbegriffe zur Beschreibung des syntaktischen Aufbaus von Sätzen werden für die Computerlinguistik wichtige Grammatikformalismen kurz dargestellt. Im zweiten Teil beschäftigt sich dieses Unterkapitel mit dem Parsing, also der Zuweisung einer syntaktischen Struktur an eine Eingabekette. Hier werden Algorithmen für einen Chartparser, aber auch kurz Grundlagen des statistischen Parsens diskutiert.

Unterkapitel 3.6 behandelt die **Semantik**, dasjenige Teilgebiet der Sprachwissenschaft, das sich mit der Bedeutung von Sprache beschäftigt. Zunächst werden die Grundlagen der satzsemantischen Analyse anhand der weit verbreiteten Montague-Semantik vorgestellt, wobei die Bedeutung der kleinsten Bestandteile, also der Wörter, als gegeben vorausgesetzt wird. Ausgehend von der Satzsemantik wird die Diskursrepräsentationstheorie DRT, die die Bedeutung ganzer Diskurse erfassen kann, vorgestellt. Ein wichtiges Problem der Computerlinguistik stellt im Rahmen der Semantik die Verarbeitung von Mehrdeutigkeiten dar. Dieses Problem und Ansätze zur Lösung desselben mittels unterspezifizierten Repräsentationen werden im Anschluss diskutiert. Schließlich wird die Grundannahme der Satzsemantik, die Bedeutung von Wörtern als kleinste Einheit nicht weiter zu untersuchen, problematisiert und damit der Bereich der lexikalischen Semantik näher beleuchtet.

Das nächste Unterkapitel 3.7 dieses Methoden-Kapitels behandelt einen sehr heterogenen Teil der (Computer-)Linguistik, der unter dem Begriff **Pragmatik** zusammengefasst ist. In den Abschnitten dieses Unterkapitels werden verschiedene Aspekte angesprochen, die mit kontextuellen Eigenschaften der Sprachanalyse und -verarbeitung zu tun haben. Zum Beispiel wird diskutiert, wie Bezüge innerhalb eines Diskurses hergestellt werden können, welche impliziten Aussagen hinter einer Äußerung stehen und in welcher Hinsicht das Modell eines Benutzers für ein sprachverarbeitendes System relevant ist.

Unterkapitel 3.8 ist der **Textgenerierung** gewidmet, also der Erzeugung von Texten aus semantischen Repräsentationen. Die Generierung kohärenter Texte umfasst mehr als die Generierung aneinander gereihter einzelner Sätze, denn Texte sind satzübergreifend organisiert. Daher erfordert die Textgenerierung Methoden für die Planung globaler Organisationsstrukturen für Texte und entsprechende Mittel für deren sprachliche Umsetzung.

Das letzte Unterkapitel 3.9 im Methodenteil dieses Buches widmet sich einer Übersicht über die verschiedenen Programmierparadigmen und **Programmiersprachen**, die in der Computerlinguistik vornehmlich Verwendung finden.

3.1 Phonetik und Phonologie

Dafydd Gibbon

Die **Computerphonologie** und die **Computerphonetik** befassen sich mit der Modellierung und Operationalisierung linguistischer Theorien über die lautsprachlichen Formen und Strukturen der ca. 7000 Sprachen der Welt. Die lautsprachlichen Eigenschaften der Sprachen sind sehr vielfältig, ebenso die theoretischen und methodologischen Ansätze, die in der Linguistik und der Phonetik entwickelt worden sind, um diese Vielfalt zu beschreiben. Einige dieser Theorien haben eine formale Grundlage, sie werden aber in der Linguistik oft recht informell gehandhabt. In der **phonologischen** Literatur hat man es also oft mit recht informellen textuellen Beschreibungen und Visualisierungen von Formen

und Strukturen zu tun, deren formale Beschaffenheit nicht explizit gemacht wird. Eine Interpretation der Literatur zu erreichen, die computerlinguistischen Standards genügt und eine explizite Modellierung und Operationalisierung erlaubt, ist daher oft nicht einfach.

Dieser Beitrag verwendet zwar weitgehend standardsprachliche deutsche und englische Beispiele, beschränkt sich vom Anspruch her jedoch nicht darauf, sondern geht auch auf allgemeinere Aspekte von lautsprachlichen Systemen ein, die nicht nur andere Sprachen, sondern auch dialektale, soziale und stilistische **Aussprachevarianten** einzelner Sprachen betreffen. Die spontansprachlichen Eigenschaften der Varianten des Deutschen sind z.B. teilweise ‚exotisch‘ im Vergleich zur Standardsprache und gehen weit über die mit der Standardorthographie darstellbaren Zusammenhänge hinaus.

Auf die Fachliteratur wird nicht in den Hauptabschnitten des Beitrags, sondern in einem gesonderten Schlussabschnitt verwiesen.

Es hat sich in den letzten ca. dreißig Jahren herausgestellt, dass die theoretische, methodologische und empirische Vielfalt in der Phonologie und der Phonetik sich mit relativ einfachen formalen Mitteln modellieren und operationalisieren lässt: in erster Linie mit **regulären Modellen** (**endlichen Automaten** und **endlichen Transduktoren**) und mit Attribut-Wert-Strukturen. Die Teildisziplin der Computerphonologie existiert im Prinzip seit den 1970er Jahren, als erste **Intonationsmodelle** auf der Grundlage von endlichen Automaten entwickelt wurden und als auch entdeckt wurde, dass klassische phonologische Regeln sich mit endlichen Transduktoren modellieren lassen. In den 1980er Jahren kamen reguläre Silbenmodelle und die Zweiebenenmorphologie hinzu, in den 1990er Jahren dann reguläre Modelle der Optimalitätstheorie.

Seit den 1980er Jahren gehört auch die Einführung von statistisch gewichteten endlichen Automaten als **Hidden-Markov-Modelle (HMM)** in die Sprachtechnologie (s. Unterkapitel 3.2) zum weiteren Umfeld der formalen Modellierung und der computerbasierten Operationalisierung von sprachlautlichen Systemen.

In den 1980er Jahren wurden über den Bereich der regulären Modelle hinaus (und z.T. damit verbunden) Anwendungen der **Attribut-Wert-Logik** in die Computerlinguistik eingeführt, vorwiegend in der Syntax, aber auch für die Modellierung von phonetischen Merkmalen. Auch asymmetrische Markiertheitsrelationen zwischen den Werten phonologischer Attribute (Merkmale) wie bei *stimmhaft* und *stimmlos* konnten mit **defaultlogischen** und **unifikationstheoretischen** Mitteln modelliert werden.

Ziel dieses Beitrags ist es, diese wesentlichen phonetischen und phonologischen Fakten, Generalisierungen und Modellierungsstrategien so einzuführen und zu erläutern, dass die weitergehende Literatur, die am Ende des Beitrags angegeben wird, nutzbar gemacht werden kann. Zuerst soll die empirische sprachlautliche Domäne mit ihren Teildomänen Phonetik, Phonologie und Prosodie besprochen werden, um dann in den folgenden Abschnitten auf die Anwendung empirischer und formaler Methoden in diesen Teildomänen einzugehen. Besondere Aufmerksamkeit wird zum Schluß dem Bereich der lautsprachlichen Eigenschaften der Prosodie gewidmet.

3.1.1 Grundlagen der Computerphonologie

Die Lautlehre wird konventionell in drei Bereiche eingeteilt: Phonetik, Phonologie und Prosodie, die untereinander Abhängigkeiten aufweisen. Als erste Annäherung kann festgehalten werden, dass die **Phonetik** sich mit allen Details der Physiologie der Lautproduktion, der Akustik der Lautübertragung und der Physiologie der Lautrezeption meist mit experimentellen und quantitativen Methoden befasst, während die **Phonologie** sich auf die wortunterscheidenden Lauteigenschaften, die Lautstrukturen und die Relationen zwischen den Lauten und größeren Einheiten wie Morphem, Wort und Satz meist mit symbolverarbeitenden Methoden spezialisiert. Die **Prosodie** wird aus historischen Gründen oft als selbständiger Teil der Lautlehre behandelt. Es ist aber möglich, wie später in diesem Beitrag kurz gezeigt wird, eine systematische und integrative Modellierung der drei Bereiche vorzunehmen, wie sie für die Integration umfassender computerlinguistischer oder sprachtechnologischer Modelle erforderlich ist. Eine Einführung in alle Aspekte der Phonetik, Phonologie und Prosodie ist an dieser Stelle nicht möglich. Dafür wird auf den Literaturabschnitt 3.1.4 verwiesen.

Phonetik

Die Phonetik behandelt die Modellierung aller Eigenschaften von Sprachlauten und wird unterteilt nach Teildomäne und Untersuchungsmethoden. Die wichtigste Teildomänenunterscheidung basiert auf den drei Hauptphasen der Lautverarbeitung entsprechend einem einfachen Kommunikationsmodell (Abbildung 3.1): **artikulatorische Phonetik (Produktionsphonetik)**, **akustische Phonetik (Übertragungsphonetik)** sowie **auditive Phonetik (Rezeptionsphonetik)**.

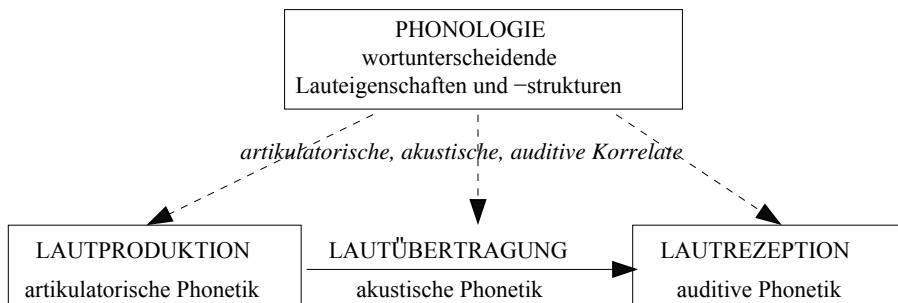


Abbildung 3.1: Korrelatrelationen zwischen der Phonologie und den phonetischen Teildomänen.

Die drei Teildomänen stellen eine hilfreiche, aber sehr vereinfachende Abstraktion dar. Die Teildomänen und die Schnittstellen zwischen ihnen sind wesentlich vielfältiger:

1. Nervensignale zwischen Gehirn und Muskeln,
2. Muskelkonfigurationen in Kehlkopf und Mundraum,
3. Gewebeoberflächenformen in Rachen und Mund,
4. Resonanzraumkonfigurationen in Rachen und Mund,
5. Akustisches Signal,
6. Transformationen im Übertragungsmedium,
7. Transformationen in den Hörorganen (Ohrkanal, Trommelfell, Gehörknöchelchen, Schneckenorgan, Gehörnerven).

Das vereinfachte Modell ist aber für Überblickszwecke nützlich und üblich.

Die drei Teildomänen der Phonetik werden an dieser Stelle überblicksweise erläutert. Für weitere Informationen steht eine reichhaltige phonetische Einführungsliteratur zur Verfügung (siehe Literaturabschnitt 3.1.4).

Laute, die in **phonologischen Kontexten** zitiert werden, sind durch Schrägstreiche gekennzeichnet, z.B. /te:/ „Tee“. Laute, die in **phonetischen Kontexten** zitiert werden, sind durch eckige Klammern gekennzeichnet, z.B. [tʰe:], um die phonetische Realisierung des /t/ mit behauchtem [tʰ] darzustellen.

Artikulatorische Phonetik

Als erste Annäherung kann die artikulatorische Phonetik in zwei Bereiche eingeteilt werden: **Schallquellen** und **Schallfilter**. Die Schallquellen bewirken die Erzeugung eines **Klangs** (eines harmonischen Lauts mit wohldefinierten Obertönen) oder eines **Geräusches** (eines Lauts ohne regelmäßige Obertonstruktur) oder einer Kombination von beiden. Die Klänge sind **Vokale** und andere stimmhafte Laute, und die **Klangeigenschaft** wird im Kehlkopf durch die rapide Schließung und Öffnung der **Glottis** (Spalte zwischen den Stimmlippen) erzeugt. Die Laute mit **Geräuschanteil** sind die Obstruenten (Plosive bzw. Verschlusslaute und Frikative bzw. Reibelaute), die durch Verschluss und Öffnung bei Zunge-Gaumen-Kontakt, Luftreibung bei Lippenverengung usw. erzeugt werden. Das Filter besteht im Wesentlichen aus zwei Resonanzräumen: dem Mund-Rachenraum und dem Resonanzraum der Nase. Die Anordnung der Artikulationsorgane und Resonanzräume wird in Abbildung 3.2 schematisch wiedergegeben.

Das hier beschriebene Modell wird **Quelle-Filter-Modell** genannt: Eine Schallquelle erzeugt einen komplexen Klang oder ein komplexes Geräusch und die **Intensität** der einzelnen Frequentanteile des komplexen Schalls (aber nicht diese Frequenzen selbst) wird dann durch ein Filter verändert (im Prinzip wie der Equaliser einer Audioanlage). Das Quelle-Filter-Modell genügt zwar nicht einer sehr präzisen akustischen Modellierung, ist aber dennoch hilfreich für ein intuitives Verständnis der Grundprinzipien der Sprachschallproduktion.

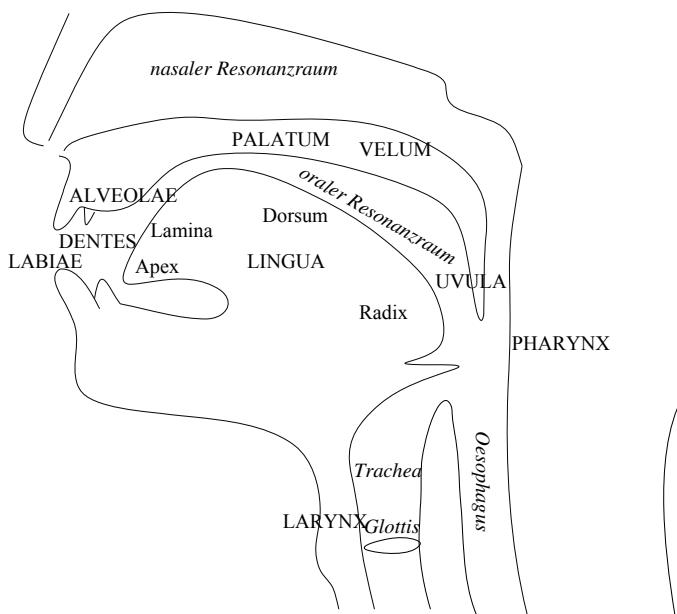


Abbildung 3.2: Schematische Darstellung des Sprechapparates. Artikulationsorgane: Labiae (Lippen), Dentes (Zähne), Lingua (Zunge), Apex (Zungenspitze), Lamina (Zungenblatt), Dorsum (Zungenrücken), Radix (Zungenwurzel), Alveolae (Zahndamm), Palatum (harter Gaumen), Velum (Gaumensegel, weicher Gaumen), Pharynx (Rachenwand), Larynx (Kehlkopf), Oesophagus (Speiseröhre), Trachea (Luftröhre). Resonanzräume: Nasenraum, Mundraum, Glottis (Stimmlippenspalte).

Der Hauptvorgang im Kehlkopf, der Schallquelle für stimmhafte Laute, ist die **Phonation**. In der normalen Phonation werden im Kehlkopf beide Stimmlippen aneinander angenähert und in der Glottis fließt der Luftstrom aus der Lunge schneller. Die Annäherung bewirkt durch den **Bernoulli-Effekt** eine Minderung des Luftdrucks zwischen den Stimmlippen, die diese aneinanderzieht (der Bernoulli-Effekt ist auch für den Auftrieb der Tragflächen von Flugzeugen, den Vorwärtstrieb von Segeln, oder auch die unerwünschte Annäherung von klammen Duschvorhängen an den Körper verantwortlich). Durch die Schließung der Glottis wird der Luftstrom blockiert, der Luftdruck steigt wieder und die Stimmlippen werden wieder auseinandergedrückt. Dieser Zyklus wiederholt sich und bestimmt die Grundfrequenz des Sprachsignals (bei Männern zwischen ca. 70 und 200 Hz, bei Frauen zwischen ca. 140 und 400 Hz, bei Kindern bis ca. 600 Hz). Während der normalen Phonation entsteht, bedingt durch die regelmäßige Schließung und Öffnung der Glottis, ein periodisches Schwingungssignal, das als Ton wahrgenommen wird.

Quelle	Filter	Beispiel
Stimmbänder	Zunge hoch, weit vorne	Vokal [i]
Stimmbänder	Zunge hoch, weit hinten	Vokal [u]
Unterlippe, obere Zähne	vorn im Ansatzrohr	Konsonant [f]
Unterlippe, obere Zähne; Stimmbänder	vorn im Ansatzrohr	Konsonant [v]
Hinterzunge, Zäpfchen	hinten im Ansatzrohr	Konsonant [r]

Tabelle 3.1: Beispiele für Quelle-Filter-Konfigurationen bei der Lautproduktion.

ßige Obertonreihe, ein Signal mit etwa sägezahnförmigem Hüllkurvenverlauf. Die Frequenzen der Obertöne bzw. der Harmonischen sind als ganzzahlige Vielfache der **Grundfrequenz** definiert.

Stimmlose Laute entstehen, wenn die Stimmlippen weit auseinandergehalten werden. Die Phonationsart Flüstern entsteht, wenn die Stimmlippen aneinandergelegt werden und nur ein kleine Öffnung am Ende bleibt. Weitere Phonationsarten sind die Knarrstimme, die Hauchstimme und die Falsettstimme.

Die zweite wichtige Schallquelle ist die Friktion (Reibung) des Luftstroms in einer engen Ritze: Wenn ein bewegliches Artikulationsorgan (z.B. die Zungenabschnitte, die Unterlippe) an ein statisches Artikulationsorgan (Oberlippe, Oberzähne, Zahndamm, Gaumen) so angelegt wird, dass nur eine sehr schmale Ritze bleibt, entsteht ein Reibungsgeräusch, die Frikative (Reibelaute) wie [f, v, s, z, ʃ, ʒ, x, h] und Affrikate wie [pf, ts] charakterisiert. Die Frikative [s, z, ʃ, ʒ] entstehen an den scharfen Zahnkanten, enthalten daher besonders hochfrequente Anteile und werden Sibilanten (Zischlaute) genannt.

Der durch eine Quelle erzeugte Schall wird in den durch die Sprechorgane geformten Resonanzräumen in Mund und Rachen gefiltert.

Die wichtigsten **Konsonanttypen** nach dem Quelle-Filtermodell werden in Tabelle 3.1 gezeigt. Bei **Konsonanten** unterscheidet man beispielsweise folgende Hauptmerkmale:

Quelle: Stimmbänder (bei stimmhaften Lauten, Klängen); Zunge oder Lippen an statisches Artikulationsorgan (bei **Geräuschen**, Verschlusslauten, Reibelauten),

Art und Weise: Verschlusslauten, z.B. [p, b, t, d, k, g], Reibelaute, z.B. Frikative [f, v, ʃ, ʒ, x, h];

Stimmhaftigkeit: Stimmlose Laute, Stimmbänder weit auseinander, nicht schwingend (stimmlose Verschluss- und Reibelaute); Stimmhafte Laute, Stimmbänder aneinandergelegt, schwingend (**Vokale**, **Nasalkonsonanten** [m, n], Liquiden [r, l], Gleitlaute [w, j]).

Bei **Vokalen** gestaltet sich das Quelle-Filter-Modell etwas einfacher: Die Quelle ist bei normalen Vokalen die Glottis mit der normalen Phonation, das Filter ist der in Form und Größe durch Zunge und Lippen variierbare Mundraum und,

bei nasalen Vokalen (und Konsonanten), der ein- und ausschaltbare und (außer durch Schnupfen) nicht variierbare Nasenraum.

Eine umfassende Notation für die artikulationsphonetischen Eigenschaften der Laute der Sprachen der Welt wird in der Symboltabelle der **Internationalen Phonetischen Assoziation (IPA)** bereitgestellt (vgl. Abbildung 3.15 auf Seite 213 am Ende dieses Beitrags). Der IPA-Tabelle liegt ein implizites und mit attribut- und defaultlogischen Mitteln formalisierbares universelles Modell von Lautobjekten zugrunde, deren Defaulteigenschaften in der **Konsonantentabelle** und dem **Vokaldiagramm** dargestellt werden. Die Defaulteigenschaften können mit diakritischen Zeichen durch speziellere Eigenschaften der Laute einzelner Sprachen überschrieben werden. Dieses Modell kann aus theoretischen wie empirischen Gründen kritisiert werden, aber es gilt mangels bewährter Alternativen als allgemein akzeptierter *de facto*-Standard und wird von den Experten der Internationalen Phonetischen Assoziation gepflegt. Das IPA wird nicht nur in der Allgemeinen Linguistik zur Beschreibung der Lautsysteme der Sprachen der Welt, sondern auch im Sprachunterricht, in der Lexikographie, in der klinischen Phonetik und in der Sprachtechnologie verwendet.

Für die im IPA enthaltenen Symbole existieren unzählige, miteinander nicht kompatible und auch nur partielle Fontimplementierungen, z.B. viele TTF-Fonts, das L^AT_EX-Font **tipa**; auch der Unicode-Standard ist nur partiell implementiert. Für die praktische Verwendung des IPA in Veröffentlichungen in der Computerphonologie, Computerphonetik und Sprachtechnologie ist daher nicht so sehr das Objektmodell des IPA, sondern die fehlende Systematisierung der bisher implementierten Fonts problematisch. Um den praktischen Datenaustausch und die einfache Verarbeitung phonetischer Daten zu ermöglichen, wurde in den 1980er Jahren im europäischen Forschungsprojekt SAM (*Speech Assessment Methods*) von Phonetikern und Sprachingenieuren eine tastaturfreundliche ASCII-Kodierung des IPA entwickelt, **SAMPA** (‘SAM Phonetic Alphabet’). Das SAMPA-Alphabet ist immer noch sehr verbreitet, weil die Unicode-Zeichenkodierung nur maschinenlesbar und eher an Druckausgabe als an Eingabeergonomie orientiert ist, weil derzeit nur unvollständige Unicode-Font-Implementierungen existieren, und auch weil Unicode die kohärente phonetische ‘Semantik’ der Zeichen nicht berücksichtigt, sondern die Verwendung von Zeichen aus verschiedenen Bereichen erfordert.¹

Akustische Phonetik

Die Schallschwingungen, die durch die artikulatorischen Quellen und Filter erzeugt wurden, werden durch die Luft und andere Medien als komplexe Druckwellen übertragen, die mit handelsüblichen Aufnahmegeräten aufgenommen werden. Das Sprachsignal wird mit den Methoden der akustischen Phonetik auf unterschiedliche Weise analysiert und dargestellt, wovon Folgende die drei wichtigsten sind:

¹Vgl. die Internetseiten von John Wells, University College, London, zu IPA-Fonts, SAMPA und Unicode.

1. Als Signal in der **Zeitdomäne**, das als **Oszillogramm** visualisiert wird, in dem die regelmäßigen **Klangabschnitte** und die unregelmäßigen **Geräuschabschnitte** des Sprachsignals relativ leicht erkennbar sind.
2. Als transformierter Signalabschnitt in der **Frequenzdomäne**, das als **Spektrum** visualisiert wird, das die **Energien** der Frequenzanteile des komplexen Signals als Funktion der Frequenz anzeigt. Am Spektrum können **Klanganteile** (**Grundfrequenz** und ihre Obertöne) sowie **Geräuschteile** des Sprachsignals erkannt werden.
3. Als dreidimensionale Darstellung von Sequenzen von zeitlich benachbarten Spektren in einem **Spektrogramm**. Am Spektrogramm können die gegenseitigen Beeinflussungen der Lautproduktionsvorgänge (Koartikulation) studiert werden.
4. Als **Grundfrequenzspur**, die die niedrigste Frequenz (**Grundfrequenz**, F0) eines Klangs als Funktion der Zeit darstellt, deren ganzzahlige Vielfache die Obertonreihe definieren.



Abbildung 3.3: Visualisierungen eines Sprachsignals (weibliche Stimme) mit der Praat-Software: Oszillogramm, Spektrogramm, Grundfrequenzverlauf (pitch track), Lautfolge in SAMPA-Kodierung.

In Abbildung 3.3 können die Vokale anhand ihrer größeren Amplitude und der regelmäßigen Struktur des Oszillogramms sowie der dunkleren Bereiche des Spektrogramms (**Formantstreifen**: Verstärkungen einzelner Frequenzbereiche durch die Filterwirkung der Resonanzräume des Artikulationstrakts) erkannt werden. Das Frikativgeräusch des [s] in *Maus* ist durch die geringere Amplitude und unregelmäßige Schwingungen im Oszillogramm sowie durch die etwas dunklere Färbung in den hochfrequenten Bereichen des Spektrogramms, ebenfalls mit Formantstreifen, zu erkennen. Im Beispiel ist auch ein Abschnitt mit Knarrstimme (die spitzen Ausschläge beim SAMPA-kodierten Diphthong [au]) zu sehen.

Für weitere Beschreibungen der zahlreichen Transformationen, Darstellungsweisen und Analyseverfahren für das Sprachsignal wird auf die Spezialliteratur im Literaturabschnitt 3.1.4 verwiesen; siehe auch Unterkapitel 3.2).

Auditive Phonetik

Die auditive Phonetik befasst sich mit den Vorgängen im Ohr und ist im Gegensatz zur artikulatorischen Phonetik nicht der direkten Beobachtung zugänglich. Im Gegensatz zur akustischen Phonetik ist die auditive Phonetik nicht für relativ einfache Messungen zugänglich, sondern benötigt eine Zusammenarbeit mit Fachmedizinern.

Die auditive Phonetik ist von herausragender Bedeutung in der klinischen Phonetik für die Diagnose und Therapie sowie in Zusammenarbeit mit der Sprachtechnologie und der Hörakustik für die Entwicklung von prosthetischen Vorrichtungen wie Hörgeräte und Cochlearimplantate.

Aufgrund der Unzugänglichkeit dieser Teildomäne der Phonetik (außer in enger Zusammenarbeit mit klinischen Phonetikern, Medizinern und Hörakustikern) wird diesem Bereich der Phonetik keine weitere Aufmerksamkeit an dieser Stelle geschenkt. Es wird stattdessen auf die weiterführende Spezialliteratur verwiesen (vgl. den Literaturabschnitt 3.1.4).

Phonologie

Die Phonologie behandelt die Funktion, Struktur und die für die Wortunterscheidung wesentlichen Eigenschaften der Grundobjekte der Lautsprache sowie der Strukturen, zu denen diese Objekte kompositionell zusammengesetzt werden. Darüber hinaus behandelt die Phonologie die Abbildung dieser Objekte, ihrer Eigenschaften und ihrer Strukturen auf artikulatorische, akustische und auditive Korrelate in den drei phonetischen Teildomänen (artikulatorische Gesten, akustische Ereignisse und auditive Vorgänge). Die Grundeinheiten der Lautsprache bilden eine Hierarchie, die manchmal als **prosodische Hierarchie** bezeichnet wird: das **Phonem**, aus denen **Silbenkonstituenten** Anlaut, Reim, Kern und Auslaut) zusammengesetzt werden, die **Silbe**, das phonologische **Wort**, der prosodische **Takt**, und **Intonationseinheiten** verschiedener Größe.

Phoneme

Das Phonem ist die kleinste wortunterscheidende Lauteinheit und gilt traditionell als das wichtigste Grundobjekt der Phonologie. Die **Phoneminventare** der Sprachen der Welt unterscheiden sich sehr in der Größe (ca. 20 bis 50) und in den Elementen des Inventars. Generalisierungen (implikative Universalien) sind möglich: z.B. wenn Frikative in einer Sprache vorhanden sind, dann sind auch Verschlusslaute in der Sprache vorhanden. Zur Unterscheidung von der normalerweise detaillierteren Repräsentation von Lauteinheiten in der Phonetik werden Phoneme nicht in eckige Klammern, z.B. [p^b], sondern zwischen Schrägstriche gesetzt, z.B. /p/.

In der Einführungsliteratur wird ein Phonem oft als ‚die kleinste bedeutungsunterscheidende Einheit‘ definiert. Als formale Definition taugt diese Formulierung nicht viel, weil ein Definiens voraussetzt, dass die in ihm enthaltenen Terme entweder evident oder bereits definiert sind. Dies gilt für ‚Bedeutung‘ jedoch nicht, erklärt auch nicht z.B., wie Unsinnswörter oder noch nicht gelernte, nicht verstandene aber bereits existente und durch ihre Form identifizierbare Wörter unterschieden werden. Da die Einheit ‚Wort‘ ein formales Objekt bezeichnet, das zumindest in sehr vielen Sprachen eine intuitiv relativ gut angebbare Einheit ist, die eine Bedeutung haben kann oder auch nicht, eignet sich folgende Definition besser:

Ein Phonem ist die kleinste sequentielle wortunterscheidende Lauteinheit.

Aber auch diese Definition ist noch relativ dürftig, vor allem, weil das Phonem eine Generalisierung bzw. Abstraktion von den phonetischen Äußerungsdetails und nicht direkt im Sprachsignal beobachtbar ist. Die kleinsten wortunterscheidenden Segmente im Sprachsignal sind **Phone**; wenn Phone in unterschiedlichen **Silben-** oder **Wortkontexten** vorkommen (komplementär verteilt sind) und im Vergleich zu anderen Phonem phonetisch ähnlich sind, gelten sie als **Allophone** desselben Phonems.

Geeigneter als diese eindimensionale Definition ist eine komplexere Charakterisierung des Phonems als semiotische Einheit. Wie andere sprachliche Einheiten kann ein Phonem als **Zeichen** verstanden werden, das anhand der semiotischen Dimensionen Struktur (intern und extern) und Interpretation (semantisch und phonetisch) definiert wird:

Struktur: Sprachliche Zeichen haben zwei Strukturdimensionen, die einerseits ihre interne Zusammensetzung und andererseits den externen Kontext, in dem sie vorkommen, kompositionell bestimmen:

Interne Struktur: Phoneme sind die kleinsten sequentiellen wortunterscheidenden Lauteinheiten und haben als solche keine interne sequentiell-temporale Struktur. Sie werden aber auch als Mengen von distinktiven Eigenschaften aufgefasst. Eine traditionelle Definition lautet demnach: Phoneme sind Bündel von distinktiven **Merkmalen** und haben eine simultan-temporale Struktur. Die Merkmale übernehmen dann die Funktion der kleinsten wortunterscheidenden Lauteinheiten.

Externe Struktur: Phoneme sind die kleinsten Bestandteile von **Silben** (die ihrerseits als Bestandteile von größeren Einheiten in der prosodischen Hierarchie definiert werden).

Interpretation: Zeichen haben in einer modellorientierten Sicht zwei Interpretationen im Hinblick auf eine wahrnehmbaren Realität, die als Interpretationspaar den Kern des **semiotischen** Charakters des Zeichens bilden: die mediale Interpretation („Bezeichnendes“) und die semantisch-pragmatische

Interpretation („Bezeichnetes“). Für Phoneme heißt dies, dass es sich um das Paar aus der phonetischen und der semantischen Interpretation der Äußerung handelt:

Phonetische Interpretation: Phoneme werden je nach Position in der externen Struktur als unterschiedliche Allophone (Phon, die einem Phonem zugeordnet werden) interpretiert, die die kleinsten temporalen Segmente von sprachlichen Äußerungen sind. Zum Beispiel wird das Phonem /p/ in „Panne“ behaucht (aspiriert) ausgesprochen und phonetisch als [p^h] dargestellt, in „Spanne“ wird das Phonem aber unbehaucht ausgesprochen und folglich phonetisch mit dem Default-Symbol [p] geschrieben. Diese extensionale oder denotationelle Definition durch Interpretation lautet also: Ein Phonem wird durch eine Menge von Allophenen interpretiert, beispielsweise /p/ =_{def} {[p], [p^h]}) (tatsächlich hat /p/ noch weitere **Allophone**).

Semantische Interpretation: Phoneme haben die Funktion, Wörter zu unterscheiden. Kombinationen von wenigen Phonemen können viele Tausende einfache Wörter kodieren (die ihrerseits durch morphologische Kombinationen weitere Wörter bilden können). Diese Funktion von Phonemen kann also als **Kodierung** definiert werden.

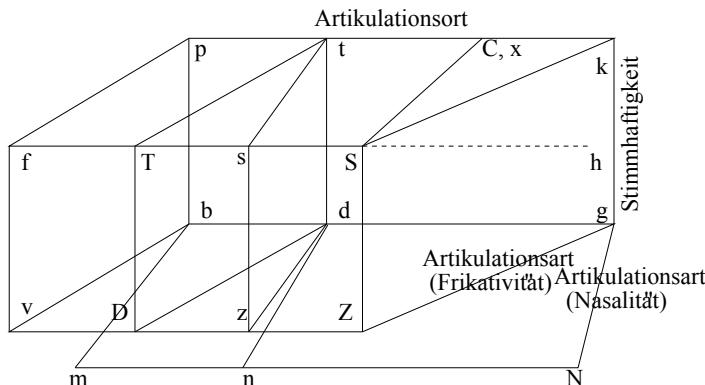


Abbildung 3.4: Paradigmatische Relationen zwischen Konsonanten.

In struktureller Hinsicht gehen Phoneme, wie andere sprachliche Einheiten, zwei Arten von Relationen miteinander ein:

1. klassifikatorische Relationen aufgrund ihrer Eigenschaften, die Ähnlichkeiten und Unterschiede zwischen den Phonemen charakterisieren und traditionell **paradigmatische Relationen** genannt werden,
2. kompositorische Relationen, die das Vorkommen von Phonemen in unterschiedlichen Positionen in Silben bestimmen und traditionell **syntagmatische Relationen** genannt werden.

Die paradigmatischen Relationen zwischen den Konsonanten des Deutschen (ohne Approximanten) werden in Abbildung 3.4 visualisiert. Die syntagmatischen Relationen werden in einem folgenden Abschnitt im Kontext der Silbenstruktur besprochen.

Das Phonem als Grundobjekt ist nicht ganz unkontrovers. In der **Generativen Phonologie** werden abstraktere Generalisierungen, **Morphophoneme**, als einzige abstrakte Grundobjekte angenommen, die direkt phonetisch interpretiert werden. Nach dieser Auffassung ist das Phonem ein Artefakt, das die lautsprachliche Struktur nicht adäquat modellieren lässt. In **prosodischen Phonologien** werden einige Merkmale mit phonematischer Funktion, deren zeitliche Ausdehnung über das einzelne Phonem hinausgeht, als gleichberechtigte Grundobjekte angesehen und **Prosodien** genannt. Die Phoneme oder **phonematischen Einheiten** sind damit unterspezifiziert und müssen durch prosodische Eigenschaften ergänzt werden. Die Stimmlosigkeit von Obstruenten (Plosive, Frikative, Affrikate) im deutschen Silbenauslaut (Auslautverhärtung, z.B. im Auslaut /kst/ von „Axt“ /akst/) wäre demnach eine Prosodie, da sie nicht ein einzelnes Phonem, sondern alle Obstruenten des Auslauts betrifft. In der **Autosegmentalen Phonologie** werden Prosodien, wie z.B. Töne, mit selbständiger klarer Struktur oder Funktionalität Autosegmente genannt und graphisch durch parallele temporal gerichtete Graphen repräsentiert, die durch Bezugskanten (*association lines*) miteinander verbunden werden.

Silbe

Die Silbe ist die kleinste Lautfolge, die als eigenständiges Wort funktionieren kann und besteht in der Regel aus einem Vokal und einem oder mehreren vorangestellten oder nachgestellten Konsonanten. Die Kombinatorik bzw. die Distribution von Phonemen wird im Kontext der Silbe beschrieben. Allerdings wird die Silbe nicht immer als universelles phonologisches Objekt anerkannt, vor allem im Kontext der indo-europäischen Sprachen, die komplexere Lautfolgen aufweisen. Dennoch dient die Silbe als nützliche Vereinfachung beim Verständnis der Lautstrukturen der Sprachen. Es soll hier lediglich angemerkt werden, dass dieser phonologische Silbenbegriff sich vom orthographischen Silbenbegriff unterscheidet.

Aus phonetischer Sicht wird die Silbe etwas anders definiert: Eine Silbe hat eine glockenförmige Sonoritätskontur (etwa: Intensitätskontur), die am Silbenanfang eine geringere Sonorität (Konsonanten) hat, mit dem Vokal die höchste Sonorität erreicht und zum Silbenende eine geringere Sonorität (Konsonanten) hat.

Die Sprachen der Welt unterscheiden sich nicht nur sehr stark in den Phonem-inventaren sondern auch in ihrer Kombinatorik im **Silbenkontext**. Die germanischen Sprachen Deutsch, English, Niederländisch, Dänisch, Norwegisch, Schwedisch, Isländisch haben komplexe Silben mit bis zu 8 Phonemen (der Affrikat /pf/ zählt als ein Phonem): „strümpfst“ /strympfst/ im Kunstwort „bestrümpfst“ (vielleicht: „jemandem die Strümpfe anziehen“) mit der Struktur KKKVKKKK. An den einzelnen k-Stellen können nicht alle Konsonanten vorkommen; an der er-

sten Stelle beispielsweise nur /ʃ/, wie in diesem Beispiel, oder /s/, wie in „Skat“ /skat/. Andere Sprachen haben nur KV- oder KKV-Strukturen, wobei die zweite K-Stelle nur mit den Liquiden /l, r/ besetzt werden kann. Die Möglichkeiten sind sehr vielfältig.

Um **Silbenstrukturen** darzustellen, gibt es viele Notationsarten:

1. Constraints über Lautklassensequenzen:

V, KV, KVK, VK, ..., KKKVKKK, ...

2. Constraints über Lauteigenschaftssequenzen:

z.B. Deutsch: wenn $\#X[\text{Verschlusslaut}][\text{Liquid}]$ eine Phonemsequenz beschreibt und $X = [\text{Konsonant}]$, dann $X = [\text{stimmloser Zischlaut}]$, d.h. in einer Dreikonsonantensequenz am Wortanfang (bezeichnet mit '#') muss der erste Laut ein /s/ oder ein [ʃ] sein.

3. Sonoritätsunterschiede (etwa Unterschiede in der Intensität einzelner Lauten):

$\text{son}_{\text{initial}} > \text{son}_{\text{gipfel}} > \text{son}_{\text{final}}$

4. Eingebettete Strukturen, visualisiert als Baumgraphen oder Klammerungen, z.B. (*Anlaut* str (*Reim* (*Kern* i) (*Auslaut* k))) „Strick“.

Alle diese Notationen haben gewisse Vorteile bei der Generalisierung über wesentliche Eigenschaften von Silbenstrukturen, alle haben aber auch Nachteile, weil sie unterschiedliche Abstraktionen darstellen und daher jeweils auf unterschiedliche Weise unvollständige, fragmentarische Modelle sind:

1. Die KVK-Notation generalisiert nicht über die Detailbeschränkungen in der Konsonantenkombinatorik.
2. Die Eigenschaftsnotationen, zu denen traditionelle phonologische Regelnotationen gehören, erfassen jeweils nur Fragmente der Gesamtstruktur.
3. Die Sonoritätsnotation, wie die erste Notation, erfordert weitere Informationen über spezifische Sprachlauttypen und ihre Positionen in der Silbe.
4. Die hierarchischen Notationen bedürfen einer Ergänzung mit zusätzlichen Einschränkungen der linearen Kombinationsmöglichkeiten (besonders im Auslaut), die quer zur Verzweigungsstruktur des Baumgraphen verlaufen.

Diese Notationen, die oft recht informell gehandhabt werden, können dennoch *lokal* explizit und präzise sein. Aber sie lassen trotzdem vieles offen, insbesondere die Fragen der Konsistenz, der Präzision, der Vollständigkeit und der Korrektheit des *globalen* lautsprachlichen Gesamtsystems. In Abschnitt 3.1.3 wird mit regulären Modellen (endlichen Automaten ; vgl. Unterkapitel 2.2) eine computerlinguistisch adäquate Antwort auf dieses Problem in der Form einer vollständigen Modellierung von Silben als Elemente regulärer Mengen gegeben.

Merkmalstheorie

Die Eigenschaften von Sprachlauten werden in der Regel nicht unabhängig voneinander (z.B. „stimmhaft“, „stimmlos“, usw.) aufgezählt, sondern zu kontrastierenden Elementen der Wertemengen von Attributen gruppiert. Damit partitiert jedes Attribut die Menge der Phoneme in Teilmengen, von der jede Teilmenge mit einem der Werte in der Wertemenge des Attributs assoziiert wird. In den traditionellen phonologischen Theorien sind die Attribute binär, d.h. sie haben eine Wertemenge mit zwei Werten, heißen „Merkmale“, und werden mit einer Notation dargestellt, in der der Wert dem Merkmal vorangestellt wird: [+ stimmhaft] bedeutet z.B. „stimmhaft“, [- stimmhaft] bedeutet „stimmlos“. In einer in der Computerlinguistik gewohnten Schreibweise können die Merkmale als [Stimmhaft: +] und [Stimmhaft: -], oder expliziter als [Stimmhaftigkeit: stimmhaft] und [Stimmhaftigkeit: stimmlos] ausgedrückt werden. Das Phonem /p/ kann beispielsweise mit einer der gängigen auf artikulatorischen Korrelaten basierenden Merkmalstheorien folgendermaßen definiert werden:

$$/p/ = \begin{bmatrix} + & \text{konsonantisch} \\ - & \text{vokalisch} \\ - & \text{kontinuierlich} \\ - & \text{stimmhaft} \end{bmatrix}$$

Die phonologische Regel, die die Auslautverhärtung von Obstruenten im Deutschen ausdrückt und dabei über die **natürliche Klasse** von Phonemen{/b/, /d/, /g/, /v/, /z/} generalisiert, lautet in Merkmalsnotation, einmal in der konventionellen unterspezifizierten Form, einmal in der voll spezifizierten Form:

1. $[+ \text{ stimmhaft}] \rightarrow [- \text{ stimmhaft}] / \begin{bmatrix} + \text{ konsonantisch} \\ - \text{ vokalisch} \end{bmatrix} \#$
2. $\begin{bmatrix} + & \text{konsonantisch} \\ - & \text{vokalisch} \\ + & \text{stimmhaft} \end{bmatrix} \# \rightarrow \begin{bmatrix} + & \text{konsonantisch} \\ - & \text{vokalisch} \\ - & \text{stimmhaft} \end{bmatrix} \#$

Phonologie und Orthographie

Die Relation zwischen Phonologie und Orthographie ist ein relativ eigenständiger Gegenstandsbereich. Die Schriftsysteme der Sprachen sind in gewisser Hinsicht komplexer und variabler als ihre Phonemsysteme. In logographischen Systemen (z.B. in der chinesischen Orthographie) kodieren die Schriftzeichen Morpheme und nicht Phoneme. Im lateinischen alphabetischen System der europäischen Sprachen werden ebenfalls Logogramme verwendet, aber nur für Zahlen und mathematische Operatoren (z.B. Ziffern 0, ..., 9, Operatoren '+', '-' usw.), die damit sprachunabhängig, aber mit völlig unterschiedlichen Aussprachen verwendet werden können. Einem ähnlichen Prinzip folgen Emoticons in schriftlichen

Kurznachrichten, wobei diese oft auch ikonischen Charakter (Ähnlichkeit zwischen Form und Bedeutung) haben.

Eine alphabetische Orthographie kann streng phonematisch mit einer eindeutigen Beziehung zwischen Graphemen und Phonemen sein. Dies gilt vor allem für Sprachen, deren Orthographieentwicklung noch relativ neu ist oder mit phonematischer Orientierung reformiert wurde. Bei alphabetischen Orthographien, die bereits seit vielen Jahrhunderten mit relativ wenigen Veränderungen bestehen (vgl. Englisch, Französisch), hat sich die Aussprache weit mehr verändert als die Orthographie, so dass die Phonologie-Orthographie-Relation sehr komplex geworden ist: französisch „eaux“ /ø:/ (Plural von „eau“ „Wasser“) und die berühmte englische „-ough“-Reihe: „tough“ /taf/, „through“ /θru:/, „cough“ /kɒf/, „though“ /ðəʊv/, „thorough“ /θʌrəʊ/, „bough“ /baʊ/, mit jeweils unterschiedlichen Aussprachen der Reimsequenz „ough“.

Für ältere Sprachstufen vor dem 20. Jahrhundert existieren, wenn überhaupt, nur schriftliche Zeugnisse. Daher kommt der Phonologie-Orthographie-Relation für die Rekonstruktion früherer Sprachstufen, die auch eine herausfordernde computerlinguistische Aufgabe ist, große Bedeutung zu.

Für die Sprachtechnologie hat die Phonologie-Orthographie-Relation ebenfalls eine zentrale Bedeutung in der Form von Graphem-Phonem-Übersetzungsregeln in Sprachsynthesesystemen und automatischen Spracherkennungssystemen (siehe Unterkapitel 3.2).

Die Vielfalt phonologischer Theorien

Das Panorama phonologischer Theorien ist immens, wobei der Eindruck manchmal entstehen kann, dass neue Theoriennamen eher aus strategischen als aus theoretischen oder empirischen Gründen eingeführt werden. Es gibt einige nützliche Artikelsammlungen zur Geschichte der Phonologie, die einen guten Überblick ermöglichen (siehe den Literaturabschnitt 3.1.4).

In diesem Abschnitt sollen extrem kurze Charakterisierungen der wichtigsten Richtungen in der Phonologie als kleine Wegweiser für die Literaturrecherche gegeben werden, natürlich bei Gefahr grob unzulässiger Verallgemeinerungen und ohne mit der Spezialliteratur konkurrieren zu wollen.

Strukturalismus: Die strukturalistische Denkweise in der Phonologie wurde von de Saussure in den ersten beiden Jahrzehnten des 20. Jahrhunderts eingeführt. Sie postuliert, dass sprachliche Formen und Strukturen synchron (zu einer bestimmten Zeit) ein zusammenhängendes System von paradigmatischen und syntagmatischen Relationen bilden. Diese Idee stand im Kontrast zu früheren Ansätzen in der komparativen Philologie (diachrone Rekonstruktion früherer Sprachstufen), in der pädagogischen Grammatik, in der Logik, der Rhetorik und in der hermeneutischen Behandlung der Sprache in Philosophie und Theologie. Nach dieser allgemeinen Charakterisierung könnten prinzipiell alle modernen phonologischen Theorien als im weiteren Sinne strukturalistisch angesehen werden. Spätere Entwicklungen änderten diese Grundsätze zwar nicht, führten aber weitere Gesichtspunkte

ein. Repräsentanten unterschiedlicher Ausprägungen des Strukturalismus im engeren Sinne sind z.B. in Europa neben Ferdinand de Saussure auch Louis Hjelmslev (Glossematik), in den USA Leonard Bloomfield, Zellig Harris, Charles Hockett, Kenneth Pike (amerikanischer Strukturalismus, Distributionalismus). Vorgänger der Strukturalisten waren unter den komparativen Philologen die Junggrammatiker, die die Regelmäßigkeit aller diachronen Lautveränderungen betonten.

Funktionalismus: Der Funktionalismus in der Phonologie ist eine besondere Ausprägung des Strukturalismus, die die sprachlichen und situativen Kontexte fokussiert, in denen sprachliche Strukturen zu lokalisieren sind, beispielsweise im Prager Funktionalismus, der die Funktionen der Sprache in der Kommunikation und in der Kognition (Gestaltpsychologie) hervorhob und mit der Unterscheidung Sprachgebilde – Sprechakt der späteren Chomsky'schen Kompetenz – Performanz- bzw. I-Language – E-Language-Unterscheidung zuvorkam. Die Londoner Ausprägung bei Firth führte zu einer genaueren Differenzierung zwischen phonematischen und prosodischen Funktionen der Eigenschaften von Sprachlauten, die zu einem verallgemeinerten Prosodiebegriff führte. Der Funktionalismus von Halliday baut auf dem Firth'schen Funktionalismus auf, führte zur ersten theoretisch und empirisch differenzierten Intonationstheorie, ist aber im Hinblick auf den Prosodiebegriff konservativer als bei Firth. Auch der bereits erwähnte Ansatz von Pike (Tagmemik) hat starke funktionalistische Züge.

Generative (und verwandte) Phonologien: Die generative Phonologie betont im Gegensatz zum Strukturalismus eher die formalen Aspekte von phonologischen Beschreibungen als deren empirische Basis, indem von Prämissen (zugrundeliegenden lexikalischen Repräsentationen von Wörtern) und sequentiell angewendeten Ableitungsregeln (phonologischen Regeln) eine phonetische Repräsentation wie ein mathematischer Beweis hergeleitet wird. Vorläuferarbeiten von Halle, Chomsky und ihren Schülern führten zum Standardwerk *Sound Pattern of English* („SPE“) in 1968, das eine Kontroverse über die Abstraktheit phonologischer Repräsentationen auslöste. Das grundlegende Modell, das auf Phoneme und Silben verzichtete, verwendete lineare Verkettungen von Merkmalsbündeln (flachen Attribut-Wert-Strukturen) und Informationen über Wort- und Satzkonstituentengrenzen, führte aber auch eine Theorie der rekursiven Zuordnung von Betonungen und der Konstruktion komplexer Wörter ein. Die Abstraktheitskritik führte zur **Natürlichen Phonologie**, während die Kritik an der Linearität und Probleme bei der Anordnung phonologischer Regeln zur **Lexikalischen Phonologie** (Stratifizierung der Rekursion in autonome Schichten), zur **Autosegmentalen Phonologie** (Abstraktion quasi-autonom strukturierter Sequenzen prosodischer Eigenschaften aus den linearen Ketten), zur **Metrischen Phonologie** (Weiterentwicklung der rekursiven Betonungstheorie) und zur **Optimalitätstheorie** (Zulassung von Constraintverletzungen) führte. In der Optimalitätstheorie, die computerlinguistisch wohl die interessanteste (wenn auch die am heftigsten

umstrittene) Theorie ist, werden geordnete Regeln als sequentiell angeordnete deklarative Constraints dargestellt, die sukzessive den Suchraum für korrekte phonetische Interpretationen eines Lexikoneintrags eingrenzen. Die Constraints lassen Merkmalveränderungen zwar nicht zu, können aber verletzt werden. Die Interpretationen mit den wenigsten Constraintverletzungen gelten als ‚optimal‘. Die **Generative Phonologie** und einige ihrer hier genannten Nachfolger wurden durch Kay, Kaplan sowie Karttunen mit endlichen Automaten modelliert.

Neben diesen bekannteren Richtungen wurden einige Ansätze auf formallogischer Grundlage entwickelt, die weitere Entwicklungen zwar beeinflussten, jedoch wenig allgemeine Beachtung gefunden haben: Deklarative Phonologie (Bird, Ellison), Defaultlogische Phonologie (Gibbon), Montague-Phonologie (Bach und Wheeler), Mereologische Phonologie (Batóg).

Natürlich lässt sich die Vielfalt von Phonologien nicht vollständig in dieses einfache Schema pressen. Eigenschaften von Theorien in der einen Gruppe sind teilweise auch bei Theorien in anderen Gruppen zu finden.

Prosodie

Bezogen auf die indoeuropäischen Sprachen gehören traditionell zum Bereich der Prosodie diejenigen funktionalen lautlichen Eigenschaften, die eine längere Zeitspanne beanspruchen als ein Phonem, und die phonetisch durch die Grundfrequenz, die Intensität oder die zeitliche Organisation der Äußerung, z.B. den Rhythmus, interpretiert werden. Diese Definition ist jedoch nicht unkontrovers. Es gibt andere Eigenschaften, die länger sein können als ein Phonem, z.B. bei Assimilationen (Anpassung benachbarter Laute aneinander), in denen beispielsweise die Artikulationsart über mehr als ein Phonem beibehalten wird, etwa bei der Labialisierung von /n/ vor einem labialen Verschlußlaut: „in Bonn“ /im bɔn/. In **prosodischen Phonologien** werden auch solche Eigenschaften als Prosodien in einem weiteren Sinne klassifiziert. Bereits erwähnt wurde auch die Auslautverhärtung im Deutschen als Prosodie im weiteren Sinne.

Die Sprachen der Welt bieten eine breite Palette prosodischer Eigenschaften im engeren Sinne, mit Funktionen, die teilweise ganz anders sind als in den indoeuropäischen Sprachen, z.B. **phonematische** oder **morphematische Töne** (funktionale Grundfrequenzmuster), die phonematisch als distinktive Merkmale, oder als grammatische Morpheme, oder konfigurativ als Markierungen bestimmter grammatischer Strukturen funktionieren. Die Lautdauer als phonematisches Merkmal ist auch in den indoeuropäischen Sprachen verbreitet, aber in afrikanischen Sprachen der Niger-Congo-Familie kommt die Lautdauer mitunter auch mit morphematischer Funktion, z.B. als Negativmarkierung vor.

An dieser Stelle kann nur ein kurzer Überblick über Wort-, Satz- und Diskursprosodie gegeben werden; weiterführende Lektüre wird im Literaturabschnitt 3.1.4 angegeben.

Wortprosodie

Die wortprosodischen Merkmale tragen phonematisch zur Wortunterscheidung bzw. morphematisch zur Wortbedeutung oder zur Wortstrukturmarkierung bei. Solche lautsprachlichen Eigenschaften sind vor allem von großer Bedeutung für die Sprachsynthese. Folgende Eigenschaften können unterschieden werden:

Phonematische Wortprosodie: Die phonematischen, d.h. wortunterscheidenden wortprosodischen Mittel in den Sprachen der Welt umfassen die Kategorien Ton, Tonakzent und Betonung, die möglicherweise sprachtypologisch ein Kontinuum bilden:

Ton: Ein erster prosodischer Sprachtyp wird durch die phonematische Verwendung des Grundfrequenzverlaufs als Ton in Silben zur Wortunterscheidung charakterisiert. Das Mandarin-Chinesische hat z.B. 4 Töne: hoch flach, mitte-hoch steigend, mitte-tief-hoch fallend-steigend, hoch-tief fallend und zusätzlich der ‚tonlose Ton‘, der sich aus dem weiteren tonalen Kontext ergibt; die Interpretation der Töne im Kontext ist recht komplex. Die meisten Niger-Kongo-Sprachen in West-, Zentral-, Ost- und Südafrika haben zwei, drei oder vier sogenannte Registertöne, also Töne, die nur durch die Tonhöhe und nicht durch eine **Tonveränderung** (Kontur) charakterisiert sind; eventuell vorkommende Konturen lassen sich historisch, im Dialektvergleich und sprachintern als Kombinationen von Registertönen begründen.

Tonakzent: Ein anderer Sprachtyp kennt Tonakzente, die im Gegensatz zu den Tönen in der Regel eine einzige Form haben, jedoch an unterschiedlichen Stellen im Wort vorkommen. Beispiele für solche Sprachen sind Japanisch und Schwedisch.

Betonung: Ein dritter Sprachtyp, zu dem auch Deutsch, Niederländisch und Englisch gehören, verwendet Betonungen, d.h. phonetisch variable Interpretationen einer im Lexikon ausgezeichneten betonten Silbe durch Erhöhung oder Absenkung der Tonhöhe oder durch verlängerte Silbendauer. Ein deutsches Wort wie „Tenor“ bedeutet „männlicher Sänger mit hoher Stimme“ oder „ungefährer Inhalt“, je nachdem, ob die zweite oder die erste Silbe betont wird.

Morphematische und morphosyntaktische Wortprosodie: In den meisten Niger-Kongo-Sprachen, sowie in einigen Tibeto-Burmanischen und südamerikanischen Sprachen kommen **Töne** mit grammatischer Bedeutung vor, die Flexionsmorpheme interpretieren oder die interne Grenze bei Wortkomposita markieren. Die prosodische Markierung der Wortstruktur ist im Deutschen auch zu finden, beispielsweise bei der Erstbetonung der Konstituenten von Komposita: „SCHREIBtisch“, nicht „SchreibTISCH“ (vgl. aber regionale Abweichungen bei Namen, beispielsweise allgemein „STEINhagen“ gegenüber regional „steinHAgen“).

Satz- und Diskursprosodie

In Untersuchungen zu den indoeuropäischen Sprachen ist die Satzprosodie oder **Intonation**, charakterisiert durch einen Grundfrequenzverlauf über einen Satz oder Teilsatz, wohl der klassische Bereich der Prosodie, wobei eine Unterscheidung zwischen satzorientierter Funktion und Funktion im Diskurs schwer aufrechtzuerhalten ist. Die wichtigsten satzprosodischen Funktionen, die der Intonation zugeschrieben werden, sind die **Phrasierung** (die Einteilung einer Äußerung in intonatorische Phrasierungseinheiten, **Intonationsphrasen**), die **Akzentplatzierung** (Zuordnung eines Satzakzents zu einer Satzkonstituente) und die Zuweisung eines **Terminaltons**:

Phrasierung: Sprachliche Äußerungen werden durch relativ klar erkennbare Grundfrequenzkonturen in Intonationsphrasen eingeteilt, die je nach Sprechstil in der Regel, aber nicht notwendigerweise, größeren grammatischen Einheiten wie Nominalphrasen, Satzteilen oder Sätzen zugeordnet werden. Beispiele sind in den Grundfrequenzverläufen in den Abbildungen 3.3 und 3.5 zu finden, die Gesamtkonturen mit lokalen Modulationen zeigen.

Akzentplatzierung: Innerhalb einer Intonationsphrase werden die Wortakzente (phonetische Interpretationen der abstrakten, lexikalisch festgelegten Wortbetonungen) in formellen Sprechstilen rhythmisch angeordnet, in informellen Stilen weniger rhythmisch sondern abhängig von spontanen Formulierungsprozessen und pragmatischen Constraints. Den Wortakzente überlagert sind die Satzakzente, in der Regel nur eine pro Intonationseinheit, die Fokus-, Kontrast- und Emphasefunktionen haben können. Abbildung 3.3 auf Seite 177 zeigt eine akzentuierende Erhöhung des Tonhöhenverlaufs auf den Silben „Lö“ und „Maus“ in den Wörtern „Löwe“ und „Maus“.

Terminalton: Der steigende, fallende, komplex steigend-fallende oder fallend-steigende (seltener noch komplexere) Terminalton ist wohl das auffälligste Element der Intonation bzw. der Satzprosodie. Dem Terminalton werden in der Literatur, vor allem in der sprachdidaktischen Literatur, recht spezifische grammatische (Frage, Aufforderung, Ausruf, usw.) oder pragmatische (emotionale, wertende, usw.) Bedeutungen zugeschrieben. Solche Bedeutungen werden jedoch sehr oft assoziativ aus dem Wortlaut oder dem Situationskontext heraus interpretiert und sollten nicht allein der Intonation zugeschrieben werden.

Die **Terminalkonturen** selbst haben in der Regel lediglich die Funktion, die Abgeschlossenheit oder Nichtabgeschlossenheit einer grammatischen Einheit (z.B. zwischen Subjekt und Verb, bei einer Liste) oder eines Diskurstückes (z.B. Frage-Antwortsequenzen) anzuzeigen. Diese Funktion ist in Abbildung 3.3 am Ende der Intonationskurve zu sehen: Es handelt sich um den Titel einer Geschichte, der mit einer leichten Tonhöhensteigung endet, die die Fortsetzung durch den Hauptteil der Geschichte ankündigt.

Eine Terminalkontur sowie eine globale Tonhöhenkontur können auch soziale und emotionale Funktionen haben.

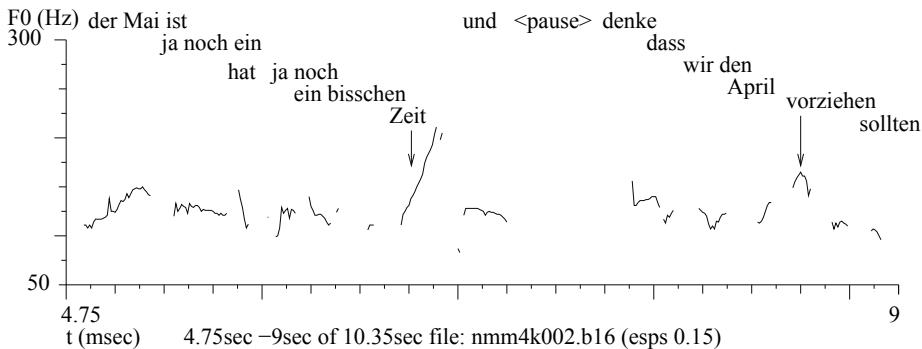


Abbildung 3.5: Grundfrequenzverlauf einer konversationellen Äußerung.

Die Abbildung des Grundfrequenzverlaufs in Abbildung 3.5 ist mit der orthographischen Transkription der Äußerung lose beschriftet, um den Frequenz-Text-Bezug anzudeuten. Die Phrasierung wird durch die Konjunktion „und“ mit nachfolgender Pause, die Akzentplatzierungen auf „Mai“ und „April“ sind durch lokale Frequenzsteigungen und durch Terminaltöne auf „Zeit“ (steigend) und „VOR“ (in „vorziehen“, fallend) durch deutliche Frequenzveränderungen markiert.

Integration von Prosodie, Phonologie und Phonetik

Wenn die Funktionen der prosodischen Objekte im Detail betrachtet werden, fällt auf, dass sie im Großen und Ganzen den Grundobjekten Phonem, Morphem, Wort und Satz zugeordnet werden können. Es liegt also nahe, auch den prosodischen Objekten eine semiotische Charakterisierung unter Bezugnahme auf die Grundobjekte zukommen zu lassen, wie bereits bei den Phonemen. Die phonetische Interpretation eines Phonems ist entweder ein Allophon oder ein Ton. Die phonetische Interpretation eines Morphems ist eine Funktion der phonetischen Interpretation der Phoneme, die ihm zugeordnet sind, und einer diesen Phonemen zugewiesenen prosodischen Einheit, z.B. ein Ton oder ein Akzent.

Die kompositionelle Hierarchie kann fortgesetzt werden: Die phonetische Interpretation eines Wortes ist eine Funktion der phonetischen Interpretation seiner Bestandteile und der prosodischen Strukturmarkierung, die phonetische Interpretation eines Satzes ist eine Funktion der phonetischen Interpretation seiner Bestandteile und der prosodischen Markierungen der Phrasierung, der Akzentsetzung und des Terminaltons. Die integrierte Hierarchie, die eine generalisierte phonetische Interpretation darstellt, wird in Abbildung 3.6 visualisiert. Diese Sichtweise ermöglicht es dem Computerlinguisten, die umfangreiche, aber recht fragmentierte Literatur zum Thema Prosodie zu systematisieren und im Rahmen bekannter kompositorischer Prinzipien zu formalisieren und implementieren.

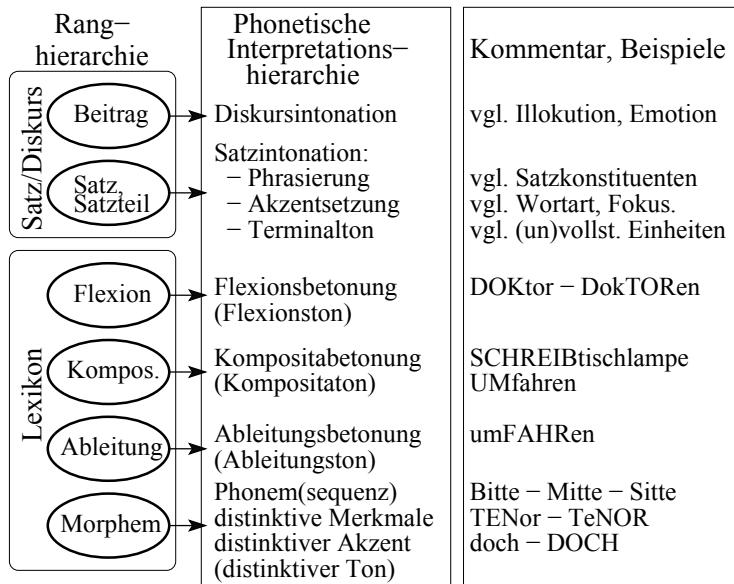


Abbildung 3.6: Generalisierte phonetische Interpretation zur Integration von phonologischen und prosodischen Einheiten.

3.1.2 Empirische Methoden

Die empirischen Grundlagen für die Phonetik und Phonologie sind im Prinzip gleich und ergeben einen dreidimensionalen empirischen Methodenraum:

Korpus: Ein Korpus ist eine Sammlung beobachteter, aufgenommener und auf Speichermedien verfügbarer sprachlicher Äußerungen, die entweder einzeln durch Selbst- oder Fremdbefragung direkt elizitiert, oder für Experimente geplant, oder als systematische oder authentische (nicht für phonetische Untersuchungszwecke erstellte) Datensammlungen aufgebaut werden. Ein Korpus enthält aber auch in der Regel eine mit Metadaten systematisch dokumentierte Menge von solchen akustischen (zunehmend auch multimedialen) Aufnahmen mit den dazugehörigen Transkriptionen, Annotationen, und eventuell auch ein Korpuslexikon.

Analyse: z.T. durch unterschiedliche Werkzeuge unterstützte Kategorisierungen von Äußerungen in einem Korpus im Hinblick auf ihren sprachlichen Status, ihre Bestandteile und die wahrnehmbaren Eigenschaften dieser Bestandteile durch den phonetisch ausgebildeten Experten. Die Kategorisierungen werden in der Regel unter Zuhilfenahme von standardisierten phonetischen Alphabeten und Merkmalssystemen und Annahmen über Silben- und Wortstruktur durchgeführt.

Werkzeuge: intellektuelle Werkzeuge (z.B. phonetische Alphabete und Merkmalsysteme, Parameterbeschreibungen, Ontologien usw.) und operationale Werkzeuge für instrumentelle Messungen, sowie deren Visualisierung und statistische Auswertung, sowie für die symbolorientierte Analyse und Modellierung von lautsprachlichen Äußerungen. Die am meisten verwendeten operationalen Werkzeuge sind Programme zur Anzeige der akustischen Eigenschaften von Sprachsignalen und zur Annotation (Zuordnung von Transkriptionen zu Sprachsignalen).

Die beiden Disziplinen Phonetik und Phonologie unterscheiden sich in ihrer Gewichtung der verschiedenen Spielarten der beiden empirischen Grundlagen. Es gibt aber nicht nur Überlappungen zwischen den Disziplinen: Die Disziplinen positionieren sich tendenziell an ganz anderen Stellen im empirischen Methodenraum. Es gibt aber keine scharfe Trennlinie zwischen phonetischen und phonologischen Methoden, wie die Bezeichnungen von Ansätzen wie „Phonology as Functional Phonetics“ oder „Laboratory Phonology“ andeuten.

Die methodologischen Überlappungen und die Schnittstellen (im Sinne von gemeinsamen Repräsentationen von Fakten und Regeln) zwischen Phonetik und Phonologie werden seit mehr als einem Jahrhundert kontrovers diskutiert. Je nach empiristischer, kognitivistischer oder anwendungsorientierter Einstellung werden die Dimensionen Korpus, linguistische Kategorisierung oder Werkzeuge in den Vordergrund gestellt. Am sinnvollsten scheint es zu sein, den gemeinsamen empirischen Methodenraum einzusetzen und einzelne Ansätze oder Studien entlang der drei Dimensionen des Methodenraums zu charakterisieren.

Die folgenden Teilabschnitte geben einen kurzen Überblick über empirische Methoden, Techniken zur Transkription und Annotation, experimentelle und korpusphonetische Methoden, sowie Anwendungsbereiche der Phonetik.

Methodenüberblick

Auf die drei Teildomänen der Phonetik lassen sich unterschiedliche Methoden anwenden. Die **Ohrenphonetik**, die auf dem geschulten Hörsinn des ausgebildeten Phonetikers aufbaut, wird von der **Instrumentalphonetik**, bei der Messinstrumente und -software verwendet werden, unterschieden. Eine ohrenphonetische Analyse zur Bestimmung des genauen Gegenstandsbereichs ist stets Voraussetzung für eine sinnvolle instrumentalphonetische Analyse. Verwirrend ist die oft anzutreffende Verwendung von „auditiv“ nicht nur für die auditive Teildomäne der Phonetik sondern auch für die ohrenphonetische Methode, die dann „auditive Methode“ heißt. Manchmal wird „impressionistische Phonetik“ statt „Ohrenphonetik“ benutzt. Die Bezeichnung „Wahrnehmungsphonetik“ wird manchmal auch in beiden Bedeutungen verwendet: für die Untersuchung der Wahrnehmung und für die Untersuchung durch Wahrnehmung, z.B. in Wahrnehmungsexperimenten. Gegenstandsbereich und Methode sollten aber auf jeden Fall konsistent auseinander gehalten werden.

Orthogonal zur Unterscheidung zwischen Ohren- und Instrumentalphonetik ist die weitere Unterscheidung zwischen **qualitativen Methoden**, bei denen

einzelne Sprachsignale beobachtet und transkribiert oder gemessen, dann analysiert und illustriert werden, von **quantitativen Methoden**, bei denen größere Datenmengen aus Experimenten und Korpora statistisch untersucht werden.

Bei den qualitativen Methoden wird weiter unterschieden zwischen teilnehmender Beobachtung (der Beobachter interagiert authentisch, d.h. nicht als Forscher zu erkennen) und nicht-teilnehmender Beobachtung (der Forscher wird klar von den untersuchten Personen unterschieden). Teilnehmen und Beobachten sind nicht unbedingt miteinander kompatibel. Bei der teilnehmenden Beobachtung muss also abgewogen werden, ob eher der teilnehmende oder der beobachtende Aspekt bevorzugt wird. Auf jeden Fall müssen ethische und juristische Gesichtspunkte bei der teilnehmenden Beobachtung berücksichtigt werden.

Qualitative Untersuchungen sind auch stets Voraussetzung für sinnvolle quantitative Untersuchungen. Insofern geht die phonetische Analysearbeit einen Weg von ohrenphonetisch-qualitativen zu ohrenphonetisch-quantitativen Untersuchungen, oder von ohrenphonetisch-qualitativen über instrumentalphonetisch-qualitativen (direkte Inspektion von Messungen) zu instrumentalphonetisch-quantitativen, statistisch auswertenden Methoden.

Bei den quantitativen Methoden unterscheidet man ferner zwischen **experimentellen Methoden**, bei denen sorgfältig strukturierte Datentypen in Rezeptions- und Produktionsexperimenten untersucht werden, und **korporusphonetischen Methoden**, bei denen große Mengen an weniger homogenen Sprachaufnahmen eines Korpus aus einem allgemeiner spezifizierten Szenario untersucht werden.

Die Verwendbarkeit der Methoden hängt von der Teildomäne ab. Mit qualitativen Methoden lassen sich die artikulatorische Domäne (durch Selbstwahrnehmung der Sprechorgane) und die akustische Domäne (durch Höreindrücke vom Schall) untersuchen, aber nicht die auditive Domäne. Vorgänge im Ohr kann man nicht direkt beobachten. **Instrumentalphonetische Methoden** lassen sich auf alle drei Teildomänen anwenden, allerdings erfordert die messphonetische Untersuchung der Produktion (teilweise) und der Rezeption (vollständig) die Zusammenarbeit mit Fachmedizinern bzw. mit Neurologen. Nur die akustische Domäne ist für medizinische Laien problemlos messtechnisch zugänglich, wenngleich erhebliche technische Kenntnisse der Akustik des Sprachsignals und der Signalverarbeitung für die erfolgreiche Arbeit notwendig sind. Diese Domäne ist für die bekanntesten Anwendungen der Phonetik in der Sprachtechnologie – automatische Sprachsynthese, Sprecher- und Spracherkennung – relevant. Technologische Anwendungen in der Produktionsdomäne (z.B. Sprechprothesen) oder in der Rezeptionsdomäne (z.B. Hörgeräte) erfordern medizintechnische Zusammenarbeit.

Ressourcen: Aufnahme, Transkription, Annotation

Die Qualität phonetischer Untersuchungen und damit auch indirekt die Qualität auch von phonologischen Untersuchungen hängt von der Qualität der empirischen Ressourcen ab, die durch den empirischen Methodenraum bereits definiert wurden: Korpus, Analyse, Werkzeuge. Der Qualitätssicherung phonetischer

Ressourcen ist viel Aufmerksamkeit gewidmet worden, vor allem im Kontext der Anwendung phonetischer Analysen in der Sprachtechnologie bei der Entwicklung von Sprachsynthese- und Spracherkennungssystemen, aber auch in hochqualitativer Dokumentation der vom Aussterben bedrohten Sprachen der Welt (siehe den Literaturabschnitt 3.1.4).

Die Direkttranskription einzelner Zufallsbeobachtungen ohne akustische Aufnahmen wird z.B. noch in der Analyse von Versprechern (die nicht leicht zu elizitieren sind) und in der Fehleranalyse im Fremdsprachenunterricht verwendet. In der deskriptiv-linguistischen Feldforschung werden auch z.T. noch Direkttranskriptionen ohne akustische Aufnahmen angefertigt; diese Methode verschwindet aber allmählich mit dem zunehmenden Bewusstsein der Bedeutung wiederverwendbarer hochqualitativer phonetischer Ressourcen.

In phonologischen Untersuchungen wurden traditionell keine oder kaum Beobachtungen im üblichen empirischen Sinne gemacht. Vor allem Muttersprachendaten wurden (und werden noch) manchmal nur introspektiv vom Phonologen erdacht. Die introspektive Methode wird vor allem von Soziolinguisten kritisiert: Es ist bekannt, dass introspektive Urteile stark durch normativ-subjektive Kategorisierungsschemata beeinflusst werden, die mit der Äußerungswirklichkeit nicht gut übereinstimmen. Diese Variante der qualitativen Methode wird immer weniger verwendet, sondern durch empirisch abgesicherte qualitative und quantitative Methoden ergänzt.

Aufnahme

Zur **Aufnahmeplanung** gehören drei Phasen, die über den eigentlichen Aufnahmevergäng hinausgehen und sorgfältig durchgeführt werden müssen, um den heutigen Ansprüchen an Wiederverwertbarkeit (*reusability*) und Nachhaltigkeit (*sustainability*) zu genügen: die Designphase (*pre-recording phase*), die Aufnahmephase (*recording phase*) und die Bearbeitungsphase (*post-recording phase*).

Designphase: In der Designphase geht es darum, den Rahmen für die Datenaufnahme zu spezifizieren: die Fragestellung, den Untersuchungstyp (z. B. Produktions-, Wahrnehmungs- oder Reaktionsexperiment, oder der Korpusdatentyp für dialogische Interaktionen), das Szenario und (bei experimentellen Fragestellungen) die Instruktionen und Vorlagen sowie Datenverwendungsvereinbarungen mit den Versuchspersonen, Aufnahmeausrüstung, sowie Aufnahmeort und -zeit. Diese Informationen gehen in die Metadaten zum Korpus ein. Probeaufnahmen werden durchgeführt, um den Aufnahmealblauf zu testen.

Aufnahmephase: In die Aufnahmephase Sprachaufnahme fällt der tatsächliche Ablauf der Datenerhebung, im Studio oder in einer natürlichen Umgebung, je nach Designspezifikation. Für die Aufnahme müssen alle Materialien bereitgestellt werden (vorbereitete Unterlagen, Geräte, Stromversorgung), Trinkwasser für die Sprecher (z.B. ein Schluck alle 5 oder 10 Minuten, um eine Austrocknung der Stimmbänder zu vermeiden, die die Aufnahmequalität beeinträchtigen würde). Während der Aufnahme muss für korrekte

Mikrofonplatzierung und Signalaussteuerung gesorgt werden. Gleichzeitig werden standardisierte Metadaten über Aufnahmematerialien und -verlauf festgehalten.

Bearbeitungsphase: Der erste Schritt in der Bearbeitungsphase ist die Archivierung der aufgenommenen Daten und der Metadaten mit systematischen und eindeutigen Dateinamen. Die folgenden Schritte der Transkription, Annotation und Korpuslexikon- oder Sprachmodellerstellung werden gesondert behandelt.

Transkription

Eine Transkription ist die Zuordnung einer symbolischen Repräsentation zu einer sprachlichen Äußerung, heutzutage normalerweise zu einer Audio- oder Videoaufnahme einer sprachlichen Äußerung. Die Möglichkeiten der symbolischen Repräsentation sind vielfältig und hängen von der Fragestellung ab. Auf jeden Fall müssen die **Transkriptionskonventionen** exakt spezifiziert werden, nicht ad hoc erfunden; hierzu ist häufig eine Testphase erforderlich, wenn noch wenige Erfahrungen mit dem Datentyp vorliegen.

In diesem Beitrag werden Transkriptionskonventionen für Videoaufnahmen von Äußerungsvorgängen und sprachlichen Interaktionen nicht behandelt. Diese sind noch relativ wenig standardisiert und werden immer weiter entwickelt (siehe aber den Literaturabschnitt 3.1.4).

Die wichtigsten Transkriptionstypen für Audiodaten verwenden die IPA-Transkriptionskonventionen (Abbildung 3.15), für die maschinelle Verarbeitung auch die SAMPA- und Unicode-Kodierungen des IPA. Die wichtigsten Transkriptionstypen werden hier beschrieben.

Orthographische Transkription: Die orthographische Transkription folgt den Standardregeln der Orthographie und bedarf in dieser Hinsicht keines weiteren Kommentars. Zitate im Textzusammenhang werden mit den üblichen Zitierzeichen gekennzeichnet.

Modifizierte orthographische Transkription: In der Konversationsanalyse oder in der Transkription der Kindersprache wird oft eine modifizierte orthographische Transkription verwendet, die Vokalisationen, Geräusche und nicht-standardisierte Aussprachevarianten andeuten soll. Diese Transkriptionsart ist als Grundlage für funktional orientierte sprachliche Analysen entwickelt worden und für phonetische und sprachtechnologische Zwecke nicht gut geeignet. Zitate im Textzusammenhang werden mit den üblichen Zitierzeichen gekennzeichnet.

Phonotypische Transkription: Die phonotypische oder morphophonematische Transkription setzt eine morphologische Analyse der Sprache voraus und generalisiert über morphologisch bedingte Varianten, beispielsweise „Hund“ mit den Stämmen /hunt/ und /hund/ (vgl. „der Hund“ ausgesprochen

„Hunt“, „des Hundes“). Morphophoneme werden manchmal großgeschrieben und können mit Schrägstrichen zitiert werden, z.B. /hʊnD/, oder, zur Unterscheidung von phonematischen Transkriptionen, zwischen spitzen oder geschweiften Klammern.

Kanonische phonematische Transkription: Die kanonische phonematische Transkription ist die wortunterscheidende Transkription, die in einem Aussprachewörterbuch verwendet wird und die phonematischen Kriterien für die Abstraktion von phonetischen Details aufgrund der phonetischen Ähnlichkeit und der komplementären Distribution der Allophone erfüllt. Neben der orthographischen Transkription ist die kanonische phonematische Transkription die nützlichste Transkriptionsart in der Computerlinguistik und Sprachtechnologie. In der Sprachtechnologie wird eine kanonische phonematische Transkription in der Regel automatisch anhand eines Aussprachelexikons und Graphem-Phonem-Übersetzungsregeln erzeugt, eine Prozedur, die Graphem-Phonem-Übersetzung oder Phonetisierung genannt wird. Kanonische phonematische Zitate im Textzusammenhang werden mit Schrägstrichen gekennzeichnet.

Weite phonetische Transkription: Die weite phonetische Transkription ist eine phonematische Transkription, die nicht unbedingt dem Kriterium der kanonischen lexikalischen Repräsentation entspricht. Diese Art der Transkription wurde primär für den Fremdsprachenunterricht entwickelt, um bestimmte Arten der Assimilation zu verdeutlichen, wie etwa „in Bonn“ /in bɔn/ in der Aussprache „im Bonn“ /im bɔn/. Diese Transkriptionsart ergänzt die kanonische phonematische Transkription, kann sie aber nicht ersetzen. Zitate im Textzusammenhang werden mit Schrägstrichen gekennzeichnet.

Enge phonetische Transkription: Die enge phonetische Transkription zeigt mehr phonetische Details der Aussprache an, als für die einfache Unterscheidung von Wörtern notwendig wäre, beispielsweise die Behauchung von stimmlosen Plosiven am Wortanfang („Tanne“ [tʰanə]). Die enge phonetische Transkription ist in der allgemeinen Linguistik, der Soziolinguistik und der Dialektologie sowie in phonetischen Detailuntersuchungen unerlässlich. In der Sprachtechnologie wird die enge phonetische Transkription in der Regel nicht verwendet. Das Sprachsignal wird direkt auf eine kanonische phonematische Transkription abgebildet. Die enge phonetische Transkription erfordert eine intensive Phonetikausbildung, ist sehr zeitaufwändig und bei einer Abbildung von Phonemen in Kontext (z.B. als Diphone oder Tripheme) außer bei sehr auffälligen Allophenen nicht notwendig.

Erweiterte Transkriptionen: Die bisher aufgeführten Transkriptionsarten können um vielerlei weitere Informationen erweitert werden, wie z.B. prosodische Informationen, Informationen über Pausen, Häsitationssignale, Abbrüche und Neustarts, nonverbale Vokalisierungen (Lachen, Seufzen, Weinen, usw.), Lautungen bei Sprachbehinderungen, sowie bei Husten,

Niesen und anderen nichtsprachlichen Geräuschen. Für prosodische Transkriptionssysteme gibt es mehrere Vorschläge, von denen die wichtigsten hier nur genannt werden sollen: die IPA-Symbole (Abbildung 3.15), ToBI (Tones and Break Indices), INTSINT (International Transcription System for Intonation), SAMPROSA (aus demselben europäischen Projekt wie SAMPA, eine Zusammenstellung gebräuchlicher prosodischer Transkriptionssymbole). Im Gegensatz zu den orthographischen oder phonematischen und phonetischen Transkriptionstypen und auch zu den prosodischen Transkriptionskonventionen sind die Symbole für andere Vokalisierungen und Lautungen nicht standardisiert, daher werden hier keine weiteren Hinweise dazu gegeben. Im Internet können allerdings zahlreiche Hinweise auf die Konventionen für solche Lautungen gefunden werden.

Annotation

Zur Analyse der Schallwellen haben Phonetiker und Sprachingenieure vielfältige Software für alle gängigen Betriebssysteme zur Verfügung gestellt, die im Internet erhältlich sind und in einigen Linux-Distributionen zur automatischen Installation bereitgestellt werden. Die bekanntesten freien Software-Werkzeuge für die akustische Sprachsignalanalyse sind *Praat*, *WaveSurfer* sowie *Transcriber*. Auch andere, eher für das Editieren von Musik und Audio-Reportagen vorgesehene freie Software wie *Audacity* eignet sich für das Schneiden, Filtern usw. von Sprachaufnahmen.

Der wichtigste Schritt in der computerphonologischen, computerphonetischen und sprachtechnologischen Korpusanalyse ist die Korpusannotation, auch „labeling“, „time alignment“ oder „Etikettierung“ genannt. Die Annotation in diesem Sinne ist als eineindeutige Relation zwischen Symbolen einer Transkription und Zeitstempeln von Segmenten in einem Sprachsignal definiert.

Ein Beispiel für eine Annotation auf Phonemebene wird in Abbildung 3.3 auf Seite 177 wiedergegeben: Einzelne Phonemsymbole in der Transkription von „der Löwe und die Maus“ werden mit Zeitstempeln versehen, die eine genaue Zuordnung zum Sprachsignal ermöglichen. Die Transkriptionseinheiten, die annotiert werden, hängen von der Fragestellung ab und können Phoneme, Silben, Wörter usw. sein. Üblich sind auch orthographische Annotationen. Die Hilfe-Dokumentation der Praat-Software bietet eine ausgezeichnete Einführung in die Annotation an. Vielfältige Informationen, auch zur prosodischen Annotation, sind im Internet erhältlich, beispielsweise zum ToBI-System oder zum IntSint-System.

Korpuslexikon, Syntheseeinheiten, Sprachmodelle

Die Transkriptionen und Annotationen werden auf verschiedene Weisen weiterverarbeitet, in der Praxis oft mit Skriptsprachen wie z.B. Perl, Python oder einer UNIX-Shell-Sprache.

Ein weiterer wesentlicher Schritt vor allem bei der sprachtechnologischen Korpusbearbeitung ist die Erstellung eines Korpuslexikons, mit Häufigkeitsstatisti-

stiken über die für Untersuchungen relevanten Einheiten und Kombinationen von Einheiten. Aus solchen Korpuslexika werden statistische oder symbolische **Sprachmodelle** für die Suchraumeingrenzung für Wörter in der Spracherkennung erstellt.

3.1.3 Formale Methoden

Die eher signalverarbeitenden formalen Methoden in der Phonetik und die eher symbolverarbeitenden formalen Methoden in der Computerphonologie sind verschieden und werden daher getrennt behandelt.

Formale Methoden in der Phonetik

Die in der Phonetik zur Anwendung kommenden formalen Methoden betreffen einerseits quantitative Parameter des Sprachsignals, andererseits statistische Verfahren zur Analyse des Sprachsignals. Eine ausführliche Einführung in diesen Bereich kann an dieser Stelle nicht geleistet werden (s. aber den Literaturabschnitt 3.1.4).

Hier sollen lediglich die wichtigsten Begriffe in der akustischen Phonetik erläutert werden, weil diese Teildomäne wohl für die meisten Bereiche der heutigen Phonetik und Sprachtechnologie die wichtigste ist.

Die Zeitdomäne

Das Sprachsignal soll zunächst in der Zeitdomäne beschrieben werden, in der die Amplitude als Funktion der Zeit dargestellt wird. Hierzu dient Abbildung 3.7, die einen Auszug aus dem bereits in Abbildung 3.3 dargestellten Signal visualisiert. Abbildung 3.7 zeigt den Übergang vom Vokal [ʊ] (dem zweiten Teil des Diphthongs [au]) auf den Frikativ [s] im Wort „Maus“ [maʊs]. Sehr deutlich zu sehen ist der Unterschied zwischen der regelmäßigen Schwingung des Klangs [ʊ] und dem unregelmäßigen Verlauf des Geräusches [s].

Für die Beschreibung des Sprachsignals in der Zeitdomäne sind die Parameter **Amplitude**, **Intensität**, **Energie** (oder **RMS-Amplitude**), **Periode**, **Frequenz** und **Phase** die wichtigsten Grundbegriffe, sowie für digitalisierte Sprachsignale das **Digitalisieren**, die **Abtastfrequenz** und das **Aliasing**.

Größe: Die Größe, die in der Zeitdomäne als Funktion der Zeit gemessen wird, ist der variable Druck der Schallwellen im Medium Luft (oder einem anderen Medium), der die Bewegungen des Trommelfells oder der Mikrofonmembran verursacht. Der Druck wird im Innenohr in Nervensignale umgesetzt, im Mikrofon in elektrische Potentiale.

Amplitude: Die Amplitude des Drucks ist die Abweichung der Druckstärke vom Ruhewert (Nullwert) und hat bei einer Schallwelle positive und negative Werte um den Nullwert. Die durchschnittliche Amplitude hat somit bei einem Signal, das um den Nullwert symmetrisch ist und vollständige Perioden enthält, den Wert 0. Bei einem nicht symmetrischen Signal heißt die

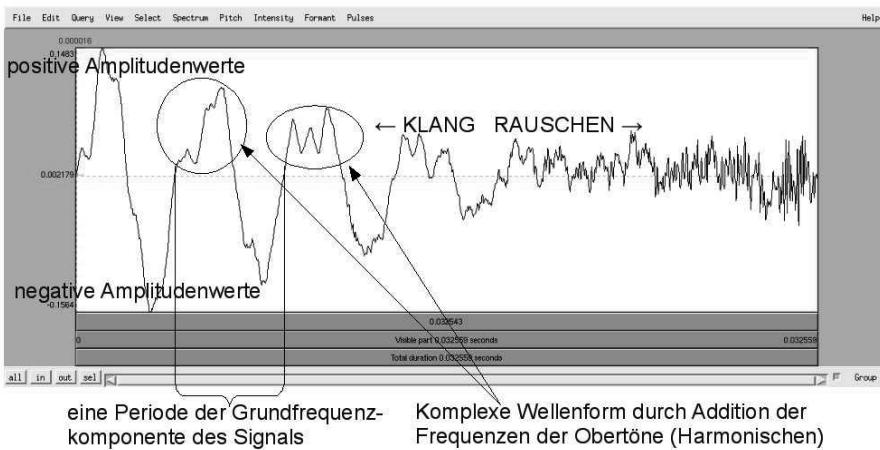


Abbildung 3.7: Übergang von [v] zu [s] in [maus].

Abweichung **additive Konstante** oder **y-Abschnitt** (engl. *offset* oder auch *DC offset*). Das Signal in der Zeitdomäne wird mit der Amplitude als Funktion der Zeit dargestellt:

$$A(\text{signal}_x) = f(t_x)$$

Oszillogramm: Das Oszillogramm ist eine Visualisierung des Amplitudenverlaufs (der Wellenform) in der Zeit. In Abbildung 3.3 wird im oberen Bereich ein Oszillogramm gezeigt. Abbildungen 3.7 und 3.8 (rechts, oben) zeigen ebenfalls Oszillogramme.

Intensität: Die Intensität des Signals ist die Amplitude im Quadrat:

$$I = A^2$$

Intervall: Ein zeitliches Intervall ist eine Zeitspanne (Zeitdifferenz, Zeitunterschied), dargestellt als $t_i - t_{i-1}$, δ_t , d_t , usw.

RMS-Amplitude (RMS-Intensität, Energie): Die RMS-Amplitude entspricht der durchschnittlichen Intensität in einem Intervall t_1, \dots, t_n :

$$E = \sqrt{\frac{\sum_{i=1}^n A(x_i)^2}{n}}$$

Periode, Frequenz, Phase: Die **Periode** eines Signals ist das Intervall einer vollständigen Welle. Die **Frequenz** in Hertz ist die Anzahl der Perioden in einer Sekunde (δ_t ist die **Periodendauer**):

$$f = \frac{1}{\delta_t}$$

Ein Zyklus eines periodischen Signals fängt bei Phase 0 an und durchläuft 360° , bevor der Zyklus neu anfängt. Die Phasen verschiedener Obertöne des Signals müssen nicht unbedingt miteinander übereinstimmen. Wenn die Phasen zweier sonst gleicher Signale sich um 180° unterscheiden, heben sie sich auf. Nach diesem Prinzip funktionieren geräuschneutralisierende Kopfhörer. Die Phase eines Sprachsignals ist im Allgemeinen nicht von großer Bedeutung in der phonetischen Analyse.

Digitalisieren, Abtastfrequenz, Nykvist-Theorem: Das Digitalisieren ist die Messung der Amplitude des Sprachsignals in (gewöhnlich) regelmäßigen Abständen. Die **Abtastfrequenz** (engl. *sampling rate*) ist die Anzahl der Messungen des Signals pro Sekunde in Hertz. Das **Nykvist-Theorem** besagt:

Wenn f die höchste zu messende Frequenz ist, dann muss die Abtastfrequenz mindestens $2f$ sein.

Andernfalls wird die Frequenz nicht korrekt gemessen, weil bei kleineren Abtastfrequenzen **Phantomfrequenzen** erscheinen, die die Messung verfälschen. Die Frequenz $2f$ heißt auch die **Nykvist-Frequenz**.

Die Abtastfrequenz für Audio-CDs beträgt beispielsweise 44100 Hz, aus 2 Gründen:

1. Wenn für die höchste von Menschen wahrnehmbare Frequenz gilt: $f = 22$ kHz, dann: $2f = 44$ kHz.
2. Die Summe der Quadrate der ersten vier Primzahlen ergibt eine geringfügig höhere Zahl als 44 kHz und wurde gewählt, um eine möglichst vielseitige digitale Frequenzteilung ohne aufwändiges Rechnen zu ermöglichen:

$$2^2 + 3^2 + 5^2 + 7^2 = 44100$$

Mit heutigen digitalen Signalverarbeitungstechniken (**DSP-Techniken**) wäre die Berechnungskosten, die die Zahl 44100 ermöglicht, eigentlich nicht mehr notwendig. Die Standardabtastfrequenzen für digitale Audio-bänder sind 48 kHz und 32 kHz (letztere häufig mit einer praktisch nicht wahrnehmbaren aber dennoch verlustbehafteten Signalkompression verbunden und daher nicht unbedingt für phonetische Analysen geeignet). Übliche Abtastraten für die phonetische Signalanalyse sind 16 kHz (wird seltener verwendet) und 22,05 kHz (die Hälfte von 44,1 kHz). Als Standardformat dafür hat sich das WAV-Format der Fa. Microsoft durchgesetzt. Das MP3-Format des Fraunhofer Instituts ist für viele Arten der phonetischen Analyse ungeeignet, weil das Frequenzspektrum entsprechend einem optimierenden Hörmodell verzerrt wird und also verlustbehaftet ist; Grundfrequenz und Formantfrequenzen werden aber erhalten. In speziellen sprachtechnologischen Anwendungen können das MP3-Format und andere komprimierte Formate jedoch vorkommen.

Zeitfenster: Ein Zeitfenster ist eine Funktion, die das Signal in einem bestimmten Intervall transformiert. Die Identitätstransformation ist einfach eine Kopie des Signals in diesem Intervall. Die Funktion einer Fenstertransformation ist häufig, z.B. mit einer Kosinus- oder Gaußfunktion, eine allmähliche Absenkung der Amplitude am Anfang und am Ende des Intervalls zu bewirken, um das Vortäuschen hoher Frequenzen durch ein plötzliches Abschneiden des Signals zu vermeiden. Ein Zeitfenster ist also nicht einfach ein Intervall.

Die Frequenzdomäne

Die Darstellung des Sprachsignals in der Zeitdomäne ist die grundlegende Darstellung. Die zweite wichtige Darstellung ist die **Frequenzdomäne**, die aus der Zeitdomäne unter Verwendung einer Transformation (z.B. Fourier-Transformation) berechnet wird:

Spektralanalyse: Sprachsignale sind komplexe Signale, die durch die Spektralanalyse in ihre Teilkomponenten zerlegt werden. Wenn die Frequenzen im komplexen Signal aus einer Grundfrequenz und deren ganzzahligen Vielfachen bestehen, dann handelt es sich um einen **Klang** (vgl. Vokale). Wenn die Frequenzen im komplexen Signal in keinem einfachen Verhältnis zueinander stehen, handelt es sich um ein **Geräusch** (vgl. Reibelaute). Das Signal kann durch eine Spektralanalyse mittels der **Fourier-Transformation** in ihre einzelnen Frequenzanteile zerlegt werden. Die **Energie** der Komponenten mit den so ermittelten Frequenzen bildet das **Spektrum** und wird als Funktion der Frequenz dargestellt:

$$\text{Intensität}(x) = f(\text{Frequenz}(x))$$

Ein **Spektrogramm** ist eine Aneinanderreihung von Spektren, die eine dreidimensionale Darstellung des Verlaufs der Signalkomponenten in der Zeit ermöglicht.

Fourier-Transformation: Die am häufigsten verwendete Methode der Spektralanalyse ist die Fourier-Transformation, die von der Annahme ausgeht, dass jedes komplexe Signal als die punktweise Addition von reinen Sinusschwingungen unterschiedlicher Frequenz, Phase und Amplitude zusammengesetzt ist (**Fouriersynthese**) bzw. in solche Sinusschwingungen zerlegt werden kann (**Fourieranalyse**). Die Fourier-Transformation kann intuitiv als ein Korrelationsverfahren verstanden werden: Sinusförmige Vergleichssignale mit systematisch variierender Frequenz, Phase und Amplitude werden mit dem zu analysierenden Signal korreliert. Der Korrelationswert zeigt dann den Grad der Übereinstimmung der Frequenz-, Phasen- und Amplitudeneigenschaften des Vergleichssignals mit Komponenten des zu analysierenden komplexen Signals. Die Frequenzen der Teilsignale und deren Intensität werden als Spektrum dargestellt. In der Phonetik und der Sprach-

technologie wird die Phaseninformation meist nicht benötigt. Zur Berechnung der Fourier-Transformation bei digitalen Signalen wird die **Diskrete Fourier-Transformation (DFT)** über die Abtastwerte verwendet, meist in einer effizienten Variante, der **Fast Fourier Transformation (FFT)**, bei der die Punktzahl eine Zweierpotenz sein muss.

Grundfrequenz: Die Grundfrequenz ist die tiefste Frequenz in einem Klang. Die anderen Frequenzen, die in einem Klang ganzzahlige Vielfache der Grundfrequenz sind, sind die Obertöne. Die Grundfrequenz entspricht in etwa dem Höreindruck der Tonhöhe, die die Sprachmelodie (Ton, Akzent, Intonation) bestimmt, und der Phonationsrate der Glottis in der Sprachproduktion. Zur Bestimmung der Grundfrequenz können viele Methoden angewendet werden. In der Zeitdomäne können z.B. die Perioden zwischen Nulldurchgängen, zwischen Signalgipfeln, oder auch zwischen Korrelationsmaxima bei Vergleich eines Teils des Signals mit überlappenden nachfolgenden Teilen des Signals verglichen werden (**Autokorrelation**). In der Frequenzdomäne können z.B. die Abstände zwischen den Obertönen (die Abstände gleichen der Grundfrequenz) mit verschiedenen Methoden berechnet werden.

Formant: Ein Formant ist aus der Perspektive der akustischen Analyse ein Frequenzbereich, in dem Obertöne stärker erscheinen als in anderen Frequenzbereichen. Formanten dürfen nicht mit Obertönen verwechselt werden. Vor allem die Vokale werden durch ihre Formantstruktur charakterisiert. In Abbildung 3.8 werden die ersten drei Formanten des [i:] in „liegen“ [li:gən] als Spektrum (links) und als Spektrogramm (rechts) visualisiert, die in dieser Aufnahme einer weiblichen Stimme bei 330 Hz, 2700 Hz und 3700 Hz liegen. Für die Bestimmung des Vokals sind die ersten beiden Formanten am wichtigsten: Bei [i:] liegen sie weit auseinander. Bei [u:] liegen sie eng beieinander. In Abbildung 3.8 (rechts) wird außerdem die Grundfrequenz gezeigt. Die Formantfrequenzen sind prinzipiell unabhängig von der Grundfrequenz; daher können verschiedene Vokale auf derselben Tonhöhe gesprochen werden, und ein Vokal auf verschiedenen Tonhöhen.

Computerphonetische Methoden

Neben der Signalanalyse können weitere Verarbeitungen von annotierten Daten mit nicht-signalverarbeitenden Methoden vorgenommen werden. Beispiele sind die Erstellung von Korpuslexika und Diphon- und Triphonlisten, Berechnung der relativen Häufigkeit annotierter Einheiten, das Trainieren von Hidden-Markov-Modellen (HMM) in der Spracherkennung, die Analyse von Dauerrelationen zwischen annotierten Silben, und die Grundfrequenzmuster auf betonten Silben. Solche Analysen werden in vielen Arten von Anwendungen verwendet. Im Folgenden werden zwei Ansätze aus diesem Bereich angeführt, die computerlinguistisch besonders interessant sind.

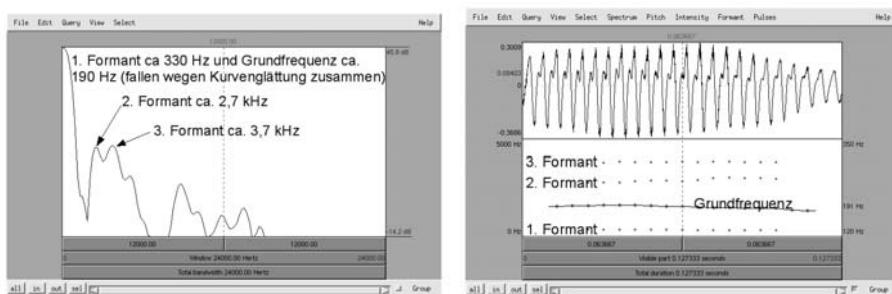


Abbildung 3.8: Spektralanalyse einer weiblichen Stimme. Links: Spektrum von [i:] mit Angabe der ersten drei Formanten. Rechts: Oszillogramm und Visualisierung der Formanten von [i:] sowie Grundfrequenzspur.

Lineare Zeitrelationen: Zur Phonetik des Rhythmus sind sehr viele Arbeiten vorhanden, bei keiner ist es aber jemals gelungen, den **Sprachrhythmus** vollständig als physikalisch-phonetisches Phänomen zu charakterisieren, ohne auf abstraktere linguistische Sprachstrukturen Bezug zu nehmen. Allgemein wird angenommen, dass das sprachliche Rhythmusempfinden eine komplexe kognitive Konstruktionsleistung ist und keine rein physikalische Regelmäßigkeit. Dennoch werden physikalische Maße für Dauerrelationen benötigt.

Einer der bekannteren neueren Maße ist der **Pairwise Variability Index, PVI** (s. Literaturabschnitt 3.1.4), der über die Dauerrelationen benachbarter phonetischer Einheiten berechnet wird (Taktsequenzen, Silbenfolgen, vokalische oder konsonantische Segmentfolgen wurden in der Literatur untersucht). Der PVI ist das Mittel der normierten Differenzen zwischen den Längenunterschieden von relevanten Einheiten. Er basiert auf einer bekannten Formel, die normalerweise die Homogenität einer Wertemenge bestimmen soll. Der PVI wird mit folgender Formel berechnet (zwischen den PVI-Varianten wird hier nicht unterschieden):

$$PVI = 100 \times \sum_1^{n-1} \frac{|d_i - d_{i+1}|}{(d_i + d_{i+1})/2} / (n - 1)$$

(d_i bezeichnet die Dauer einer annotierten Signaleinheit)

Der PVI kann Werte von 0 (gleiche Längen) asymptotisch bis näherungsweise 200 (sehr unterschiedliche Längen) annehmen. Durch Anwendung dieser Formel auf Dauerwerte von vokalischen und intervokalischen Intervallen in Annotationen von Äußerungen in unterschiedlichen Sprachen wurden interessante Verteilungen der Dauerrelationen in diesen Sprachen festgestellt.

Ob die Formel tatsächlich Rhythmus beschreibt, ist bezweifelt worden: Ähnliche Verteilungen lassen sich auch ohne das Sprachsignal allein anhand der Anzahl von Phonemtypen in entsprechenden Sequenzen ermitteln. Zudem setzt die Formel Binarität im Rhythmus voraus, was nicht unbedingt gegeben ist, und lässt kontrastive Vokaldauer außer Acht. Schließlich ist die Formel als Modell zwar vollständig, aber nicht korrekt: Es ist leicht überprüfbar, dass derselbe Indexwert zwar durch alternierende binäre Rhythmen, aber auch aufgrund der Verwendung des absoluten Werts der Dauerdifferenz durch eindeutig arhythmische Sequenzen (z.B. geometrisch ansteigende oder fallende Sequenzen oder Mischungen dieser drei Möglichkeiten) erreicht werden kann. Der PVI wurde als **Rhythmusmodell** eingeführt, ist aber aus den angeführten Gründen als solches ungeeignet. Dennoch kann der PVI aufschlussreiche empirische Informationen über die temporale Struktur von Äußerungen liefern.

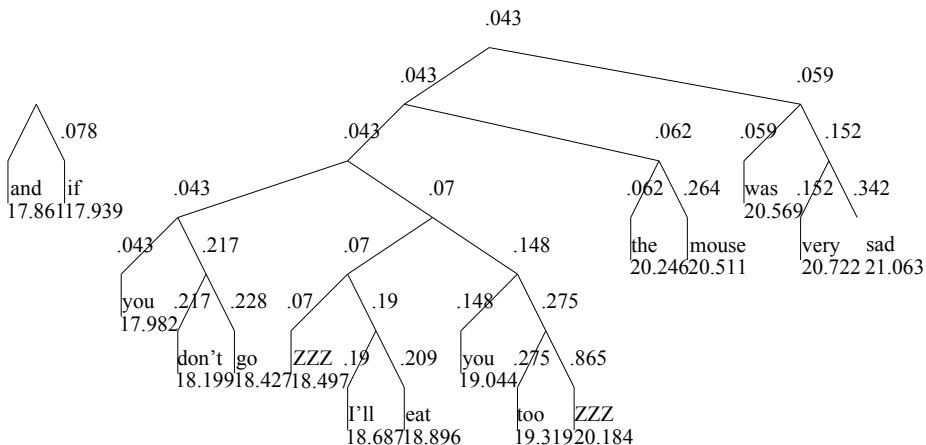


Abbildung 3.9: Durch numerisches ‘Parsen’ berechnete temporale Bäume über Wörter.

Zeitbäume: Dauerunterschiede zwischen benachbarten Einheiten können auch dazu verwendet werden, komplexere hierarchische Zeitrelationen als Zeitbäume zu ermitteln. Um eine Baumstruktur aufzubauen, werden im Gegensatz zur Berechnung des PVI nicht die absoluten Werte, sondern die rohen Werte der Dauerdifferenzen verwendet. Die Unterschiede zwischen positiven und negativen Differenzen werden gezielt eingesetzt, um wie bei einem Parser einen Baum aufzubauen. Eine Anwendung dieses Verfahrens, die interessante Korrelate mit syntaktischen Strukturen zeigt, wird in Abbildung 3.9 wiedergegeben.

Formale Methoden in der Computerphonologie

Merkmale, Attribute, Generalisierung, Defaults:

Auf die Möglichkeit, distinktive Merkmale als Attribut-Wert-Paare darzustellen, wurde bereits eingegangen. In der Phonologie kann auch zwischen markierten und unmarkierten Werten eines Attributs unterschieden werden. In einem solchen Fall stellt der unmarkierte Wert z.B. den häufigsten Wert in Korpora, in Lexika oder unter den Flexionsformen eines Worts dar. Z.B. können durch die Neutralisierung der Wortunterscheidung an bestimmten Stellen in Silben oder Wörtern Elemente eines Spezifikationspaars mehrdeutig erscheinen.

Im Deutschen bewirkt beispielsweise die Auslautverhärtung von Obstruenten (Plosiven und Frikativen) eine Neutralisierung in den homophonen Formen „Rad“ /ra:t/, „Rat“ /rat/. Dass die Stämme sich morphophonematisch unterscheiden, zeigen die flektierten Formen „Rades“ /ra:dəs/ und „Rates“ /ra:təs/. Aufgrund solcher Neutralisierungen gilt [- stimmhaft] als unmarkiert im Auslaut, [+ stimmhaft] als markiert. Es kann aber vorkommen, dass in anderen Kontexten die andere Spezifikation als unmarkiert gilt: In einigen deutschen Dialekten wird zwischenvokalisch /d/ sowohl für /d/ als auch für /t/ verwendet, z.B. „Kleid“ /klart/ aber „bekleidet“ /bøglæidət/. In der **Generativen Phonologie** werden daher als Spezifikationen auch [u stimmhaft] und [m stimmhaft] verwendet, um diese Generalisierung zu erfassen, aus denen dann kontextspezifisch die entsprechenden Spezifikationen [+ stimmhaft] und [- stimmhaft] je nach Position in Silbe oder Wort abgeleitet werden.

Nicht alle Merkmale sind bei allen Phonemen gleichermaßen relevant. Nasale Konsonanten sind in den meisten Sprachen normalerweise stimmhaft; die Spezifikation des Merkmals [+ nasal, + stimmhaft] kann also zu [+ nasal] vereinfacht werden, wenn eine **Redundanzregel** (logisch gesehen eine Konditionalaussage: wenn nasal, dann stimmhaft) eingeführt wird:

$$[+ \text{nasal}] \rightarrow [+ \text{stimmhaft}]$$

Das Merkmal [\pm stimmhaft] kann unspezifiziert bleiben; das Merkmalsbündel bleibt also underspezifiziert.

Wenn binäre (oder auch mehrwertige) Merkmale als Attribut-Wert-Paare modelliert werden, können sie mit den Operationen der Attribut-Wert-Logik (Unifikation, Generalisierung usw.) bearbeitet und computerlinguistisch implementiert werden. Solche Darstellungen wurden bereits eingeführt.

Die Modellierung von Merkmalen als Attribut-Wert-Paare eröffnet aber weitere Formalisierungsmöglichkeiten. Die Markiert-Unmarkiert-Gewichtung ist auch mit defaultlogischen Defaultlogik Mitteln behandelt und durch **Default-Unifikation** und **Default-Vererbung** computerphonologisch bearbeitet worden. In der defaultlogisch motivierten **Default-Vererbungssprache DATR** können beispielsweise Markiertheitsverhältnisse durch Unterspezifikation und Redundanzregeln durch Vererbung dargestellt werden:

KONSONANT:

<KONSONANTISCH>	==	+
<VOKALISCH>	==	-
<KONTINUIERLICH>	==	-
<STIMMHAFT>	==	-
<>	==	PHONEM.

PHONEM-P:

<LABIAL>	==	+
<>	==	KONSONANT.

PHONEM-B:

<STIMMHAFT>	==	+
<>	==	PHONEM-P.

Dieses Fragment der DATR-Implementierung eines **Vererbungsgraphen** modelliert folgende phonologische Generalisierungen:

1. Das Phonem /b/ erbt alle Merkmalswerte vom Phonem /p/, außer dem Stimmhaftigkeitswert [+ stimmhaft], der direkt zugewiesen wird und damit den Defaultwert überschreibt.
2. Das Phonem /p/ erbt alle Merkmalswerte von der natürlichen Klasse der Konsonanten, außer dem Wert [+ labial], der direkt zugewiesen wird und damit den Defaultwert überschreibt.
3. Die natürliche Klasse der Konsonanten spezifiziert alle unmarkierten Default-Werte der Konsonanten, alles Weitere wird von der hier nicht weiter spezifizierten Klasse der Phoneme geerbt. Konsonanten haben demnach typischerweise folgende Merkmale:

KONSONANT	
+	konsonantisch
-	vokalisch
-	kontinuierlich
-	stimmhaft

Mit solchen Mitteln können ausdruckstarke lexikalische Relationen formalisiert und implementiert werden, die den Aufbau konsistenter Lexika unterstützen.

Reguläre Modelle

In den drei Domänen der Phonetik, der Phonologie und der Prosodie (und auch in der Morphologie) sind **reguläre Modelle** (d.h. **endliche Automaten** (Finite State Automaton, FSA), **endliche Übergangsnetzwerke**, **endliche Transduktoren** (Finite State Transducer, FST), **reguläre Grammatiken**, **reguläre Ausdrücke**) zu Standardmodellen für die Modellierung und Operationalisierung von kompositorischen Eigenschaften von Lautsequenzen geworden.

In der Computerphonologie werden reguläre Modelle zur Modellierung folgender Strukturen eingesetzt:

1. Silbenstrukturen,
2. phonotaktische Regeln (Morphemstrukturregeln, Redundanzregeln),
3. phonetische Interpretationsregeln,
4. die GEN-Komponente der Optimalitätstheorie (Generator des Suchraums für phonetische Interpretationen),
5. die EVAL-Komponente der Optimalitätstheorie (Constraintfilter zur Einschränkung des Suchraums für phonetische Interpretationen),
6. trainierbare, gewichtete stochastische Automaten in der Form von Hidden-Markov-Modellen in der Sprachtechnologie (siehe Unterkapitel 3.2).

Die wichtigsten Modellierungskonventionen, die die Verwendung von regulären Modellen in der Phonologie nahelegen, sind:

1. Die maximale Silbenlänge in allen Sprachen ist klein (zwischen 2 und 8); es wird also keinerlei Rekursion benötigt.
2. Die Phoneminventare in allen Sprachen sind endlich (und klein, mit ca. 20 bis 50 Elementen).
3. Das Kombinationspotential der Phoneme in Silben ist sehr beschränkt und kann z.B. mit endlichen Übergangsnetzwerken übersichtlich dargestellt werden.

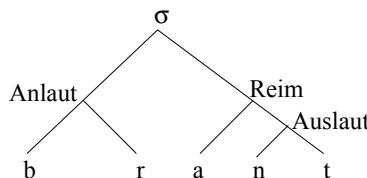


Abbildung 3.10: Baumgraph als Strukturbeschreibung einer Silbe.

4. Obwohl in der Phonologie oft Baumgraphen zur Darstellung von Silbenstrukturen verwendet werden, was einen komplexeren, kontextfreien Formalismus nahelegen könnte, haben diese Bäume eine maximale (und kleine) Tiefe (Abbildung 3.10).
5. Solche Baumgraphen können auch die linearen Einschränkungen quer zu den Baumverzweigungen nicht direkt oder anschaulich ausdrücken, wohingegen reguläre Modelle für diese Problematik optimal geeignet sind.
6. Die Interpretation von Phonemen in linearen Kontexten kann durch endliche Transduktoren modelliert werden, entweder als Kaskaden von hintereinander geschalteten Automaten oder als parallele Automaten.

7. Auf Merkmalssystemen beruhende phonologische Theorien können von endlichen **Mehrbandautomaten** modelliert werden.

Ein Beispiel für die Verwendung von regulären Modellen ist die Formalisierung von phonotaktischen Regeln:

$$\begin{array}{ll}
 \text{Regelnotation:} & K \rightarrow \int / \$ - \left\{ \begin{array}{c} \left\{ \begin{array}{c} p \\ t \\ k \end{array} \right\} \left\{ \begin{array}{c} l \\ r \end{array} \right\} \\ m \\ n \end{array} \right\} \\
 \text{Regulärer Ausdruck:} & \int (((p|t|k) (r|l)) \mid (m|n)) \\
 \text{Rechtslineare Grammatik:} & \text{Silbe} \rightarrow \int \text{KonSeq-1} \\
 & \text{KonSeq-1} \rightarrow \left\{ \begin{array}{c} p \text{ KonSeq-2} \\ t \text{ KonSeq-2} \\ k \text{ KonSeq-2} \\ m \\ n \end{array} \right\} \\
 & \text{KonSeq-2} \rightarrow \left\{ \begin{array}{c} r \\ l \end{array} \right\} \\
 \text{Reguläre Menge:} & \{ \text{spr, str, skr, spl, stl, fkl, fm, fn} \}
 \end{array}$$

Die durch solche Morphemstrukturregeln oder Redundanzregeln angegebenen Vorkommensbeschränkungen sind wohl vollständig, indem alle Silben beschrieben werden, aber nicht korrekt, indem sie übergenerieren und Ketten beschreiben, die als Silben nicht vorkommen. Beispielsweise ist der Silbenanfang /stl/ im Deutschen nicht möglich, dies wird aber nicht direkt durch eine Redundanzregel ausgedrückt. Hierfür eignet sich eine als vollständiges Übergangsnetzwerk ausgeführte Beschreibung der ganzen Silbe eher als einzelne Regeln für Silbenteile. Mit einem solchen regulären Modell kann anhand eines relativ leicht zu implementierenden Interpreters die vollständige reguläre Menge auf einfache Weise formal und empirisch überprüft werden.

Abbildung 3.11 zeigt als Beispiel eines solchen Netzwerks ein nahezu vollständiges endliches Übergangsnetzwerk für englische Silben. Aus der Übergangskombinatorik lässt sich errechnen, dass die reguläre Menge, die durch dieses Netzwerk beschrieben wird, ca. 25.000 potentielle Silben des Englischen enthält.

Ein solches reguläres Modell der Phonotaktik lässt sich auf einfache Weise als Modell der phonetischen Interpretation verwenden, indem daraus ein endlicher Transduktor gemacht wird und die korrekten Allophone auf den entsprechenden Übergängen ihren Phonemen zugeordnet werden.

Phonetische Interpretationsregeln werden auch einzeln durch endliche Transduktoren modelliert. Folgende Regel beschreibt die Interpretation des deutschen Phonems /p/ in zwei verschiedenen Kontexten:

$$/p/ \rightarrow \left\{ \begin{array}{c} [p] / \$ s - \\ [p^h] \end{array} \right\}$$

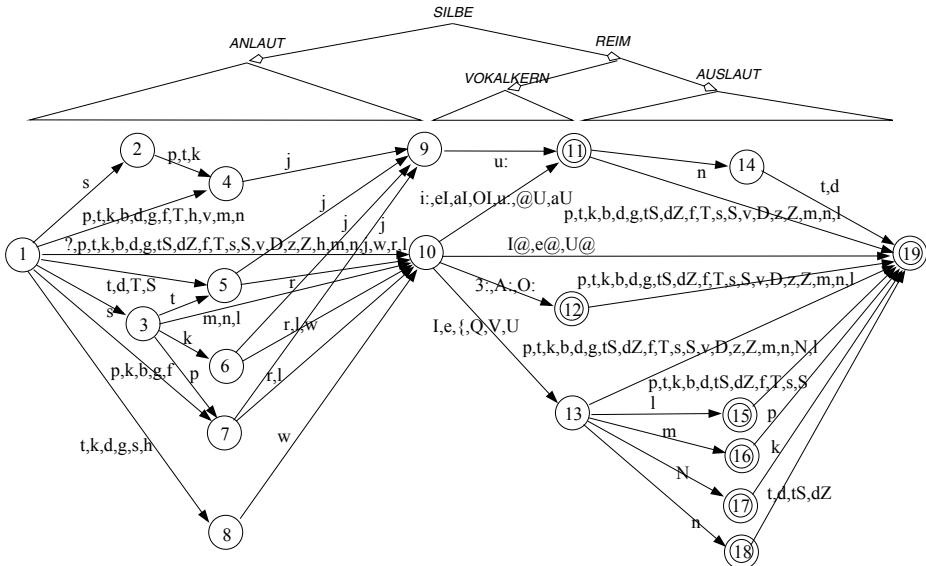


Abbildung 3.11: Endliches Übergangsnetzwerk als Grammatik für englische Silben mit Baumgraph als Generalisierung über die Graphstruktur (Beschriftung in SAMPA-Symbolen).

Die Reihenfolge der Regelalternativen ist defaultlogisch zu verstehen und bedeutet: Nach silbeninitialem /ʃ/ wird /p/ nicht behaucht, sonst wohl (oder, in der umgekehrten Reihenfolge: Typischerweise wird /p/ behaucht, nach /ʃ/ aber nicht). Als **Silbengrenze** wird hier „\\$“ verwendet.

Die Abbildung 3.12 zeigt einen Auszug aus einem endlichen Transduktor, der diese Regel modelliert. Am Silbenanfang läuft der endliche Transduktor an. Sollte ein /ʃ/ gefunden werden, wird es identisch übersetzt und es wird ein Plosiv gesucht. Wenn ein stimmloser Plosiv gefunden wird, wird dieser unbehaucht übersetzt und der Automat dann beendet, sonst wird der Automat direkt beendet. Sollte ein /ʃ/ nach dem Silbenanfang nicht gefunden werden, sondern ein stimmloser Plosiv, wird dieser behaucht übersetzt und der Transduktor dann beendet, sonst wird der Transduktor direkt beendet. Der vollständige Transduktor kann iterativ angewendet werden und relevante Kontexte ignorieren.

Im Ansatz von Kaplan und Kay werden solche Automaten in Kaskaden hintereinander angeordnet, um die Ableitung einer phonetischen Interpretation zu modellieren, wie dies auch für die Generative Phonologie möglich ist: Die Ausgabe eines Automaten bildet die Eingabe für den nächsten. Eine solche Kaskade kann durch eine Operation der Komposition zu einem einzigen Automaten zusammengesetzt werden. Im Ansatz von Koskenniemi, der als **Zweiebenenphonologie** (vgl. auch die Zweiebenenmorphologie) bekannt ist, werden endli-

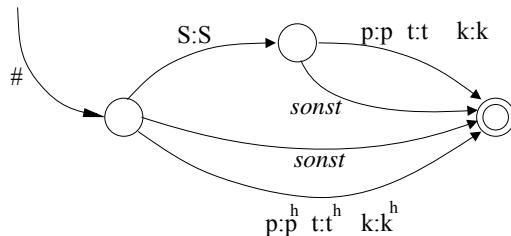


Abbildung 3.12: FST als partielle Modell der Plosivbehauchungsregel („sonst“ bedeutet die Komplementmenge der Phoneme, die auf den anderen Übergängen von einem Knoten erscheinen; der Übersicht halber werden die /p, t, k/-Übergänge zu einem Übergang zusammengefasst; „S“ bedeutet [ʃ]).

che Transduktoren verwendet, die parallel zueinander angewendet werden und aus logischer Sicht als eine Konjunktion von linearen Constraints verstanden werden. Die parallel anzuwendenden endlichen Transduktoren können ebenfalls durch eine Operation der Komposition zu einem einzelnen großen Automaten automatisch konvertiert werden.

Auch die Optimalitätstheorie lässt sich mit regulären Modellen modellieren. Die Grundidee der Optimalitätstheorie ist, dass die Abbildung von der phonologischen auf die phonetische Ebene nicht deterministisch vorgegeben ist, sondern dass ausgehend von einer phonologischen (lexikalischen) Repräsentation alle phonetischen Repräsentationsmöglichkeiten in einer Generatorkomponente GEN frei generiert werden, die dann durch eine geordnete (*ranked*) Menge von universellen Constraints CON in einer Evaluationskomponente EVAL gewichtet werden, die die Anzahl der Constraint-Verletzungen registriert, woraus schließlich die Übersetzung mit den wenigsten Constraint-Verletzungen als die optimale Übersetzung gewählt wird. Auf diese Weise wird der Suchraum für phonetische Interpretationen durch die einzelnen Constraints Schritt für Schritt verkleinert. Die Methode stammt aus der Constraintlogik und stellt im Prinzip einen Formalismus mit einer klaren Semantik dar. Karttunen hat als erster festgestellt, dass die einzelnen Constraints in der Constraint-Menge CON wie phonologische Regeln durch endliche Transduktoren modellierbar sind, ergänzt durch eine zusätzliche Default-Operation. Dieser Modellierungsansatz ist seitdem in der **Finite State Optimality Theory** vielfach angewendet worden, auch für die Modellierung der GEN-Komponente.

Ein wichtiger, aber bislang weniger bekannter Anwendungsbereich für reguläre Modelle ist die Prosodie, sowohl auf Satz- und Diskursebene als Intonationsmodelle, als auch in der phonetischen Interpretation von Tonfolgen in Tonsprachen.

Reguläre Modelle für die Intonation wurden in den 1970er Jahren von Fujisaki für Japanisch sowie von der niederländischen Arbeitsgruppe am Eindhovener Instituut voor Perceptie Onderzoek der Fa. Philips für Niederländisch und eine Reihe anderer Sprachen entwickelt. Das bekannteste reguläre **Intonationsmodell** wurde 1980 von Pierrehumbert für das Englische entwickelt (eine vereinfachte Version wird in Abbildung 3.13 gezeigt). Das Terminalvokabular des Modells besteht aus einer Relation über eine Menge von Tonbuchstaben {H, L} (für Hoch- und Tieftöne) und einer Menge diakritischer Zeichen {%, *}, die den Grenzton einer linguistischen Einheit („%“) oder eine Silbenbetonung („*“) kennzeichnen.

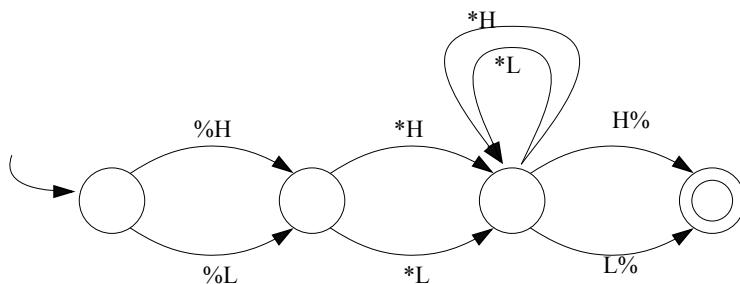


Abbildung 3.13: Vereinfachter FST für die Intonationsmodellierung.

Tonsequenzen in typologisch unterschiedlichen Tonsprachen wurden ebenfalls mit regulären Modellen beschrieben. In den meisten Niger-Kongo-Sprachen von West-, Zentral- und Südafrika werden Wörter nicht nur durch Phonemsequenzen voneinander unterschieden, sondern auch durch Töne — Silbenmelodien — mit phonematischer Funktion. Beispielsweise bedeutet in der Anyi-Sprache (Elfenbeinküste) das Wort „anouman“ /anumā/ mit steigender Tonkontur „Vogel“ und mit fallender Tonkontur „gestern“. Die Konturen werden als Sequenzen einzelner Töne analysiert, die eine bestimmte Tonhöhe relativ zu vorangegangenen **Tönen** einnehmen. Abbildung 3.14 zeigt einen endlichen Transduktoren mit drei verschiedenen Kantenbeschriftungen, die die Operationen über solche Sequenzen für typische afrikanische Zweitonsprachen anzeigen (mit Namen der Tonregeln, die in der Literatur geläufig sind; phonetischen Interpretationen, die von den Übergängen des endlichen Transduktors modelliert werden; Dreibandoperationen, die numerische Werte und Operationen anzeigen, die hier nicht weiter kommentiert werden).

Die numerische Beschriftung erzeugt eine Annäherung an den Grundfrequenzverlauf, die weiterverarbeitet werden muss, um eine realistische Detailkontur, z.B. für die Sprachsynthese, zu erzeugen. Ein solches Modell kann auch mit einigen Modifikationen entsprechend Abbildung 3.5 für die Berechnung von Grundfrequenzverläufen in Akzent- bzw. Intonationssprachen verwendet werden. Für die Detailberechnung der Grundfrequenz werden jedoch komplexere Modelle, z.B. die Modelle von Fujisaki oder Hirst, angewendet.

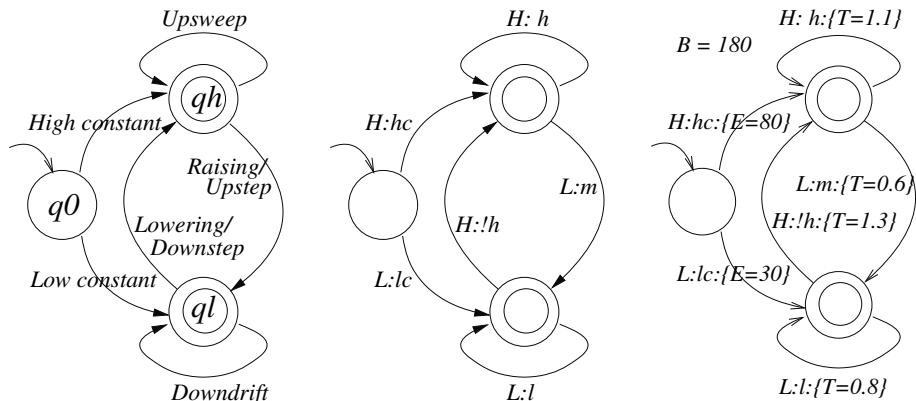


Abbildung 3.14: Endliche Transduktoren für die Tonsequenzierung in Niger-Kongo-Sprachen. (Großbuchstaben für phonologische Eingabetöne: H = Hochton, L = Tiefton; Kleinbuchstaben für phonetische Ausgabetöne: hc = konstante höhere Ansatzfrequenz, h = evtl. steigende Sequenz von höheren Frequenzen, m = angebogene tiefere Frequenz, l = evtl. fallende Sequenz von tieferen Frequenzen, !h = heruntergesetzte „downstepped“ höhere Frequenz).

3.1.4 Zusammenfassung und weitergehende Lektüre

In diesem Beitrag werden zentrale Aspekte der Phonetik und Phonologie soweit besprochen, wie sie für gängige Forschungs- und Entwicklungsarbeiten in der Computerlinguistik und Sprachtechnologie erforderlich sind. Der Beitrag fängt mit theoretischen Konzepten aus der Phonetik und Phonologie an und stellt computerlinguistische Ansätze als Modelle für diese Konzepte vor. Die inhaltlichen linguistischen Fragestellungen werden als Aufgabenbereiche dargestellt, für die Lösungen mit empirischen und formalen computerlinguistischen, phonetischen und sprachverarbeitenden Methoden angeboten werden.

Weitere Informationen zu den computerlinguistischen und sprachtechnologischen Modellen, die in diesem Beitrag vorkommen, werden in anderen Kapiteln des Handbuchs besprochen, insbesondere im Unterkapitel 3.2.

Dieser Beitrag konkurriert nicht mit der reichhaltigen, hauptsächlich englischsprachigen Einführungs- und Handbuchtiteratur zur computerlinguistischen und sprachtechnologischen Modellierung in der Phonologie und Phonetik. Eine Auswahl dieser spezialisierten technischen Literatur wird als weiterführende Lektüre zur Phonetik, Phonologie, Prosodie sowie zu einigen der spezielleren Modellierungstechniken in diesen Bereichen im Folgenden, nach Themen gegliedert, angeführt.

Phonologie: Einen Überblick über neuere Entwicklungen in der Phonologie bietet Hall (2000). Einen anspruchsvollen Einstieg in die Computerphonologie, vor allem unter Berücksichtigung regulärer Modelle, mit Anwendungen in der Sprachtechnologie, gibt Carson-Berndsen (1998).

Phonetik: Einen ersten Einstieg in die Phonetik bieten Pompino-Marschall (2003) und Ashby und Maidment (2005). Wesentlich mehr Details, recht anschaulich erklärt, sind in Reetz (2003) zu finden, während Coleman (2005) einen eher technischen Zugang bietet.

Prosodie: Die Sammelbände (Cutler und Ladd 1983) und (Gibbon und Richter 1984) beschreiben Ergebnisse der klassischen interdisziplinären Prosodieforschung in Überblicken. Neuere Forschungen aus einer phonologischen Perspektive werden in Ladd (2008) und aus interdisziplinären Perspektiven in Sudhoff et al. (2006) präsentiert.

Empirische Methoden, Ressourcen: Die aus dem europäischen EAGLES-Standardisierungsprojekt entstandenen Handbücher (Gibbon et al. 1997 und Gibbon et al. 2000) bieten einen systematischen Überblick über empirische Methoden und Evaluationsverfahren in der Sprachtechnologie, die auch für empirische Verfahren in der Computerlinguistik relevant sind. In Draxler (2008) werden sehr detaillierte Angaben zur Untersuchung von Korpora gesprochener Sprache angeboten.

Formale Methoden: Das Standardwerk (Jurafsky und Martin 2009) enthält eine Fülle von Angaben zu formalen Methoden in vielen Bereichen der Sprachtechnologien. Einen praktischen Zugang zum Programmieren (mit Python) für viele Bereiche der Computerlinguistik einschließlich Aspekte der Computerphonologie ist in Bird et al. (2009) zu finden.

Sprachtechnologien: Integrative Ansätze zu verschiedenen Teildisziplinen in den Sprach- und Texttechnologien werden in den Sammelbänden (Wahlster 2000 und Wahlster 2006) beschrieben.

THE INTERNATIONAL PHONETIC ALPHABET (revised to 2005)

CONSONANTS (PULMONIC)

© 2005 IPA

	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b		t d		t̪ d̪	c ɟ	k g	q ɢ		χ ʁ	ʔ
Nasal	m	n̪	n		n̪	ŋ		ŋ		N	
Trill	B		r							R	
Tap or Flap		v̪	f		t̪						
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ɟ	x ɣ	χ ʁ	h ɬ	h f
Lateral fricative			ɬ ɭ								
Approximant		v̪	ɹ		ɻ	ɺ	ɻ	ɻ	ɻ		
Lateral approximant			ɻ		ɻ	ɻ	ɻ	ɻ	ɻ		

Where symbols appear in pairs, the one to the right represents a voiced consonant. Shaded areas denote articulations judged impossible.

CONSONANTS (NON-PULMONIC)

Clicks	Voiced implosives	Ejectives
ʘ Bilabial	b̪ Bilabial	,
Dental	d̪ Dental/alveolar	p̪ Bilabial
! (Post)alveolar	f̪ Palatal	t̪ Dental/alveolar
ǂ Palatoalveolar	g̪ Velar	k̪ Velar
ǁ Alveolar lateral	G̪ Uvular	s̪ Alveolar fricative

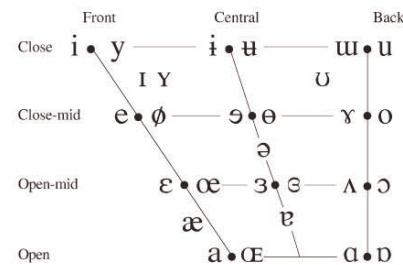
OTHER SYMBOLS

ʍ Voiceless labial-velar fricative	ç Z Alveolo-palatal fricatives
w Voiced labial-velar approximant	j Voiced alveolar lateral flap
ɥ Voiced labial-palatal approximant	ʃ Simultaneous ʃ and x
h Voiceless epiglottal fricative	
ɸ Voiced epiglottal fricative	Affricates and double articulations can be represented by two symbols joined by a tie bar if necessary.
χ Epiglottal plosive	k̪ t̪ ts

DIACRITICS Diacritics may be placed above a symbol with a descender, e.g. ȫ

o Voiceless	n̪̄ d̪̄ ..	Breathy voiced	b̪̄ ā̪ ..	Dental	t̪̄ d̪̄ ..
χ Voiced	ʂ̄ t̪̄ ..	Creaky voiced	b̪̄ ā̪ ..	Apical	t̪̄ d̪̄ ..
h Aspirated	t̪̄ h̄ ..	Lingualabial	t̪̄ d̪̄ ..	Laminal	t̪̄ d̪̄ ..
ɔ More rounded	ɔ̄	W Labialized	t̪̄ w̄ d̪̄ w̄ ..	Nasalized	ɛ̄
ç Less rounded	ç̄	J Palatalized	t̪̄ j̄ d̪̄ j̄ ..	Nasal release	d̪̄ n̄
ü Advanced	ǖ	Y Velarized	t̪̄ ȳ d̪̄ ȳ ..	Lateral release	d̪̄ l̄
œ Retracted	œ̄	Ȳ Pharyngealized	t̪̄ ȳ d̪̄ ȳ ..	No audible release	d̪̄ ..
œ̄ Centralized	œ̄ ..	Velarized or pharyngealized	œ̄ ..		
œ̄ Mid-centralized	œ̄ ..	Raised	œ̄ ..	(œ̄ = voiced alveolar fricative)	
œ̄ Syllabic	œ̄ ..	Lowered	œ̄ ..	(œ̄ = voiced bilabial approximant)	
œ̄ Non-syllabic	œ̄ ..	Advanced Tongue Root	œ̄ ..		
œ̄ Rhoticity	œ̄ ..	Retracted Tongue Root	œ̄ ..		

VOWELS



Where symbols appear in pairs, the one to the right represents a rounded vowel.

SUPRASEGMENTALS

- ‘ Primary stress
- ‘ Secondary stress
- ‘ Long
- ‘ Half-long
- ‘ Extra-short
- ‘ Minor (foot) group
- ‘ Major (intonation) group
- ‘ Syllable break .i.ækt
- ‘ Linking (absence of a break)

TONES AND WORD ACCENTS LEVEL CONTOUR

é or ē	Extra high	é or ē	Rising
é	High	é	Falling
é	Mid	é	High rising
é	Low	é	Low rising
é	Extra low	é	Rising-falling
↓	Downstep	↗	Global rise
↑	Upstep	↘	Global fall

Abbildung 3.15: Symboltabelle der IPA: Konsonantensymbole, Vokalsymbole, Spezialsymbole, diakritische Zeichen zur Modifikation von Artikulationsort und -art.

3.2 Verarbeitung gesprochener Sprache

Bernd Möbius und Udo Haiber

Im vorangehenden Unterkapitel 3.1 wurde die Phonetik als die wissenschaftliche Disziplin vorgestellt, die sich mit den physiologischen und akustischen Bedingungen der menschlichen Sprachverarbeitung aus theoretisch-(computer)linguistischer Sicht befasst. Phonetische Modelle der Sprachproduktion und -perzeption sowie akustische Modellierungen des Sprachsignals sind Voraussetzungen für die Verarbeitung der gesprochenen Sprache in technischen Systemen, und dieses Unterkapitel behandelt die Methoden der beiden Technologien, die als Analogien zur menschlichen Sprachwahrnehmung und Sprachproduktion betrachtet werden können, nämlich die automatische Spracherkennung und die Sprachsynthese.

Die Performanz der Systeme der automatischen Spracherkennung und -synthese ist inzwischen auf einem Niveau angelangt, auf dem ihre kommerzielle Nutzung zunehmend den menschlichen Alltag durchdringt. In der Informationsgesellschaft beruht der Erfolg der Sprachtechnologie nicht zuletzt auf der umfassenden Verbreitung des öffentlichen Telefon- und Mobilfunknetzes, des für menschliche Benutzer effektivsten technischen Kommunikationssystems. Dieses Netz ist die Voraussetzung, praktisch zeit- und ortsgesetzten Zugang zu rechnergestützten Dienstleistungen zu erlangen. Die vielfältigen Anwendungs- und Einsatzmöglichkeiten der Sprachein- und -ausgabetechnologien werden im eigenen Unterkapitel 5.4 ausführlich vorgestellt.

Spracherkennung und Sprachsynthese werden im Allgemeinen nur als Forschungssysteme unabhängig voneinander entwickelt und evaluiert. In praktischen Anwendungen sind sie zumeist keine eigenständigen Systeme, sondern in ein Dialogsystem eingebunden. In diesem Unterkapitel sollen sie jedoch aus systematischen Gründen als separate Systeme der Sprachtechnologie dargestellt werden. Dialogsysteme wiederum werden im Unterkapitel 5.5 genauer besprochen.

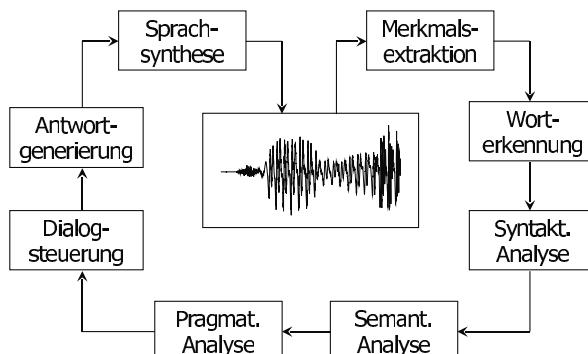


Abbildung 3.16: Blockdiagramm eines Dialogsystems nach Kompe (1979)

Abbildung 3.16 zeigt den Aufbau eines typischen Dialogsystems als Blockdiagramm. Ausgehend von einer sprachlichen Äußerung in Form eines akustischen Signal extrahiert der Spracherkenner Merkmale, die Abschnitte des Sprachsignals in einen systematischen Zusammenhang mit linguistischen Einheiten (Lauten, Silben, Wörtern) bringen. Auf der Grundlage eines Lexikons erstellt der Erkenner hypothetische Sequenzen von Wörtern, die dem System bekannt sind. Diese Sequenzen werden mit Hilfe syntaktischer Einschränkungen auf grammatisch wohlgeformte Wortfolgen reduziert. Nach einer mehr oder weniger gründlichen semantischen und pragmatischen Analyse, also einer weiteren Reduktion von Wortfolgen, die linguistisch sinnvoll und dem kommunikativen Kontext angemessen sind, inferiert das System die Bedeutung des gesprochenen Satzes und die Intention – und neuerdings in manchen Systemen sogar den emotionalen Zustand – des Sprechers und generiert daraufhin seine eigene sprachliche Ausgabe, die durch die Sprachsynthese in Form einer neuen lautsprachlichen Äußerung ausgegeben wird.

Dieses Unterkapitel ist wie folgt strukturiert. Im folgenden Abschnitt über die Methoden der automatischen Spracherkennung werden zunächst der typische Aufbau eines Erkenners und die Aufgaben seiner Komponenten vorgestellt. Es folgt die Darstellung der wichtigsten Methoden der Modellierung der gesprochenen Sprache für die Zwecke der Spracherkennung, und zwar aufgeteilt in die Modellierung der Eigenschaften des akustischen Signals und der Wahrscheinlichkeit möglicher Wortfolgen. Der anschließenden Abschnitt gibt einen Überblick über die typische Struktur von Sprachsynthesesystemen. Auch in diesen Systemen lässt sich eine Unterteilung in Symbolverarbeitung und Signalverarbeitung vornehmen. Erstere bedient sich üblicherweise linguistischer Modelle und computerlinguistischer Methoden, denen in diesem Abschnitt besondere Beobachtung geschenkt wird. Die zum Einsatz kommende Signalverarbeitung ist relativ unspezifisch für die Sprachsynthese. Ihre Grundlagen werden in anderen Teilen dieses Buches besprochen, insbesondere in den Unterkapiteln 3.1 und 2.4. Strategien zur Erzeugung des synthetischen Sprachsignals, speziell die Formantsynthese und die konkatenative Synthese, werden im Unterkapitel 5.4 vorgestellt.

3.2.1 Spracherkennung

Das akustische Sprachsignal enthält Informationen nicht nur über den linguistischen Inhalt der Äußerung – ihre „Bedeutung“ –, sondern auch über den Sprecher und über die akustischen Bedingungen, unter denen es produziert wurde, z. B. die Raumcharakteristika oder Umgebungsgeräusche. Man könnte sagen, dass es die Aufgabe der Spracherkennung ist, den linguistischen Inhalt zu extrahieren. Tatsächlich wird in aktuellen Systemen aber zunehmend auch sprecherspezifische Information mit ausgewertet, und in speziellen Anwendungen wie der Sprecheridentifikation oder -verifikation steht diese Information sogar im Vordergrund.

Struktur eines automatischen Spracherkenners

Automatische Spracherkennung wird häufig als ein Prozess des **Musterabgleichs** charakterisiert. Diese Einschätzung beruht auf dem vorherrschenden Verfahren, mit dem der Erkenner ein unbekanntes sprachliches Muster, also die zu erkennende sprachliche Äußerung, mit einer Reihe von gespeicherten Referenzmustern vergleicht. Die Referenzmuster werden zuvor in einer Lern- oder Trainingsphase erzeugt. Die Entscheidung, welche sprachliche Äußerung aktuell vorliegt, trifft der Erkenner anhand eines Ähnlichkeitsmaßes, das dem Vergleich zwischen dem aktuellen Muster und den Referenzmustern zugrundeliegt. Die automatische Spracherkennung bedient sich also vorwiegend Techniken der statistischen und strukturellen Mustererkennung, zum Teil aber greift sie auch auf wissensbasierte Prinzipien, vor allem aus der Linguistik und Phonetik, zurück.

Der Prozess der Spracherkennung kann analytisch in eine Folge von Schritten zerlegt werden, die in Abbildung 3.17 grafisch dargestellt und nachfolgend kurz erläutert werden.

Signalanalyse: Gesprochene Sprache wird in der Form von Schallwellen durch die Luft übertragen. Dieses natürliche, analoge Sprachsignal wird mit Hilfe eines Mikrofons empfangen, in elektrische Impulse umgewandelt und anschließend für den Computer digitalisiert, d.h. in eine Zahlenfolge verwandelt. Die Signalanalysekomponente des Erkenners generiert eine parametrische Repräsentation des Sprachsignals, die einerseits so kompakt wie möglich ist, aber andererseits noch die zur Erkennung notwendige Information enthält. Ganz analog zur Verarbeitung durch das periphere auditorische System des Menschen lassen sich bei der Generierung der parametrischen Repräsentation zwei aufeinander aufbauende Stufen unterscheiden.

Die **Vorverarbeitung** transformiert die im natürlichen Sprachsignal zeitlich kodierte Information in eine Darstellung, die die spektralen Eigenschaften der Lautsprache zu einem bestimmten Zeitpunkt und deren Veränderungen im Verlauf der Äußerung repräsentiert. Die spektrale Analyse liefert Informationen über die Energie in den einzelnen Frequenzbereichen. Sprachlaute unterscheiden sich untereinander durch ihre jeweils charakteristische Energieverteilung im Spektrum. Die Vorverarbeitung produziert also eine Repräsentation des Sprachsignals, wie sie uns aus der akustisch-phonetischen Analyse als Spektrogramm bereits bekannt ist (vgl. Unterkapitel 3.1). In der menschlichen Sprachwahrnehmung erzeugt das Innenohr ganz analog ein auditorisches Spektrogramm.

Im zweiten Schritt gewinnt die **Merkmalsextraktion** die zur Erkennung der Äußerung geeigneten Merkmale. Man kann sich diese Merkmale als höherwertige, robuste Merkmale vorstellen, die sich aus den akustischen Basisparametern ableiten lassen. Zugleich wird in diesem Schritt eine erhebliche Reduktion der Datenmenge und ihrer Dimensionalität vorgenommen. Auch die zumeist statistisch motivierte Merkmalsextraktion hat nach den vorherrschenden Sprachperzeptionsmodellen ihre Parallelen in der menschlichen Sprachwahrnehmung.

Im Erkenner werden die beiden Schritte, also Vorverarbeitung und Merkmalsextraktion, in regelmäßigen, sehr kurzen Abständen wiederholt, typischerweise

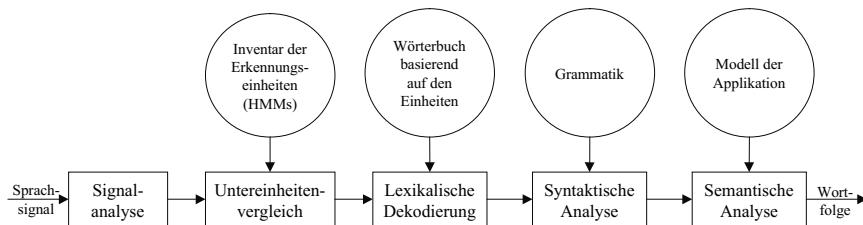


Abbildung 3.17: Blockdiagramm eines Spracherkenners nach Rabiner (1989)

alle 10 ms. Eine sprachliche Äußerung, die genau eine Sekunde lang ist, wird also durch eine Folge von 100 Merkmalsvektoren repräsentiert. Die Signalanalyse soll nicht näher vorgestellt werden, da bei diesem Prozess weniger computerlinguistische Methoden als Techniken und Algorithmen der digitalen Sprachsignalverarbeitung eingesetzt werden. Wichtig für das weitere Verständnis ist hier jedenfalls, dass die Signalanalyse das eingehende Sprachsignal in eine Folge von Merkmalsvektoren transformiert.

Untereinheitenvergleich: Dieser Vorgang, der in der Fachliteratur auch als **unit matching** bezeichnet wird, spielt im Spracherkennner eine ganz zentrale Rolle, denn hier wird eine Folge von Merkmalsvektoren in eine Folge von Lauten, Silben oder Wörtern umgesetzt. Es handelt sich um einen erheblichen Abstraktionsschritt, denn die physikalische Repräsentation der Äußerung wird nun in eine linguistische Repräsentation transformiert. Dabei variiert die Größe oder Länge von linguistischen Einheiten (*units*) von System zu System. Aktuelle Spracherkennner verwenden z. B. Phoneme, Diphone, Halbsilben, Silben oder auch ganze Wörter als atomare linguistischen Bausteine, die im Sprachsignal identifiziert werden sollen. Während des Vergleichsvorgangs (*matching*) wird mit Hilfe der dynamischen Programmierung zu jedem Zeitpunkt im Signal eine Bewertung für jede in Frage kommende Einheit geliefert. Einschränkungen möglicher Einheiten werden oft aufgrund lexikalischer und syntaktischer Einschränkungen vorgenommen.

Lexikalische Dekodierung: Lexikalische Einschränkungen erzwingen eine Beschränkung der möglichen Suchpfade auf solche, die Einheitensequenzen entsprechen, die ausschließlich aus im Lexikon aufgelisteten Einheiten bestehen. Lexikoneinträge sind bei diesem Vorgehen so kodiert, dass sie zu jedem Worteintrag auch die Sequenz von Untereinheiten spezifizieren, aus denen sich das Wort zusammensetzen lässt und die als linguistische Bausteine im Erkennern vorgesehen sind. In silbenbasierten Erkennern ist also die Silbenfolge und bei lausbasierten Erkennern zusätzlich die Lautfolge spezifiziert, während im Fall von Ganzworteinheiten der Schritt der lexikalischen Dekodierung komplett entfällt, wodurch sich die Struktur des Erkenners vereinfacht. Tabelle 3.2 zeigt die maximale Struktur eines solchen Lexikoneintrags (vgl. auch Abbildung 3.19 auf Seite 221).

Wort	/Einheiten
Bonn	/bOn/
Dortmund	/dO6t.mUnt/
fahren	/fa:.R@n/
ich	/IC/
nach	/na:X/

Tabelle 3.2: Lexikoneintrag mit Spezifizierung der Lautfolge und der Silbenfolge (Silbentrennungssymbol „.’“).

Syntaktische Analyse: Die durch die lexikalische Dekodierung bereits reduzierte Auswahl von Einheitensequenzen kann durch syntaktische Einschränkungen weiter reduziert werden. Nach diesem Schritt werden ausschließlich Modellpfade weiterverfolgt, die Wörter aus dem Lexikon ergeben *und* bei denen mehrere dieser Wörter in einer geeigneten Reihenfolge sind, wobei geeignet im Sinne der spezifizierten Grammatik zu verstehen ist (vgl. Abschnitt 3.2.1).

Semantische und pragmatische Analyse: Im Prinzip lassen sich analog zur lexikalischen und syntaktischen Dekodierung weitere Einschränkungen der Suche durch semantische und pragmatische Randbedingungen aufstellen. Dieser Schritt, häufig als **Sprachverständen** bezeichnet, wird allerdings üblicherweise im **Dialogmanager** (vgl. Unterkapitel 5.5) ausgeführt. Die semantische Analyse erfolgt im Allgemeinen erst nach Beendigung des Erkennungslaufs über die zuvor beschriebenen Schritte für eine ganze Äußerung und somit ohne Rückkopplung zu den vorausgehenden Schritten. Semantische Restriktionen werden also in der Praxis nicht in den eigentlichen Suchprozess des Einheitenvergleichs integriert.

Modellierung der gesprochenen Sprache

Abbildung 3.17 illustriert ein generisches System zur maschinellen Spracherkennung. Das vorherrschende Paradigma der automatischen Spracherkennung beruht auf **Hidden-Markov Modellen** (HMM). Dieses Unterkapitel wird sich daher bei der Darstellung der Modellierung der gesprochenen Sprache aus Platzgründen auf HMM-basierten Spracherkennern konzentrieren, da dieser Typ die weitaus größte Verbreitung findet. Andere Ansätze zur automatischen Spracherkennung nutzen z. B. Neuronale Netze oder bewegen sich als Mischform in beiden Welten (hybride Systeme).

HMMs werden im Unterkapitel 2.4 eingeführt, so dass die entsprechenden methodischen Grundlagen hier vorausgesetzt werden. Zur Erinnerung: HMMs erster Ordnung werden (a) durch eine Menge von Zuständen, (b) eine Tabelle von Übergangswahrscheinlichkeiten (Transitionen) zwischen diesen Zuständen und (c) den in jedem Übergang beobachtbaren Ausgaben (Emissionen) charakterisiert, wobei ein solcher Markov-Prozess durch eine Zufallsverteilung über die Anfangszustände in Bewegung gebracht wird. Im konkreten Kontext der automa-

tischen Spracherkennung sollen zwei zentrale Fragestellungen bezüglich HMMs aufgegriffen werden.

- Wie sollen die HMM-Parameter, also die Transitionen und die Emissionen, eingestellt werden, so dass sie eine gegebene Sequenz von Beobachtungen am besten (wahrscheinlichsten) modellieren?
- Gegeben sind nur die Beobachtungen. Welche Zustandsfolge liegt ihnen dann am wahrscheinlichsten zugrunde?

Hinter dem ersten Punkt verbirgt sich das **Trainingsproblem** eines Spracherkenners und hinter dem zweiten das **Erkennungsproblem**. Aus der Tatsache, dass man die Zustände nicht kennt, leitet sich das Adjektiv *hidden* in HMM ab. Algorithmen zur Lösung dieser Probleme sind in Unterkapitel 2.4 ausführlich beschrieben.

Unter dem abstrakten Begriff der Zustände stelle man sich zum Verständnis der Spracherkennung am besten Laute (genauer: Teile von Lauten) vor, die anhand der Beobachtungen (des Signals) aufgedeckt werden müssen. Hat man also die wahrscheinlichste Lautfolge (Zustandsfolge), so lassen sich über das Lexikon auch die wahrscheinlichsten Wörter finden.

Wie oben beschrieben ist das Ergebnis der Signalanalyse eine Folge von T Merkmalsvektoren, den sogenannten Beobachtungen $O = (o_1, \dots, o_T)$. Die Aufgabe der Spracherkennung besteht in der Dekodierung der tatsächlich geäußerten Wortfolge $W = (w_1, \dots, w_m)$ bei gegebener Beobachtung O . Dazu greift man auf diejenige Wortfolge zurück, die mit Hilfe der Bayes'schen Formel (vgl. Unterkapitel 2.4) die größte (a posteriori) Wahrscheinlichkeit

$$P(W|O) = \frac{P(O|W)P(W)}{P(O)} \rightarrow \max_W \quad (3.1)$$

erhält. Dabei kann der Nenner vernachlässigt werden, da er unabhängig von W ist. Man muss also lediglich das Produkt $P(O|W) \cdot P(W)$ maximieren. Betrachtet man der Einfachheit halber einmal nur Wortfolgen der Länge eins (man spricht dann von Einzelworterkennung), so besteht das Problem der Spracherkennung in der Berechnung des Produktes für alle Wörter des Erkennungslexikons und anschließender Ausgabe des wahrscheinlichsten Wortes.

Um diese Berechnung durchführen zu können, konzentrieren sich die folgenden Abschnitte auf die beiden Faktoren: Der erste Faktor $P(O|W)$ modelliert den akustischen Teil, wohingegen $P(W)$ als a priori Wahrscheinlichkeit der Wortfolge die Syntax einer Äußerung modelliert.

Akustische Modellierung: Zur Bestimmung des Terms $P(O|W)$ werden HMMs verwendet, die einen zweistufigen stochastischen Prozess modellieren (Abbildung 3.18).

Die erste Stufe nimmt eine Markovkette erster Ordnung ein, die durch eine Zustandsfolge $S = (S_1, \dots, S_T)$ gegeben ist. Aufgrund der Markovbedingung, dass nur der unmittelbar vorangegangene Zustand S_{t-1} Einfluss auf den aktuellen Zustand S_t hat und dass die absolute Zeit t keine Rolle für eine Transition

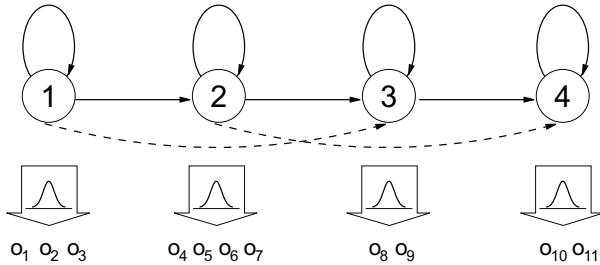


Abbildung 3.18: Links-Rechts HMM mit 4 Zuständen ($T = 11$)

spielt (stationärer Prozess), wird das stochastische Verhalten einer Markovkette nur durch die initiale Zustandsverteilung π und die Tabelle A der **Übergangswahrscheinlichkeiten** bestimmt.

In der Sprachverarbeitung werden hauptsächlich HMMs benutzt, bei denen Transitionen zu Zuständen mit kleinerem Index nicht möglich sind (Abbildung 3.18). Für diese sogenannten **Links-Rechts Modelle** ist die Transitionsmatrix A also eine obere Dreiecksmatrix.

Eine Markovkette wird aber erst dann zum eigentlichen HMM, wenn den Zuständen noch jeweils eine sogenannte **Ausgaben-** oder **Emissionsverteilung** anhaftet, nach deren Gesetz die Beobachtung o_t im Zustand S_t erzeugt wird, ohne von der Vergangenheit abhängig zu sein. Laut Beschaffenheit dieser Verteilungen unterscheidet man auch zwischen diskreten und kontinuierlichen HMMs.

Aber nun wieder zurück zum ursprünglichen Ziel, $P(O|W)$ zu bestimmen. Dafür benötigt man zunächst ein (Hidden Markov) Modell der Wortfolge W . Dieses wird nach dem Baukastenprinzip aus kleineren HMMs (den erwähnten Untereinheiten, siehe Abschnitt 3.2.1) zusammengesetzt (Abbildung 3.19). Eine Wortfolge fügt sich also ausschließlich aus Wörtern des Systemlexikons zusammen, die ihrerseits aus den Untereinheiten aufgebaut werden. Um diese Verkettung der HMMs zu ermöglichen, denkt man sich pro Einheit noch eine Transition zu einem absorbierenden (vierten) Zustand, der dann mit dem Anfangszustand des nächsten Modells identifiziert wird. Auf diese Weise erhält man schließlich ein (virtuelles) HMM für einen ganzen Satz W .

Nach einem einfachen mathematischen Gesetz lässt sich, wie in Gleichung (3.2) dargestellt, $P(O|W)$ durch Summation über alle denkbaren Zustandsfolgen S der Länge T bestimmen. Hierdurch kommen die vorausgesetzten Annahmen ins Spiel, dass eine zufällige Zustandsfolge S der Markoveigenschaft genüge und die Emission eines Merkmalsvektors o_t nur vom aktuellen Zustand S_t abhängt; dies ist in Gleichung (3.3) dargestellt.

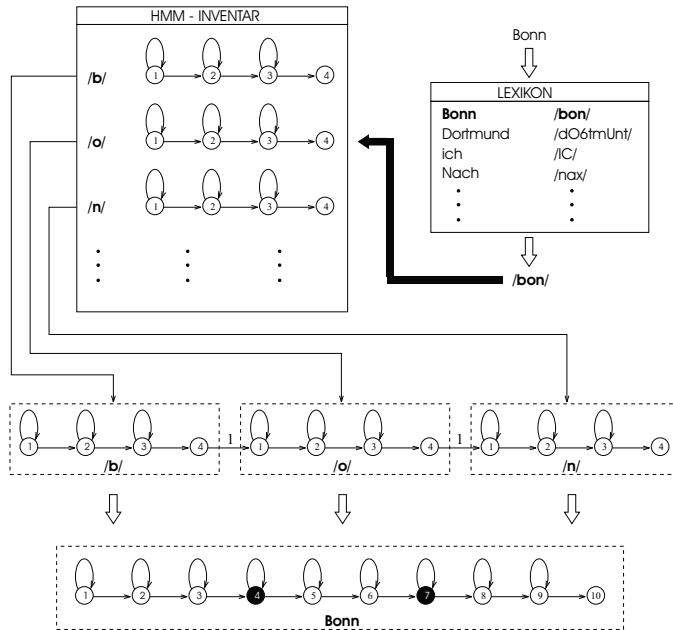


Abbildung 3.19: Anwendung des Baukastenprinzips zur Modellierung eines Wortes.

$$P(O|W) = \sum_{s \in S} P(S = s) \cdot P(O|S = s) \quad (3.2)$$

$$= \sum_{s \in S} \pi_{s_1} \left(\prod_{t=2}^T a_{s_{t-1}s_t} \right) \left(\prod_{t=1}^T f_{s_t}(o_t) \right) \quad (3.3)$$

Da in aktuellen Systemen nicht selten 5000 verschiedene Zustände und mehr benutzt werden, erhält man schon bei einer Äußerung von nur einer Sekunde (100 Beobachtungen) 5000^{100} verschiedene Zustandsfolgen (= Summanden in der Formel), was einer Eins mit 370 Nullen entspricht (!), wenn jeder Zustand auf jeden folgen kann. Deshalb verbietet die Komplexität dieser Gleichung ihren direkten Einsatz. Aber aufgrund der Markoveigenschaft wird eine sehr effiziente rekursive Berechnung dieser Formel und damit von $P(O|W)$ ermöglicht (siehe Unterkapitel 2.4).

Sprachmodellierung: Durch die Sprachmodellierung wird einer Wortfolge W eine (a priori) Wahrscheinlichkeit $P(W)$ zugeordnet. Das **Sprachmodell** (*language model*) stellt damit neben dem akustischen Modell eine weitere Wissensquelle dar, die zur Steigerung der Erkennungsleistung genutzt wird. Zur Modellierung dieser Wahrscheinlichkeit verwendet man typischerweise **statistische**

Sprachmodelle, die auf sehr großen Textkorpora (anwendungsspezifisch) trainiert werden. Um die Schätzung der Verteilung über die (unendlich vielen) Wortfolgen auf die eines endlichen Satzes von Parametern zu reduzieren, wird die folgende Näherung gemacht

$$P(W) = P(w_1) \prod_{i=2}^m P(w_i|w_1, \dots, w_{i-1}) \quad (3.4)$$

$$\approx P(w_1) \prod_{i=2}^{n-1} P(w_i|w_1, \dots, w_{i-1}) \prod_{i=n}^m P(w_i|h_i), \quad (3.5)$$

wobei $h_i := (w_{i-n+1}, \dots, w_{i-1})$ für eine verkürzte „Geschichte“ des Wortes w_i steht. Sprachmodelle, die eine derartige Beschränkung der Länge der Geschichte auf $n - 1$ Wörter voraussetzen, werden **n-gramm Modelle** genannt. N-gramm Modelle und ihre gängigsten Varianten, nämlich Unigramme ($n = 1$), Bigramme ($n = 2$) und Trigramme ($n = 3$), werden im Unterkapitel 2.4 eingeführt.

Als Schätzung der bedingten Wahrscheinlichkeit $P(w|h)$ wählt man die relative Häufigkeit der Wortfolge (h, w) unter allen Wortfolgen h mit beliebigem Nachfolger,

$$\hat{P}(w|h) = \frac{c(h, w)}{c(h)} \quad (c(\cdot) = \text{abs. Häufigkeit}), \quad (3.6)$$

wodurch die Bewertung des Trainingstextes $P(W_{\text{TRN}})$ maximal wird.

Bei der Benutzung dieser Parameter ergibt sich für Sätze, die im Training ungesehene n-gramme enthalten, eine Wahrscheinlichkeit von Null. Dies führt zu erheblichen Problemen bei der Erkennung. Um die Schätzung robuster zu machen, wird daher ein Teil der „Wahrscheinlichkeitsmasse“ von den gesehenen auf die ungesehenen Ereignisse umverteilt (*discounting*). Zumeist wird eine Rückfall-Strategie (*back-off*) angewandt, die die Wahrscheinlichkeit eines ungesesehenen n-gramms anhand der Häufigkeit des $(n - 1)$ -gramms schätzt.

Perspektiven

Trotz großer technologischer Fortschritte in den letzten Jahren wird es aber auch in absehbarer Zukunft keine unbeschränkt einsetzbaren Systeme der Spracherkennung geben, sondern weiterhin vorwiegend Systeme für spezielle Anwendungen, bei denen man Einschränkungen z. B. im Hinblick auf Benutzer, Domäne, Vokabular, Syntax oder auch Umgebung macht. So findet man heute häufig syntaktisch einfache Dialoge zur Bedienung von Geräten (Mobiltelefon im Auto) oder zum Informationaustausch im Dienstleistungssektor (Tele-Banking). Bei Diktionsystemen, die einen großen Wortschatz mit nahezu natürlichen Eingaben zulassen, müssen dafür Abstriche bei Umgebungsgeräuschen oder hinsichtlich des Sprachflusses (Pause zwischen Wörtern) bzw. der Benutzeranzahl (lange Gewöhnungsphase vor Erstnutzung) gemacht werden, um die Qualität der Erkennung zu erhöhen.

Hinter jeder dieser Einschränkungen verbergen sich offene Probleme der Grundlagenforschung in der Sprachtechnologie. So ist es beispielsweise wichtig, das Gesprochene von den Geräuschen der Umgebung zu trennen, bevor es erkannt werden kann. Eine andere Herausforderung stellt die Abhängigkeit von der Sprechervielfalt dar. Die alltägliche Erfahrung zeigt, dass manche Stimmen zu hoch, zu schnell oder zu undeutlich und damit schwerer zu verstehen sind als andere. Menschen können sich nach recht kurzer Zeit an die stimmlichen Eigenchaften des jeweiligen Sprechers anpassen, und so sollte auch ein Spracherkenner entsprechend flexibel und adaptiv sein.

Ebenfalls noch weit entfernt von menschlicher Sprachleistung ist das Vermögen, unbekannte Wörter zu analysieren und zu interpretieren. Dies liegt v.a. am Erkennungskriterium selbst, bei dem das beste Wort im Lexikon (egal wie schlecht es passt) als Hypothese ausgegeben wird, und ein bestes gibt es natürlich immer. Lösungen für diese (und mehr) Probleme müssen also erst gefunden werden, um das gesamte Potenzial dieser Technologie in eine Vielfalt von akzeptablen Produkten einbringen zu können.

3.2.2 Sprachsynthese

Sprachsynthese wird überall dort eingesetzt, wo die Ausgabe von Information nur oder vorzugsweise auf akustischem sprachlichem Weg erfolgen kann. Derzeit wird die Sprachsynthese zunehmend in Auskunftsystmen (siehe Unterkapitel 5.6 und 5.5) eingesetzt. Hier sind die Anwendungsmöglichkeiten vielfältig: Navigationssysteme, Verkehrsmeldungen, Reiseauskünfte, Kinoprogramme, Börsenkurse, Webseiten, Email, und andere mehr. Insbesondere in der Mobiltelefon-Kommunikation, aber auch etwa im Auto, wo der Gesetzgeber oder die Vernunft des Fahrers eine Informationsausgabe auf einen Bildschirm untersagt, muss auf akustische Sprachausgabe zurückgegriffen werden. Klassische Anwendungen sind weiterhin der Computerarbeitsplatz für Blinde und Sehbehinderte oder die künstliche Stimme für Sprechbehinderte (siehe Unterkapitel 5.4).

Die übergreifende wissenschaftliche Theorie hinter der Sprachsynthese kann als ein funktionales Modell der menschlichen Sprachproduktion gelten. Unter diesem Aspekt kann die Ambition der Sprachsynthese als die Modellierung der wohl komplexesten kognitiven Fähigkeit des Menschen charakterisiert werden. So wenig perfekt dieses funktionale Modell ist, so wenig ist das Problem der optimalen Sprachsynthesequalität bislang gelöst.

Struktur eines TTS-Systems

Sprachsynthese (text-to-speech, TTS) kann als ein zweistufiger Prozess beschrieben werden. In einem ersten Schritt wird der Eingabetext linguistisch analysiert, und in einem zweiten Schritt wird die aus der Analyse resultierende linguistische Repräsentation in ein synthetisches Sprachsignal umgesetzt. Ein Sprachsynthesesystem (TTS-System) ist ein komplexes System, dessen Leistungsfähigkeit durch die Qualität der einzelnen Komponenten bestimmt wird, aus denen es besteht. Abbildung 3.20 zeigt die Hauptkomponenten, die in allen TTS-Systemen anzu-

treffen sind. Obwohl es durchaus Unterschiede in der Architektur verschiedener Systeme gibt, können sie im allgemeinen auf die in der Abbildung gewählte „Pipeline“-Architektur zurückgeführt werden.

Infolge der nicht umkehrbaren Verarbeitungsrichtung lässt sich, anders als in einem Spracherkennungssystem, eine lücken- oder fehlerhafte Verarbeitung durch eine TTS-Komponente nicht in einer späteren Komponente ergänzen oder korrigieren. Fehlanalysen pflanzen sich also durch das System fort und lösen oft Folgefehler aus. Da die in den Komponenten zum Einsatz kommenden linguistischen, phonetischen und akustischen Modelle nicht perfekt sind, führt die Verarbeitung im System zu einer zunehmenden Distanz der Qualität des synthetischen Sprachsignals zur Qualität der natürlichen Sprache. Auf die verschiedenen Verfahren zur Generierung des künstlichen Sprachsignals, d.h. insbesondere auf die zugrundeliegenden Modelle der Sprachproduktion und Artikulation, kann im Rahmen des vorliegenden Buches nicht eingegangen werden.

Die folgenden Abschnitten konzentrieren sich auf diejenigen Komponenten und Verarbeitungsschritte in einem TTS-System, die aus dem schriftlichen Eingabetext eine linguistische Repräsentation herleiten und für die akustische Synthese bereitstellen.



Abbildung 3.20: Hauptkomponenten von Sprachsynthesesystemen.

Computerlinguistische TTS-Komponenten

Die in Abbildung 3.20 dargestellten Verarbeitungsblöcke bestehen üblicherweise aus mehreren, im Fall der linguistischen Textanalyse sogar oft aus einer Vielzahl von Modulen, von denen jedes einem wohldefinierten Teilproblem entspricht.

Zur Illustration der Komplexität der linguistischen Textanalyse soll der folgende Satz dienen:

Bei der Wahl am 12.3.1998 gewann Tony Blair ca. 52% der Wählerstimmen.

Welches Wissen muss ein Sprecher des Deutschen mitbringen, um ihn korrekt vorzulesen? Zunächst einmal muss er die Aussprache regulärer Wörter aus ihrer schriftlichen Form ableiten können. Dies setzt unter anderem die Kenntnis der internen Struktur von Wörtern voraus. So muss *Wählerstimmen* in die Komponenten *Wähler* und *Stimmen* zerlegt werden, um die Buchstabenfolge *st* korrekt als [st] auszusprechen, im Unterschied etwa zu dem Wort *Erstimpfung*. Weiterhin sollte *Tony Blair* als ausländischer Name erkannt und idealerweise englisch ausgesprochen werden. Die Abkürzungen *ca.* und *%* sowie die Zahl *52* und das Datum *12.3.1998* schließlich müssen in reguläre Wortformen umgewandelt werden. Eine besondere Schwierigkeit ist, dass der orthographische Punkt beim ersten und zweiten Auftreten im Beispielsatz als Teil des Datums erkannt werden muss, im dritten Fall eine Abkürzung und im vierten Fall das Satzende

markiert. Tatsächlich stellen sich dem Sprecher noch weitere Probleme, etwa die richtige Betonung von Wörtern und Silben sowie die Auswahl einer geeigneten Sprachmelodie oder Intonation.

Ein Problem der Textanalyse, das bislang nicht angesprochen wurde, ist die Zerlegung des Eingabetextes in Wörter. Dies ist selbst für das Deutsche, das Wörter in der Regel durch Leerzeichen voneinander trennt, keine triviale Aufgabe. So müssen die numerischen Ausdrücke 52% und das Datum in mehrere separate Wörter expandiert werden. Wesentlich schwieriger verhält es sich in Sprachen wie dem Chinesischen oder Japanischen, in denen Wörter keine direkte orthographische Entsprechung haben und Wortgrenzen demnach auch nicht durch Leerzeichen markiert werden. Dennoch existieren in diesen Sprachen Wörter als lexikalische Einheiten, so dass die linguistische Analyse auch hier eine Wortsegmentierung vornehmen muss.

Die Expandierung von Symbolen wie % ist in einigen Sprachen ebenfalls komplexer als im Deutschen, wo es ausnahmslos als *Prozent* gesprochen wird. So ist etwa im Russischen eine Analyse des Satzzusammenhangs erforderlich, um die korrekte grammatische Form von *Prozent* zu ermitteln, da abhängig vom Kontext eine Vielzahl von Varianten möglich ist. Die Beispiele des russischen %, aber auch des Datums 12.3.1998, zeigen, dass eine simple Vorverarbeitung oder Textnormalisierung, wie sie in manchen Systemen anzutreffen ist, unzureichend ist. Um Symbole, Abkürzungen und komplexe numerische Ausdrücke in die korrekten Wortformen zu expandieren, ist eine gründliche Analyse des Kontextes unumgänglich. Tag und Monat des Datums müssen als Ordinalzahlen ausgedrückt und in die mit der vorangehenden Präposition übereinstimmende grammatische Form (Dativ Singular) gesetzt werden: *am zwölften dritten*. Die Jahreszahl bedarf ebenfalls einer besonderen Behandlung: *neunzehnhundert achtundneunzig*, nicht *eintausend neinhundert achtundneunzig*.

Im Folgenden werden die aus computerlinguistischer Sicht wichtigsten Aspekte der Textanalyse näher betrachtet, und zwar die lexikalische und morphologische Analyse, wortübergreifende Sprachmodelle und die Ausspracheregeln. Schließlich wird ein einheitlicher Formalismus für die linguistische Repräsentation und für deren Implementierung im TTS-System vorgestellt.

Lexikalische Analyse: Die weitaus meisten TTS-Systeme verfügen über ein Lexikon, das zu jedem Eintrag Informationen über die Wortart und andere grammatische Kategorien und außerdem die Aussprache in Form einer phonetischen Transkription enthält. In vielen Fällen handelt es sich um ein Vollformenwörterbuch, d.h. es ist nicht nur jeweils die Grundform des Wortes aufgeführt, sondern auch die unterschiedlichen Wortformen.

Eleganter ist die Methode, für Wörter mit komplexer Flexionsmorphologie, im Deutschen also Nomina, Adjektive und Verben, Flexionsparadigmata oder Fortsetzungsklassen zu definieren und an jedem Wortstamm zu markieren, welches Paradigma zutrifft (siehe Unterkapitel 3.3). Die Expansion eines Wortstammes in alle legalen flektierten Wortformen kann dann automatisch und vollständig erfolgen.

Nichtflektierte und nicht abgeleitete Wortarten werden in einfachen Teillexika abgelegt. Spezielle Wortlisten lassen sich außerdem für Eigennamen, geographi-

sche Namen und ähnliche Kategorien sowie für die Expansion von Abkürzungen erstellen. Weiterhin verfügen TTS-Systeme oft über spezielle linguistische Modelle für die Behandlung von numerischen Ausdrücken. Als letzter Ausweg steht immer das Buchstabieren von Graphemsequenzen offen, die nicht weiter analysiert werden können.

Derivation und Komposition: Eine Besonderheit des Deutschen und einiger anderer Sprachen sind zusammengesetzte Wörter, also Komposita, wie *Wählerstimmen* in dem Beispielsatz. Die Bildung von Komposita ist ausgesprochen produktiv: Sprecher des Deutschen können jederzeit neue Zusammensetzungen bilden. Dies hat zur Konsequenz, dass in nahezu jedem Text Wörter auftreten, die in keinem noch so umfangreichen Lexikon aufgelistet sind. Die linguistische Analyse muss daher in der Lage sein, Komposita und auch Derivationen in ihre Bestandteile zu zerlegen. Als Grundlage hierzu kann ein Modell der morphologischen Struktur von Wörtern und der Kombinierbarkeit von Morphemen dienen.

Ein solches Wortmodell könnte beispielsweise für *Wählerstimmen* folgende mehr oder weniger plausible Analysen liefern:²

wähl [Vb-Stamm] + *erst* [Adj-Stamm] + *imme* [Nom-Stamm] + *n* [pl]
wähler [Vb-Stamm] + *st* [2per-sg] + *imme* [Nom-Stamm] + *n* [pl]
wähler [Nom-Stamm] + *stimme* [Nom-Stamm] + *n* [pl]

Die korrekte Lesart muss anhand von Wahrscheinlichkeiten, Kosten oder Auftretenshäufigkeiten in Korpora ermittelt werden, möglicherweise unterstützt durch eine Analyse des syntaktischen Kontextes.

Sprachmodelle und prosodische Analyse: Die lexikalische und morphologische Analyse liefert häufig alternative Lesarten, die erst durch lokale grammatische Sprachmodelle, die über die Wortgrenze hinaus den syntaktischen Kontext miteinbeziehen, disambiguiert werden können. Die häufigste Aufgabe für solche lokalen Grammatiken ist die Sicherstellung der syntaktischen Kongruenz (Agreement) zwischen zusammengehörigen Wörtern.

Zu den wortübergreifenden Modellen gehören auch die syntaktische und prosodische Phrasierung und die Bestimmung des Satzmodus. Viele TTS-Systeme verfügen nur über Heuristiken, um diese Aufgaben zu bewältigen. Unter den TTS-Systemen für das Deutsche zeichnen sich das SVOX-System der ETH Zürich und das IMS-Festival-System der Universität Stuttgart durch den Einsatz eines syntaktischen Parsers (siehe Unterkapitel 3.5) und Part-of-Speech-Taggers (siehe Unterkapitel 3.3) aus; die von diesen Modulen gelieferte Information bildet die Basis für die Festlegung von Phrasengrenzen und Akzenten.

Phonologische Analyse und Aussprache: In TTS-Systemen, die ein Vollformenwörterbuch verwenden, ist die Aussprache eines Wortes durch seine Transkription im Lexikon gegeben. Im Eingabetext auftretende Wörter, die nicht im Lexikon enthalten sind, werden durch Ausspracheregeln transkribiert. Solche Systeme zeichnen sich häufig durch eine Vielzahl von Ausnahmeregeln aus.

²*imme* [Nom-Stamm] – *Imme*: landschaftlich bzw. fachsprachlich für *Biene*.

Eleganter ist hier ein Design der linguistischen Analysekomponente, die jedem Wort gerade so viel morphologische Annotation mitgibt, dass generische Ausspracheregeln eine zuverlässige Transkription liefern können. Bei im TTS-Lexikon vorhandenen Wörtern ist diese Information bereits gegeben, und für „unbekannte“ Wörter liefert die Komposita- und Derivationsanalyse eine Granularität der Annotation, die der der bekannten Wörter äquivalent ist. Auf diese Weise werden Ausnahmeregeln weitestgehend überflüssig. Zur Aussprache eines Wortes gehört selbstverständlich nicht nur die Phonemfolge, sondern auch die Markierung der Silbenbetonung.

Im Deutschen hängt die Aussprache vorrangig von der morphologischen Struktur eines Wortes und erst danach von der Silbenstruktur ab. So wird in *Tonart* die Standard-Syllabifizierung (/to:-nart/) durch die Morphemgrenze außer Kraft gesetzt (/to:n+art/). Eine Syllabifizierung der ermittelten Phonemfolge muss dennoch vorgenommen werden, da die akustischen prosodischen Komponenten des TTS-Systems, also die Lautdauer- und Intonationsmodule, die Silbenstruktur als Eingabeinformation benötigen.

Ein einheitlicher Formalismus

Die Vielfalt der Probleme, die sich in den verschiedenen Sprachen im Zusammenhang mit der linguistischen Analyse stellen, scheint zunächst gegen eine generelle Lösung zu sprechen. Es ist jedoch möglich, die Problematik in einer abstrakteren Weise zu betrachten als in den angeführten Beispielen geschehen. Jedes Teilproblem kann als Transformation von einer Kette von Symbolen (konkret: Schriftsymbolen) in eine andere Kette von Symbolen (konkret: linguistische Analyse) beschrieben werden. So wird etwa die Buchstabenfolge *Wählerstimmen* in eine linguistische Repräsentation überführt, die nun auch Informationen über die Struktur des Wortes enthält: *wähler* [Nom-Stamm] + *stimme* [Nom-Stamm] + *n* [pl]. Auf vergleichbare Weise wird eine Folge von Schriftzeichen in einem chinesischen Satz in eine Darstellung überführt, die unter anderem Informationen über Wortgrenzen enthält.

Analog lässt sich auch der nächste Schritt im Rahmen der Textanalyse beschreiben, nämlich die Bestimmung der Aussprache von Wörtern. Dabei nutzen die Ausspracheregeln für eine bestimmte Sprache die aus der linguistischen Analyse gewonnenen Informationen und konvertieren die linguistische Repräsentation in eine Folge von Lautsymbolen. So ermöglicht erst die Information über die wortinterne Grenze vor *st* in *Wählerstimmen* die Bestimmung der korrekten Aussprache des Wortes.

Ein flexibles und zugleich mathematisch elegantes Modell, das die soeben skizzierte Konvertierung von Symbolketten erlaubt, beruht auf der Technologie der *Finite State Transducers* (FST, siehe Unterkapitel 2.2). Ein FST ist ein endlicher Automat, der eine Eingabe-Zeichenkette erkennt und daraus eine Ausgabe-Zeichenkette erzeugt. Ein solcher Automat enthält eine endliche Anzahl von Zuständen; für jeden dieser Zustände bestimmt eine Tabelle, zu welchen anderen Zuständen Übergänge möglich sind, und zwar in Abhängigkeit davon, welche Eingabesymbole gerade verarbeitet werden. Die Tabelle bestimmt auch, welche

Symbole daraufhin ausgegeben werden. Transducer, die eine komplexe Aufgabe wie die linguistische Analyse in einem Sprachsynthesesystem übernehmen sollen, verfügen zumeist über eine sehr große Anzahl (typischerweise einige hunderttausend) von Zuständen.

Die linguistische Analysekomponente im multilingualen TTS-System der Bell Labs (Sproat 1998) ist vollständig nach diesen Prinzipien konstruiert und verarbeitet viele der Phänomene und Probleme, die in den einzelnen Sprachen im Rahmen der Textanalyse auftreten, einschließlich der verschiedenen Schriftsysteme (lateinisch, kyrillisch, chinesisch, japanisch). Die hier skizzierte einheitliche Software-Architektur für multilinguale Sprachsynthese ermöglicht eine vergleichsweise einfache Erweiterung auf neue Sprachen, und ihre modulare Struktur erleichtert die Integration verbesserter Komponenten für bereits existierende Systeme. Die linguistische Analysekomponente der deutschen Version dieses Systems ist in (Möbius 1999) detailliert beschrieben worden. Endliche Automaten werden auch im TTS-System SVOX der ETH Zürich eingesetzt.

Perspektiven

In diesem Abschnitt wurde bislang von der meistverbreiteten und zugleich ambitioniertesten Zielrichtung der Sprachsynthese ausgegangen, der Sprachsynthese für unbeschränkte Texteingabe und für unbeschränkte Anwendungsdomänen – also dem klassischen Vorleseautomaten. Synthetische Sprache kann jedoch aus recht unterschiedlichen Eingabeinformationen erzeugt werden. Die Eingabe kann maschinenlesbarer Text sein oder ein strukturiertes Dokument oder mit speziellen Steuerzeichen annotierter Text (es gibt eigens für die Sprachausgabe entwickelte **Markup Languages**) oder auch semantische Konzepte.

Unbeschränkte textbasierte Sprachsynthese stellt hier ein Extrem in einem Quasi-Kontinuum von Szenarien dar. Am anderen Ende des Kontinuums stehen Sprachausgabesysteme, die ein kleines Inventar abgespeicherter Sprachbausteine (z. B. Systemprompts oder wiederkehrende Phrasen) neu kombinieren und wiedergeben. Solche auf *canned speech* oder *sliced speech* basierende Systeme sind nur in strikt definierten und geschlossenen Anwendungsdomänen einsetzbar. Sie erfordern keine ernsthafte computerlinguistische Verarbeitung und sollen daher hier auch nicht weiter diskutiert werden.

TTS-Systeme müssen eine sehr große, ja unbegrenzte Anzahl möglicher Eingabesätze verarbeiten können. Sie benötigen hierzu linguistische und prosodische Modelle sowie ein akustisches Inventar, das die synthetische Sprachausgabe für eine solche Texteingabe in einer Qualität ermöglicht, die für die Benutzer des Systems akzeptabel ist. TTS-Systeme bieten so die größtmögliche Flexibilität, für die jedoch ein hoher Preis in Form reduzierter Natürlichkeit der Sprachausgabe zu zahlen ist.

Hingegen ermöglicht die konzeptbasierte Sprachsynthese (concept-to-speech, CTS), üblicherweise integriert in ein Dialog- oder Übersetzungssystem (siehe Unterkapitel 5.5, 5.6 und 5.7), die Generierung synthetischer Sprache auf der Grundlage pragmatischen, semantischen und Diskurswissens. Der Vorteil gegenüber einem TTS-System ist, dass die sprachgenerierende Komponente des CTS-

Systems „weiß“, was sie sagen will, ja sogar, wie es gesagt werden soll. Sie weiß es, weil sie eine vollständige linguistische Repräsentation des Satzes selbst generiert. Die zugrundeliegende Struktur ist bekannt, die intendierte Interpretation ist möglicherweise verfügbar, und die entsprechende syntaktische Struktur ist ebenfalls bekannt.

In einem CTS-System ist der Umweg über eine Textgenerierung nicht nur unnötig, sondern hinderlich. Orthographischer Text ist eine stark verarmte Repräsentation der Sprache. Es wäre kontraproduktiv, zunächst eine vollständige linguistische Repräsentation einer sprachlichen Äußerung in orthographischen Text zu konvertieren, nur um dann aus diesem Text wieder eine linguistische Struktur zu berechnen, die gegenüber der ursprünglichen Struktur defizitär sein muss.

Da in den Schriftsystemen der meisten Sprachen die prosodischen Strukturen allenfalls rudimentär (Satzmodus und Phrasierung durch Interpunktions) wiedergegeben werden, erwartet man sich von einem CTS-System eine signifikante Verbesserung gerade der prosodischen Qualität der synthetischen Sprache. Allerdings ist die Beziehung zum einen zwischen der symbolischen Repräsentation der Intonation und ihrer akustischen Realisierung durch Grundfrequenzkonturen und zum anderen zwischen der symbolischen Repräsentation der Intonation und der Bedeutung, die sie ausdrücken soll, selbst noch Forschungsgegenstand. Die computerlinguistische Erforschung und Modellierung der Schnittstellen zwischen Pragmatik (siehe Unterkapitel 3.7), Semantik (3.6), Syntax (3.5) und der Prosodie kann somit entscheidend zu einer Verbesserung der synthetischen Sprachqualität in Dialogsystemen und anderen Anwendungen beitragen.

3.2.3 Gemeinsamkeiten und Unterschiede

In diesem Abschnitt sollen Gemeinsamkeiten und Unterschiede zwischen der automatischen Spracherkennung und der Sprachsynthese herausgearbeitet werden, und zwar hinsichtlich der Probleme, die sich diesen beiden Teilgebieten der Verarbeitung gesprochener Sprache stellen, als auch der methodischen Herangehensweise zur Lösung dieser Probleme.

Seltene Ereignisse

Eine große Herausforderung sowohl für die Spracherkennung als auch für die Sprachsynthese ist die systematische Behandlung von Elementen der gesprochenen und geschriebenen Sprache, die eine geringe Auftretenshäufigkeit haben. Die Probleme, die durch extrem ungleichförmige Häufigkeitsverteilungen sprachlicher Ereignisse für regel- wie datenbasierte Modelle entstehen, werden häufig unterschätzt oder gar nicht erst erkannt.

Extrem schiefe **Häufigkeitsverteilungen** finden sich auf allen linguistischen Beschreibungsebenen. Am besten dokumentiert und erforscht ist die Verteilung von Worthäufigkeiten in Korpora geschriebener Sprache. Wenn man die Wörter eines großen Textkorpus in abnehmender Reihenfolge ihrer Auftretenshäufigkeit sortiert, so erhält man eine Verteilung, die annähernd einer Hyperbel $1/n$

entspricht. Das **Zipfsche Gesetz** besagt: Der Rang n von Wörtern in dieser Verteilung multipliziert mit der jeweiligen Häufigkeit der Wörter ergibt eine Konstante (Zipf 1935). Tatsächlich haben empirische Worthäufigkeitsverteilungen sogar noch unangenehmere mathematische Eigenschaften als die klassische Zipf-Verteilung (Baayen 2001).

Ungeachtet der Details lässt sich feststellen, dass einige, relativ wenige Wörter eine sehr hohe Auftretenswahrscheinlichkeit haben, während die überwiegende Mehrzahl der Wörter sehr selten bis extrem selten auftritt. Dies bedeutet, dass die meisten Wörter einer Sprache selbst in einem sehr großen Textkorpus nicht auftreten werden. Anders ausgedrückt: Die Wahrscheinlichkeit, dass ein bestimmtes seltenes Wort in einem gegebenen Satz oder einer Äußerung auftritt, mag verschwindend gering sein. Zugleich ist aber die kumulative Wahrscheinlichkeitsmasse der vielen seltenen Wörter sehr groß. Die Wahrscheinlichkeit, dass in dem betreffenden Satz *irgendein* seltenes Wort auftritt, ist daher extrem hoch.

Sprachtechnologische Systeme müssen mit dieser Eigenschaft linguistischer Einheiten umgehen können, die dazu führt, dass die meisten Einheiten im Trainingsmaterial selten oder gar nicht auftreten. Hierzu werden verschiedene statistische Techniken verwendet. So werden Modelle, die auf dem Zipfschen Gesetz oder auf dem **Good-Turing-Schätzer** basieren, eingesetzt, um die Häufigkeit von Wörtern zu schätzen, die etwa durch produktive Wortbildungsprozesse gebildet werden können, aber in einem begrenzten Korpus nie beobachtet wurden. Ein anderes Standardverfahren ist der **Expectation-Maximization-Algorithmus** (EM-Algorithmus, siehe Unterkapitel 2.4). Für den Zweck der **Lautdauervorhersage** in der Sprachsynthese hat sich das Produktsummenmodell (van Santen 1997) als besonders geeignet herausgestellt. Alle diese Modelle haben gemeinsam, dass sie eine kleine positive Wahrscheinlichkeitsmasse für Ereignisse reservieren, die im Trainingsmaterial nicht beobachtet wurden, deren Auftreten in der aktuellen Anwendung aber zu erwarten ist.

In der Sprachsynthese tritt das Problem seltener Ereignisse auf allen Ebenen auf und soll in diesem Abschnitt hinsichtlich der linguistischen Textanalyse, der Syllabifizierung, der Lautdauermodellierung und des Korpusdesigns illustriert werden.

Linguistische Textanalyse: Viele TTS-Systeme stützen sich auf ein Vollformen-Aussprachewörterbuch, ergänzt durch generische Ausspracheregeln. Wörter im Eingabetext werden im Aussprachewörterbuch nachgeschlagen oder, falls dort kein Eintrag für sie vorliegt, nach Regeln transkribiert. Das Problem bei diesem Ansatz ist die Produktivität von Wortbildungsprozessen, sowohl der Derivation als auch der Komposition, und zwar nicht nur im Deutschen, sondern generell in den meisten Sprachen. Produktive Wortbildungsprozesse generieren eine prinzipiell unbegrenzte Anzahl von Wortformen. Die Erstellung eines exhaustiven Lexikons, das alle Wörter der Sprache umfasst, ist daher unmöglich.

Die Häufigkeitsverteilungen von Wortbildungsprodukten selbst in sehr großen Korpora ähneln der Zipf-Verteilung. Da die Wahrscheinlichkeit, einem nicht im Systemwörterbuch verzeichneten Wort im Eingabetext zu begegnen, somit sehr

hoch ist, muss das TTS-System in der Lage sein, unbekannte komplexe Wörter in seine morphologischen Komponenten zu zerlegen. Dies ist gerade im Deutschen notwendig, da die Bestimmung der Aussprache aus der orthographischen Form schwierig ist und Aussprache- und Betonungsregeln Informationen über die morphologische Struktur benötigen, um die korrekte Aussprache zu inferieren.

Syllabifizierung: Sprachen wie das Deutsche oder Englische, die eine komplexe Silbenstruktur aufweisen, verfügen über eine große Anzahl unterschiedlicher Silben ($> 12\,000$), und deren Häufigkeitsverteilung ist ebenfalls Zipf-artig. Einige hundert Silbentypen – etwa die 500 häufigsten – decken rund 80% der realisierten Silben in Text- oder lautsprachlichen Korpora ab. Hingegen treten die weitaus meisten Silbentypen sehr selten auf. Erfolgversprechend sind daher datengestützte Syllabifizierungsalgorithmen, die auch unterrepräsentierten oder gar nicht gesehenen, aber in der Sprache möglichen Silben positive Wahrscheinlichkeiten zuweisen können.

Lautdauermodellierung: Ähnlich unerfreuliche Häufigkeitsverteilungen lassen sich bei der Modellierung von Lautdauern nachweisen. Die Aufgabe des Dauermodells in einem TTS-System ist die Zuweisung einer konkreten akustischen Dauer zu jedem Sprachlaut in der synthetischen Zieläußerung. Zahlreiche Faktoren beeinflussen die Dauer von Sprachlauten (und generell die temporale Struktur von Äußerungen), darunter die Identität des Lautes selbst und die seiner Nachbarn, sowie positionelle und prosodische Faktoren. Die Anzahl unterschiedlicher Konstellationen dieser Faktoren ist sprachabhängig; im Englischen und Deutschen etwa existieren mehr als 10 000 unterschiedliche Konstellationen, und ihre Häufigkeitsverteilung hat die Form einer typischen Zipf-Kurve. Auch in dieser TTS-Komponente muss ein prädiktives Modell zum Einsatz kommen, das auch Sprachlauten in a priori unwahrscheinlichen Kontexten eine realistische Dauer zuweisen kann; ein solches statistisches Modell ist das Produktsummenmodell.

Korpusdesign: Die Motivation hinter der aktuell vorherrschenden Methode der konkatenativen Synthese, der Unit Selection (siehe auch Unterkapitel 5.4), ist die weitgehende Erhaltung der Eigenschaften der natürlichen Sprache. Erreicht werden soll dies durch die Auswahl der kleinsten Anzahl möglichst langer Sprachsegmente („Bausteine“ oder „Einheiten“), die zusammengesetzt die zu synthetisierende Zieläußerung ergeben, aus einem großen lautsprachlichen Korpus.

In einer idealen Welt würde die vollständige Zieläußerung im Korpus gefunden und wiedergegeben, ohne Notwendigkeit für Einheitenverkettung und Signalverarbeitung, so dass im Endeffekt tatsächlich natürliche Sprache ausgegeben wird – ein perfektes Synthesergebnis! In der Realität lässt sich dieses Ergebnis jedoch nicht erzielen, und der Grund liegt in der Komplexität und Kombinatorik der (geschriebenen und gesprochenen) Sprache. Allenfalls in extrem eingeschränkten Domänen, z. B. Wetterdienst oder Zugauskunft, ist die Repräsentation ganzer Phrasen im Korpus und deren Auswahl zur Syntheselaufzeit realistisch. Den-

noch stehen die Chancen, eine Zieläußerung durch eine relativ geringe Anzahl von Bausteinen zu generieren, deren Länge im Durchschnitt die eines klassischen Diphons übersteigt, generell nicht schlecht, und sie steigen mit der Größe des lautsprachlichen Korpus.

Die Definition der optimalen Korpusstruktur ist so etwas wie der heilige Gral der Unit Selection-Synthese. Ein Korpus sollte maximal repräsentativ für die Sprache sein, und diese Anforderung kann nur durch sorgfältiges Design, aber nicht durch Zufallsauswahl erfüllt werden. Durch Zufallsauswahl des lautsprachlichen Korpus aus einem sehr viel größeren Textkorpus entstehen gerade die extrem schiefen Verteilungen von Wort- und Silbenhäufigkeiten, die im vorangegangenen Abschnitt besprochen wurden, mit dem Ergebnis, dass seltene Einheiten – also die massiv überwiegende Mehrheit – entweder bereits im Textkorpus nicht repräsentiert sind oder im daraus extrahierten, viel kleineren lautsprachlichen Korpus nicht mehr auftreten. Vielmehr muss das Synthesekorpus so konstruiert werden, dass auch seltene Einheiten hinreichend repräsentiert sind, auch wenn dies zu untypischen Häufigkeitsverteilungen führt. So sollten z. B. alle in der Sprache möglichen Sequenzen aus zwei Lauten (Diphone) im Korpus abgedeckt sein, ungeachtet ihrer Auftretenswahrscheinlichkeit.

Zusammenfassend ist festzuhalten, dass es verfehlt wäre, seltene sprachliche Ereignisse zu ignorieren oder ihre Modellierung in den Hintergrund zu stellen. In der Praxis garantiert die kumulative Wahrscheinlichkeit der vielen seltenen Ereignisse, dass zumindest eines von ihnen in jedem beliebigen Satz oder jeder beliebigen Äußerung auftreten wird. Ein System, das solche Phänomene nicht angemessen behandeln kann, wird ständig Fehler produzieren.

Algorithmen

In der Spracherkennung haben sich für die Behandlung seltener Ereignisse mittlerweile bestimmte Techniken etabliert, die teilweise auch für die Sprachsynthese nutzbar gemacht werden können. So ist es in der Spracherkennung beispielsweise üblich, die Eigenschaften ungesehener Einheiten (z. B. Triphone oder subphonemische Einheiten) durch Interpolation der bekannten Eigenschaften ähnlicher Einheiten vorherzusagen. Auf diese Weise können Lücken bezüglich der Repräsentativität von lautsprachlichen Korpora virtuell geschlossen werden. Dieses Vorgehen ist grundsätzlich auch auf die korpusbasierte **Unit-Selection-Synthese** anwendbar, insbesondere unter Verwendung der auch in der Spracherkennung verwendeten **Back-off**-Strategie: Wenn kein Baustein verfügbar ist, der exakt die gewünschten Eigenschaften hat, so greift man auf Bausteine zurück, die eine wohldefinierte Untermenge der gewünschten Eigenschaften aufweist. Angenommen, es würde der Vokal [a] angefordert, und zwar im Kontext eines nachfolgenden Konsonanten [t]. Falls die gewünschte [a]-Variante im Korpus nicht existiert, so könnte man auf eine Variante zurückgreifen, die im Kontext eines nachfolgenden [d] oder [s] aufgenommen wurde. Beide Laute haben mit [t] gemeinsam, dass sie alveolare Obstruenten sind; [d] unterscheidet sich von [t] durch das Merkmal [\pm stimmhaft], [s] von [t] durch das Merkmal [\pm kontinuierlich]. Den stärksten Einfluss auf die spektrale Struktur des [a] hat

aber die Artikulationsstelle des Konsonanten, und die ist in allen Fällen alveolar. Solche Back-off-Strategien sind in der Spracherkennung seit langem etabliert.

Besonders deutlich wird die Verwendung derselben oder zumindest ähnlicher Methoden in Spracherkennung und Sprachsynthese bei der Repräsentation des Netzwerks von Einheiten, die zusammen die zu erkennende oder zu synthetisierende Äußerung ergeben, und bei der Suche nach dem optimalen Pfad durch dieses Netzwerk. In dem klassischen Unit Selection-Algorithmus von (Hunt und Black 1996) wird jede Einheit im Korpus durch einen Zustand in einem Zustands-Übergangs-Netzwerk repräsentiert, wobei die Zustandskosten die Eignung („Güte“) eines Einheitenkandidaten für den gewünschten Einheitentyp und die Übergangskosten die Passgenauigkeit bei der Verkettung mit den Nachbareinheiten angeben. Das Design erinnert stark an die HMM-basierte Spracherkennung, wobei der Unterschied darin liegt, dass in der Spracherkennung probabilistische Modelle und in der Synthese Kostenfunktionen verwendet werden. Bei der aktuellen HMM-basierten Version der Unit Selection-Synthese verschwindet dieser Unterschied vollends.

Die in den beiden Typen von sprachtechnologischen Systemen verwendeten Suchalgorithmen sind in der Tat dieselben. Wenn der global optimale Pfad durch das Netzwerk von Kandidaten gefunden werden soll, wird üblicherweise der **Viterbi-Algorithmus** eingesetzt. Wenn hingegen aus Laufzeiterwägungen heraus eine nur approximativ beste Einheitensequenz akzeptabel ist, greift man üblicherweise auf die A*- oder die Strahlsuche zurück.

Prosodie

Große Unterschiede gibt es zwischen Spracherkennungs- und Sprachsynthesesystemen hinsichtlich der **Modellierung der Prosodie**. Prosodische Merkmale und Strukturen sind integrale Eigenschaften der natürlichen Sprache und spielen im natürlichen Kommunikationsprozess eine herausragende Rolle, da sie linguistische und paralinguistische Funktionen tragen (siehe auch Unterkapitel 3.1). Die Gliederung von Äußerungen durch Phrasengrenzen und Pausen, die Akzentuierung von Äußerungsteilen, die Betonung von Silben in mehrsilbigen Wörtern, die Markierung des Satzmodus, die Informationsstruktur der Äußerung und die Einbettung der Äußerung in einen Diskurs – alle diese linguistischen Funktionen werden durch intonatorische und temporale Merkmale der Lautsprache getragen. Die gleichen Merkmale sowie zusätzlich die Stimmqualität transportieren außerdem den emotionalen oder affektiven Zustand des Sprechers (z. B. Freude, Ärger, Langeweile) und seine Einstellung zum Inhalt der Äußerung oder zur kommunikativen Situation (z. B. Ironie oder nachdrücklicher Ernst). Es ist eine große Herausforderung für sprachtechnologische Systeme, diese Merkmale der natürlichen Sprache zu erkennen oder zu generieren.

Erkenner und TTS-Systeme haben deutlich unterschiedliche Anforderungen an Modelle der Prosodie, und die vorherrschenden Intonations- und Dauermodelle bieten keine einheitliche Lösung für beide Sprachtechnologien. Als Folge daraus sind prosodische Modelle in Spracherkennungssystemen bislang nur sporadisch eingesetzt worden. Im Unterschied dazu werden prosodische Modelle in jedem

TTS-System benötigt und eingesetzt, allein schon aufgrund der offensichtlichen Notwendigkeit, synthetische Äußerungen prosodisch zu strukturieren, um ein Mindestmaß an Natürlichkeit zu erreichen.

Die Notwendigkeit, Prosodie zu synthetisieren, ist jedoch keineswegs in einen allgemein akzeptierten Modellansatz gemündet, und zwar weder bei den Intonations- noch bei den Dauermodellen. Die Intonationsforschung ist äußerst divers bezüglich Theorien und Modellen. Auf der phonologischen Ebene gibt es kaum Konsens über die Basiselemente: statische Töne, Tonsequenzen, dynamische Ziele, uni- oder multidirektionale Bewegungen oder Gesten. Ebenso divers ist die Modellierung der Phonetik der Intonation, für die unter anderem eine Interpolation zwischen statischen tonalen Zielen, eine Superposition von zugrundeliegenden Phrasen und Akzentkurven oder auch die Konkatenation von Liniensegmenten vorgeschlagen wurden. Tatsächlich sind alle bedeutenden Typen von Intonationsmodellen in der Sprachsynthese mit mehr oder weniger großem Erfolg eingesetzt worden. Ein Konsensmodell ist daraus aber bislang nicht entstanden.

Etwas weniger vielfältig ist die Modellierung der zeitlichen Struktur für die Sprachsynthese. Die Rolle der Silbe als einer zentralen Verarbeitungseinheit in der menschlichen Sprachproduktion und -perzeption ist unumstritten, doch gibt es eine anhaltende Kontroverse darüber, wie die entsprechenden Effekte in einem Dauermodell am besten implementiert werden können. Die beiden aktuell erfolgreichsten Modelle haben unterschiedliche Zieleinheiten, nämlich einerseits die Silbe, deren Gesamtdauer dann auf die Einzellaute heruntergerechnet wird, aus denen sie besteht; und andererseits die Einzellaute, aus deren Dauern sich die Dauern größerer Einheiten zusammensetzen.

In der natürlichen Sprache werden tonale und temporale Eigenschaften koproduziert, und es gibt deutliche Evidenz dafür, dass der Sprecher tonale, temporale und spektrale Eigenschaften der Äußerung gemeinsam plant und sorgfältig miteinander synchronisiert, und zwar so, dass sie für den Hörer optimal wahrnehmbar sind. Die konventionelle Lösung in einem TTS-System ist allerdings keine gemeinsame Modellierung der prosodischen Strukturen, sondern eine sequenzielle Verarbeitung: zunächst wird die Dauer sprachlicher Einheiten zugewiesen, dann wird die Intonationskontur für die Äußerung berechnet.

In der Spracherkennung wurden die prosodischen und hier besonders die tonalen Merkmale, lange Zeit geradezu als „Störgeräusche“ behandelt und eliminiert. Die Lautdauern werden immerhin selbst in der Standardimplementierung der HMM-basierten Erkennung durch wiederholte Verwendung desselben Zustands (Übergang auf sich selbst) berücksichtigt, doch ein prädiktives LautdauermodeLL kommt nicht zum Einsatz. Erst seit einigen Jahren werden prosodische Merkmale in einigen Spracherkennern systematisch dekodiert und im Rahmen eines Dialogsystems dann auch ausgenutzt, und zwar einerseits durch Erkennung der linguistisch-prosodischen Struktur der Äußerung selbst und andererseits durch Detektion des emotionalen Benutzerzustandes.

3.2.4 Literaturhinweise

Eine gründliche Übersicht über die mathematisch-technischen Grundlagen der Spracherkennung und Sprachsynthese geben die empfehlenswerten Lehrbücher von Jurafsky und Martin (2009) und Huang et al. (2001). Das im deutschsprachigen Raum zum Standardwerk avancierte Buch von Schukat-Talamazzini (1995) bietet neben der erschöpfenden Themenbreite auch eine erstaunliche Tiefe, die kaum Fragen offen lässt. Ergänzend zu diesen Querschnittswerken findet man z. B. in Rabiner (1989) eine weiterführende und mit Beispielen abgerundete Darstellung der Welt der HMMs. Die Monographie von Junqua und Haton (1996) gibt einen ausführlichen Einblick in die verschiedenen Verfahren der Sprachsignalanalyse.

Für die Sprachsynthese galten die Bücher von Dutoit (1997) und Sproat (1998), die sich in vielerlei Hinsicht optimal, auch aus der Sicht des Computerlinguisten, ergänzen, bis vor Kurzem als Standardwerke. Die neuesten Methoden und Techniken der Sprachsynthese werden in Taylor (2009) ausgezeichnet dargestellt. Diese drei Bücher behandeln alle Komponenten von TTS-Systemen, setzen dabei aber unterschiedliche Schwerpunkte: Dutoit (1997) und Taylor (2009) bieten die vollständigste Darstellung der Signalverarbeitungsmethoden für die Sprachsynthese, während Sproat (1998) eine Fülle von linguistischen Textanalyseproblemen, und elegante Lösungen, für eine ganze Reihe von Sprachen bereithält. Einsatzmöglichkeiten für **Finite State Transducers** in der Sprachtechnologie (Synthese und Erkennung) werden von Mohri (1997) besprochen; dieser Artikel setzt allerdings fortgeschrittene Kenntnisse in der Automatentheorie voraus.

3.3 Morphologie

Jochen Trommer

Wörter erscheinen im Deutschen wie in vielen anderen Sprachen in verschiedenen Formen (z. B. *Zwerg*: *Zwerge*, *Zwergs*, *Zwergen*) und dienen gleichzeitig als Basis zur Bildung neuer Wörter (etwa ausgehend von *Zwerg*: *Zwerglein*, *zwerghaft*, *Gartenzwerg*). Die **Morphologie** („Formenlehre“) untersucht die systematischen Beziehungen zwischen Wörtern und Wortformen oder – prozedural ausgedrückt – die Regeln, nach denen Wörter/Wortformen gebildet werden. Die formale Umsetzung solcher Regeln in der Computerlinguistik dient dazu, Vollformenlexika zu ergänzen oder zu ersetzen.

3.3.1 Überblick

Im Zentrum dieses Unterkapitels stehen zwei Formalismen (**DATR** – eine Wissensrepräsentationssprache für lexikalisches Wissen – und **endliche Automaten**), anhand derer gezeigt werden soll, welche Probleme bei der Formalisierung morphologischer Phänomene auftreten. Dabei ergibt sich, dass einfache Finite-State-Ansätze auf der Basis endlicher Automaten für viele Phänomene inadäquat sind (Abschnitt 3.3.4). DATR erlaubt, die aufgeworfenen Probleme zu lösen, ist aber für praktische Anwendungen problematisch (Abschnitt 3.3.5). In Abschnitt 3.3.6 werden erweiterte Finite-State-Ansätze vorgestellt, die bestimmte Vorteile von DATR integrieren. Die einleitenden Abschnitte führen die relevanten Begriffe und Probleme ein (Abschnitt 3.3.2) und stellen grundlegende Modelle aus der generativen Morphologie vor (Abschnitt 3.3.3). Das Unterkapitel schließt mit einigen Bemerkungen zur mathematischen Komplexität von natürlichsprachiger Morphologie (Abschnitt 3.3.7), einer Zusammenfassung (Abschnitt 3.3.8) und weiterführenden Literaturhinweisen (Abschnitt 3.3.9).

3.3.2 Grundbegriffe und -probleme

Die genaue Definition vieler morphologischer Begriffe ist – angefangen mit dem Begriff **Wort** – immer noch Gegenstand lebhafter Diskussionen. Im Folgenden soll mit **Wort** eine abstrakte Einheit bezeichnet werden, die verschiedenen Formen zugrundeliegt und dem Eintrag eines Wörterbuchs (Lexikon) entspricht. Ein Synonym ist der technische Begriff **Lexem**. Im Gegensatz dazu sind **Wortformen** die verschiedenen Formen eines Lexems. Wortformen in diesem Sinn sind z. B. die Nominativ-Singular-Form (*der*) *Mensch* und die Akkusativ-Singular-Form (*den*) *Menschen*. Verschiedene Wortformen können phonologisch oder orthographisch zusammenfallen, z. B. die entsprechenden Formen des Lexems *Haken*: (*der*) *Haken* und (*den*) *Haken*. Dieses Zusammenfallen nennt man auch **Synkretismus**. Die Menge der Wortformen eines Lexems (oder eine Teilmenge davon) heißt **Paradigma**.

Flexion und Wortbildung

In engem Zusammenhang mit der Unterscheidung zwischen Wörtern und Wortformen steht die Aufteilung der Morphologie in **Flexion** (lat. *Beugung*) und **Wortbildung**. Letztere lässt sich weiter unterteilen in **Komposition** (Zusammensetzung, Bildung von neuen Wörtern auf der Basis mehrerer Ausgangswörter, etwa *Garten-Zwerg* oder *Zwergen-Garten* aus *Garten* und *Zwerg*) und **Derivation**, die Bildung von Wörtern auf der Basis einzelner Ausgangswörter (zu *Zwerg*: *zwergenhaft*, *Zwerglein*). Abbildung 3.21 illustriert die Beziehungen zwischen den bisher eingeführten Begriffen anhand einiger Beispiele.

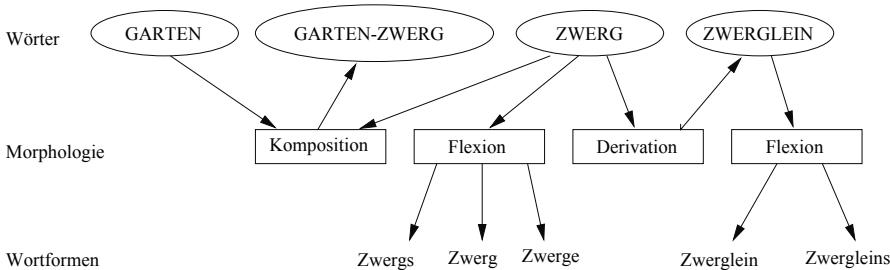


Abbildung 3.21: Flexion und Wortbildung

Morpheme

Die Graphemkette *Gartenzwerg* lässt sich in *Garten* und *Zwerg* zerlegen. Genauso wie *Gartenzwerg* haben *Garten* und *Zwerg* eine bestimmte Bedeutung, die auch in der Bedeutung des Kompositums enthalten ist: Ein Gartenzwerg ist ein (künstlicher) Zwerg, der typischerweise im Garten aufgestellt wird. Natürlich kann man auch *Garten* und *Zwerg* weiter zerlegen, z. B. *Zwerg* in *zw* und *erg*. Das sind zwar auch Graphemketten, aber ohne identifizierbare Bedeutung. *Garten* und *Zwerg* sind sogenannte **Morpheme**: Minimale Phonem/Graphem-Ketten mit einer festgelegten Bedeutung. *Gartenzwerg* fällt nicht unter diese Definition, weil es nicht minimal ist. *zw* und *erg* sind keine Morpheme, weil sie keine festgelegte Bedeutung haben. Eine abstraktere Definition des Morphembegriffs, die auf dieser Grundidee beruht, findet sich im Folgenden.

Klassifikation von Morphemen

Garten und *Zwerg* sind sehr spezielle, sogenannte **freie Morpheme**. Freie Morpheme können auch ohne den Kontext anderer Morpheme geäußert werden. Morpheme, die diese Eigenschaft nicht haben, heißen **gebunden**. Z. B. ist das Morphem *-s* in (*des*) *Zwergs* mit der Bedeutung *Genitiv Singular* gebunden, da es nur nach einem Nomen auftauchen kann.

Eine weitere Einteilung für Morpheme ist die in **Grund- und peripherie Morpheme**. Grundmorpheme stellen den Ausgangspunkt für Derivation und Flexion dar (*Zwerg* in den angeführten Beispielen). Statt von Grundmorphem spricht

man auch von **Wurzel**. Peripherie gebundene Morpheme heißen **Affixe**. Affixe, die vor Wurzeln auftreten, heißen **Präfixe**, solche, die nach Wurzeln stehen, **Suffixe**.

Morphemkombinationen ohne Flexionsaffixe werden als **Stämme** bezeichnet. Z. B. ist *kauf* sowohl eine Wurzel als auch ein Stamm, *ver-kauf* ist ein Stamm, aber keine Wurzel; *ver-kauf-st* und *kauf-st* sind weder Stämme noch Wurzeln, da sie beide das Flexionsaffix *-st* enthalten.

Allomorphe

Oft scheinen verschiedene Morpheme zusammenzugehören: Sie realisieren dieselbe Bedeutung in verschiedenen morphologischen Kontexten. Z. B. wird Plural in *Kind-er* durch *-er* in *Wind-e* aber durch *-e* ausgedrückt. Diese Beobachtung ist die Grundlage für einen etwas abstrakteren Morphembegriff, bei dem man Ketten mit festgelegter Bedeutung nicht als Morpheme, sondern als **Morphe** bezeichnet und Morphem in Abhängigkeit von Morph definiert.

Man spricht davon, dass eine Menge von Morphemen in **komplementärer Verteilung** zu einander steht, wenn es keinen Kontext gibt, in dem wahlweise das eine oder das andere Morph auftauchen kann: Das trifft auf *-e* und *-er* zu, da *-er* nicht rechts von *Wind* (**Wind-er*) und *-e* mit der Bedeutung Plural nicht rechts von *Kind* (**Kind-e*) vorkommen kann.

Ein Morphem ist dann eine maximale Menge von bedeutungsgleichen Morphemen in komplementärer Verteilung. Die Elemente dieser Menge sind die **Allomorphe** oder **Morphemalternanten** des Morphems. Ein Beispiel ist die Menge aller Pluralmorpheme des Deutschen: {-e, er, -s,...}, die zusammen das Pluralmorphem bilden, dessen Allomorphe wiederum *-e*, *er*, *-s*, ... sind.

Morphophonologie

Oft unterscheiden sich verschiedene Allomorphe eines Morphems nur minimal voneinander. Man spricht dann von **partieller Allomorphie**. Ein Beispiel ist das Nomen das in *hand-lich* als *hand*, aber in *Händ-chen* als *Händ* realisiert wird.

Derartige Allomorphie lässt sich in vielen Fällen auf reguläre phonologische Prozesse zurückführen, das heißt auf regelmäßige Lautveränderungen, die in bestimmten lautlichen Kontexten einheitlich angewendet werden.

Beispiel 3.3.1

Ein typischer phonologischer Prozess ist die Auslautverhärtung, die bestimmte stimmhafte Konsonanten (*b,d,g*) am Ende einer Silbe stimmlos werden lässt (*p,t,k*) (siehe Unterkapitel 3.1). Die Auslautverhärtung sorgt dafür, dass z. B. *Hund*, *Trieb* und *Weg* am Ende mit stimmlosem Konsonanten ausgesprochen werden:

[hʌndə]	→	[hʌnt]	(Hunde/Hund)
[tri:bə]	→	[tri:p]	(Tribe/Trieb)
[we:gə]	→	[we:k]	(Wege/Weg)



Allomorphie dieser Art wird auch als **phonologisch konditionierte** oder **un-eigentliche Allomorphie** bezeichnet.

Auch der Prozess der Umlautung, der aus *Hand* vor *-chen* *Händ* macht, sieht auf den ersten Blick wie ein phonologischer Prozess aus. Hier wird der Stammvokal des Ausgangsmorphems einheitlich nach vorne ($a \rightarrow ä$, $u \rightarrow ü$, $o \rightarrow ö$) verschoben.

<i>Hand</i>	<i>Schloss</i>	<i>Mut</i>
<i>Händchen</i>	<i>Schlösschen</i>	<i>Mütchen</i>

Der Umlaut ist aber zumindest teilweise morphologisch determiniert oder **morphophonemisch**, da sich kein einheitlicher phonologischer Kontext angeben lässt, in dem Umlaut eintritt. Ein extremes Beispiel für diese **morphologische Konditionierung** ist die Umlautung bei der Pluralbildung von Wörtern wie *Mutter* (pl.: *Mütter*) und *Vater* (pl.: *Väter*). Die (nicht umgelautete) Ausgangsform für *Väter* ist offensichtlich *Vater*. Wenn der Umlaut eine rein phonologische Regel wäre, müsste diese phonologische Form also immer zur Umlautung führen. Das ist aber für die Singular-Form offensichtlich falsch, die ebenfalls *Vater* heißt, aber keine Umlautung auslöst.

Nichtkonkatenative Morphologie

Eine alternative Möglichkeit ist es, den Umlaut als eine Form von **nichtkonkatenativer Morphologie** zu betrachten. *Nichtkonkatenativ* bedeutet in diesem Zusammenhang, dass eine Kategorie wie Plural nicht durch ein Morphem, sondern durch andere Mittel wie den Umlautungsprozess ausgedrückt wird.

Weitere Formen von nichtkonkatenativer Morphologie sind Prozesse, bei denen Morpheme verkürzt (**subtraktive Morphologie**) oder teilweise wiederholt werden (**Reduplikation**). Ein Fall von subtraktiver Morphologie ist die Bildung von *i*-Wörtern (z. B. *Katharina* ⇒ *Kathi*, siehe Beispiel 3.3.3). Reduplikation taucht etwa bei der Bildung des lateinischen Perfekt auf: *pend-o-ich hänge*; *pend-i-ich hing*.

Eine Form von Lautmodifikation, die noch deutlicher morphologisch bedingt ist als der Umlaut, ist der sogenannte **Ablaut**, der im Deutschen bei der Flexion vieler unregelmäßiger (**starker**) Verben relevant ist. Während der Umlaut zwar nicht in phonologisch einheitlichen Kontexten auftritt, aber einen phonologisch einheitlichen Prozess darstellt (er frontiert Vokale), hat der Ablaut u.U. entgegengesetzte Effekte. Z. B. wird im Imperfekt von *schieben i* zu *o* (*schob*) während bei *stoßen o* zu *i* wird (*stieß*).

Grundprobleme der morphologischen Beschreibung

Von den spezifischen Problemen, die sich bei der Beschreibung und Formalisierung von Morphologie ergeben, sollen im weiteren Verlauf dieses Unterkapitels vor allem die folgenden diskutiert werden:

Neutralisierung Wörter/Wortformen spiegeln ihre Bedeutung oder Funktion oft nur teilweise wieder. Z. B. kann *Haken* sg. oder pl. sein. Man spricht in

diesem Zusammenhang davon, dass ein Kontrast (hier der zwischen sg. und pl.) **neutralisiert** ist.

Nichtkonkatenativität Wie lassen sich Phänomene der nichtkonkatenativen Morphologie beschreiben?

Regularitäten und Ausnahmen Bestimmte Formen oder Affixe sind regelmäßig, andere ganz oder teilweise unregelmäßig. Z. B. erscheint die Pluralendung *-er* nur bei einer begrenzten Anzahl von Nomen im Neutrum, während viele Nomen – gerade auch Neubildungen – den Plural mit *-e* verwenden.

Allomorphie und Phonologie Wie lassen sich Fälle beschreiben, in denen Allomorphie ganz oder teilweise phonologisch bedingt ist?

Im nächsten Abschnitt werden dazu Lösungsansätze aus dem Bereich der generativen Morphologie diskutiert. Die Modelle, die dabei vorgestellt werden, sind auch für computerlinguistische Formalismen relevant, die in den Abschnitten 3.3.4 bis 3.3.6 besprochen werden.

3.3.3 Modelle aus der Generativen Linguistik

Die meisten bisher eingeführten Begriffe sind ursprünglich analytisch definiert, z. B. über Verfahren, die es erlauben, Wörter oder Morpheme in Äußerungen zu identifizieren. In der generativen Morphologie (siehe Spencer 1991) ist die Herangehensweise genau umgekehrt. Es werden Modelle konstruiert, die durch explizite Regeln Wörter aus einem gegebenen Input ableiten. In Abschnitt 3.3.3 werden die konkurrierenden Grundmodelle der generativen Morphologie eingeführt, in den darauffolgenden Abschnitten wird diskutiert, inwieweit diese Modelle Lösungen für die genannten Problemfelder bieten. Dabei ergibt sich ein Spannungsfeld zwischen der Restriktivität des Modells und seiner Flexibilität, mit den Problemen umzugehen, ein Tatbestand, der auch bei computerlinguistischen Modellierungen zentral ist.

Drei Modelle von Morphologie

In der generativen Morphologie gibt es drei Grundmodelle dafür, wie morphologische Regeln und der Input, auf den sie angewendet werden, konzipiert sind. Im Folgenden wird von **morphembasierten, wortbasierten** und **realisierungsbasierten Ansätzen** gesprochen. In der Praxis werden natürlich oft Elemente der verschiedenen Richtungen kombiniert oder zusätzliche Elemente eingeführt. In **wortbasierten Ansätzen** werden durch die Anwendung von Regeln aus Wörtern neue Wörter gebildet. In **realisierungsbasierten Ansätzen** wird von der vorgegebenen Bedeutung oder Funktion einer Wortform ausgegangen. Die Regeln legen dann fest, wie diese durch eine Wortform realisiert werden. In **morphembasierten Ansätzen** werden Morpheme kombiniert, um vollständige Formen zu erhalten: Die Konzeption dieser Grundmodelle wird in Abbildung 3.22 an einem Beispiel illustriert.

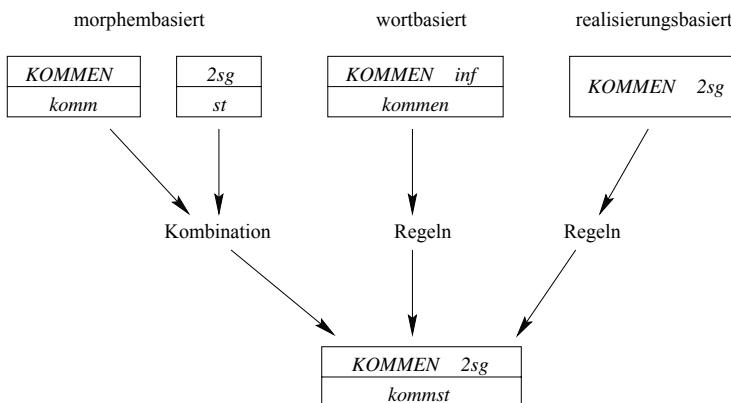


Abbildung 3.22: Die drei Grundmodelle der generativen Morphologie

KOMMEN steht hier für die Bedeutung des Verbs *kommen*. Solche Symbole, die die Bedeutung eines Morphems identifizieren, werden im Folgenden auch **abstrakte Morpheme** genannt. Das Ergebnis der Ableitung ist in allen drei Fällen die phonologische Form *kommst* mit der Bedeutung *KOMMEN 2sg*. Diese wird im wortbasierten Ansatz, ausgehend von der Infinitivform des Verbs (*inf*), im realisierungsbasierten auf Basis der zu realisierenden Bedeutung *KOMMEN 2sg* und im morphembasierten, ausgehend von den Teilmorphemen *KOMMEN/-komm* und *2sg/st*, realisiert.

Im nächsten Abschnitt werden kurz die Vorteile von realisierungsbasierten Ansätzen herausgearbeitet, die für das weitere Unterkapitel relevant sind. Ein morphembasierter Ansatz wird in Abschnitt 3.3.4 anhand von endlichen Automaten beschrieben, ein realisierungsbasierter im Rahmen von DATR in Abschnitt 3.3.5. Wortbasierte Ansätze spielen vor allem bei der Modellierung von Wortbildung und in Merkmals-Wert-Formalismen Anwendung. Diese werden hier aus Platzgründen nicht ausführlicher behandelt.

Formale Restriktivität vs. Flexibilität

Der formal restriktivste Ansatz ist der morphembasierte. Anstatt spezifischer Regeln gibt es hier im Idealfall nur eine Operation, die Verkettung, die die phonologischen Formen einzelner Morpheme kombiniert. Natürlich muss dabei festgelegt werden, welche Morpheme in welcher Reihenfolge verkettet werden können. Dies kann z. B. durch eine Phrasenstrukturgrammatik oder durch endliche Automaten geschehen (siehe Abschnitt 3.3.4).

Im Unterschied dazu enthalten die Regeln in realisierungsbasierten Ansätzen sehr spezifische Informationen. Z. B. müssten die Regeln für das Beispiel in Abbildung 3.22 festlegen, dass Formen in der 2. Person singular (2.sg) auf *-st* enden. Dies ergibt sich bei morphembasierten Ansätzen schon aus den entsprechenden Morphemen. Realisierungsbasierte Ansätze sind andererseits aber auch in vielen

Fällen flexibler.

Das zeigt sich u.U. in Fällen von Unterspezifikation, wie bei der Pluralform von *Haken*, die einfach aus dem Nominalstamm *Haken* besteht. Wenn in dieser Form ein Pluralmorphem enthalten ist, dann muss es phonologisch leer sein, ein sogenanntes **Nullaffix**. Solche „Geisteraffixe“ werden aber von vielen Linguisten abgelehnt. In einem realisierungsbasierten Ansatz ist die Annahme eines solchen Nullmorphems überflüssig, wenn angenommen wird, dass die Grammatik keine Regel enthält, die Plural für diesen Stamm realisiert.

Außerdem sind Regeln in realisierungsbasierten Ansätzen meistens nicht auf Affigierung beschränkt, sondern können Stämme in beliebiger Weise verändern. Z. B. könnte man die Pluralbildung von *Vater* und *Mutter* durch eine Regel beschreiben, die den Stammvokal frontiert. Das erlaubt eine unmittelbare Umsetzung von **nichtkonkatenativer Morphologie**.

Defaultregeln und Neutralisierung

Schließlich erlauben realisierungsbasierte Ansätze die Anwendung von **Defaultregeln**. Defaultregeln sind Regeln, die nur dann angewandt werden, wenn nicht ausdrücklich eine andere Regel mit höherer Priorität gegeben ist. Man könnte z. B. für die deutschen Pluralendungen die folgenden Regeln annehmen, wobei (3.7) die Regel mit der höheren Priorität sein soll:

(3.7) (Plural-Nomen) X/Kind, Ei ... → X+er

(3.8) (Plural-Nomen) X → X+e

X steht hier für die phonologische Form des Stamms, an den -er bzw. -e suffigiert wird. Die Bedingung *Kind, Ei ...* in (3.7) legt fest, dass die Regel nur bei den angegebenen Nomen angewendet werden darf. Da die Anwendung von (3.8) bei allen Nomen möglich ist, überschneiden sich die Bedingungen für die Anwendung der Regeln, aber da (3.7) höhere Priorität hat, wird (3.8) nur angewendet, wenn die erste Regel nicht anwendbar ist. Andernfalls wird (3.8) **blockiert**.

Ein etwas anderer Einsatzbereich für Defaultregeln sind Fälle von **Neutralisierung**. Z. B. enden die meisten Formen des Verbparadigma gleichlautend auf -(e)n. (z. B. *klau-en* = Infinitiv, Präsens 1.pl. und Präsens 3.pl). Hier könnte man annehmen, dass die Regel, die -(e)n an Verben suffigiert als Default anderen Regeln gegenübersteht, die gegebenenfalls Endungen wie -(e)st, 2.sg. einführen. Dadurch wird es überflüssig, drei verschiedene Regeln anzunehmen, die -(e)n affigieren.

Allomorphie und Phonologie

In strikt morphembasierten Ansätzen wird oft versucht, einen möglichst großen Anteil von Allomorphie durch phonologische Regeln abzudecken, was oft nur durch abstrakte Ausgangsrepräsentationen möglich ist.

Beispiel 3.3.2

Im Fall der Umlautung in *Väter* lässt sich ein zugrundeliegendes Pluralsuffix *-I* annehmen, das Umlautung auslöst, dann aber durch eine weitere phonologische Regel gelöscht wird:

Repräsentation	phonologische Regel
Vater-I	
↓	a → ä / <u>I</u> (a wird vor I zu ä)
Väter-I	
↓	I → Ø (I wird gelöscht)
Väter	

□

Wenn Segmente wie *I* in Beispiel 3.3.2 nicht an der Oberfläche auftauchen und nicht unabhängig motiviert werden können, werden sie auch als **diakritische Symbole** bezeichnet, d.h. als Symbole, die ausschließlich dem Zweck dienen, bestimmte Regeln auszulösen. Der stipulative Charakter von **Diakritika** ist oft als Argument angeführt worden, solche Alternationen durch morphologische Regeln zu beschreiben, die unmittelbar von morphologischen Merkmalen ausgelöst werden, wie es für realisierungsbasierte Ansätze typisch ist (siehe Abschnitt 3.3.3).

Die realisierungsbasierte Sichtweise von Morphologie kann in vielen Fällen auch dadurch umgangen werden, dass man Morpheme als Merkmalsstrukturen beschreibt, die nicht von vornherein an Segmente gebunden sind. Dies ist der Grundgedanke der **Autosegmentalen Morphologie**. Z.B. lässt sich für den Umlaut in *Väter* und *Mütter* ein Pluralmorph annehmen, das lediglich aus dem phonologischen Merkmal *vorne* besteht (Wiese 1994). Wird dieses Merkmal mit *Vater* kombiniert, assoziiert eine phonologische Regel dieses Merkmal mit dem Stammvokal, so dass aus dem *a* in *Vater* ein vorderer Vokal wird (*ä*). Die Wechselwirkung von Morphologie und Phonologie spielt auch in der sogenannten **Prosodischen Morphologie** eine entscheidende Rolle. Prosodie bezeichnet phonologische Einheiten wie Silben und metrische Füße, die größer sind als Segmente. Viele morphologische Prozesse können nur in Bezug auf prosodische Einheiten wie die Silbe beschrieben werden.

Beispiel 3.3.3

Bei der Bildung von *i*-Wörtern werden die Ausgangsstämme so verkürzt, dass sich genau zwei Silben ergeben. Die eckigen Klammern [] markieren Silbgrenzen.

- a. [Ka][tha][ri][na] → [Ka][thi]
- b. [A][bi][tur] → [A][bi]
- c. [Stu][dent] → [Stu][di]

□

Die zugrunde liegende Regel lautet also in etwa: „Entferne so viele Segmente vom rechten Rand des Ausgangswortes, dass sich (unter Suffigierung von *-i*) zwei Silben ergeben.“ Man beachte, dass in a. 5, in b. 3 und in c. 4 Segmente abgeschnitten werden. Umgekehrt enthält das Ergebniswort jeweils 5 (a.), 3 (b.) bzw. 5 (c.) Segmente. Regeln der Form: „Entferne *n* Segmente vom rechten Rand“ oder „Entferne soviele Segmente vom rechten Rand, dass die Form genau *n* Segmente enthält“ können diese Formen nicht systematisch erfassen. Der Bezug auf die Silbe ist also wesentlich.

3.3.4 Morphologie mit endlichen Automaten

Endliche Automaten (siehe 2.2) sind der einfachste und zugleich verbreitetste Formalismus bei der Modellierung von morphologischen Regelsystemen. In diesem Abschnitt soll gezeigt werden, wie sich mit endlichen Automaten eine einfache morphembasierte Modellierung von Morphologie entwickeln lässt, die – aufgrund der formalen Eigenschaften von endlichen Automaten – auch eine optimale Abstimmung mit phonologischen Regeln erlaubt. Wie nach der bisherigen Diskussion zu erwarten, ergeben sich aus der Morphembasiertheit dieses Modells Probleme bezüglich nichtkonkatenativer Morphologie und Neutralisierung. Hinzu kommt die Problematik nichtlokaler Abhängigkeiten, die sich speziell aus der Verwendung endlicher Automaten ergibt.

Nach der Diskussion derselben Daten in DATR (Abschnitt 3.3.5) werden in Abschnitt 3.3.6 erweiterte Finite-State-Methoden vorgestellt, die – v.a. durch die Möglichkeit, endliche Automaten auf verschiedene Weisen zu kombinieren – erlauben, die meisten dieser Probleme zu entschärfen und auch realisierungsisierte Elemente in einen Finite-State-Rahmen zu integrieren.

Einfache endliche Automaten

In den folgenden Abschnitten wird die Umsetzung eines kleinen Ausschnitts des deutschen Verbparadigmas und einiger zusätzlicher Daten aus dem Bereich der *i*-Wörter durch endliche Automaten diskutiert. Dies soll vor dem hypothetischen Hintergrund einer praktischen Anwendung stehen, nämlich der Aufgabe, im Rahmen eines Rechtschreibprogramms wohlgeformte deutsche Verben zu erkennen. Um auch den Analyseaspekt mit einzubeziehen, werden die endlichen Automaten zu Finite-State-Transducern (FSTs) erweitert. Ausgangspunkt der Formalisierung ist das folgende Verbparadigma:

Person	sg.	pl.
1	<i>wate</i>	<i>waten</i>
2	<i>watest</i>	<i>watet</i>
3	<i>watet</i>	<i>waten</i>

Diese Formen lassen sich durch den Automaten in Abbildung 3.23 darstellen. Das entspricht nicht ganz der hypothetischen Aufgabenstellung, da ein Rechtschreibprogramm Wortformen nicht vorsegmentiert erhält, sondern als unstrukturierte Graphemketten, d.h. der Automat in Abbildung 3.23 muss durch den in Abbildung 3.24 ersetzt werden.

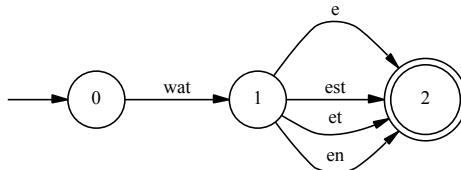


Abbildung 3.23: Formen von *waten* in einem endlichen Automaten (Morpheme)

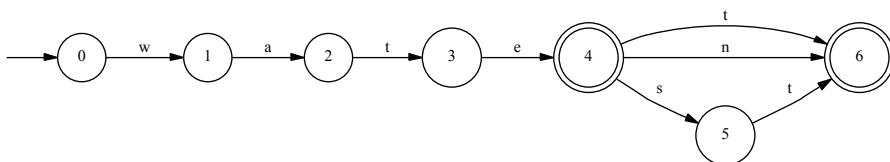
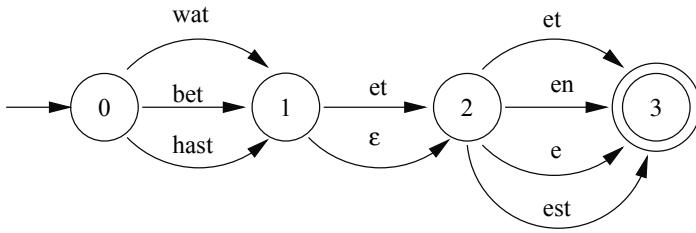


Abbildung 3.24: Formen von *waten* in einem endlichen Automaten (Segmente)

Automaten, wie der in Abbildung 3.23, bei denen die Übergänge mit Segmentketten annotiert sind, lassen sich aber in trivialer Weise auf Segmentautomaten wie den in Abbildung 3.24 abbilden, indem man neue Zustände einführt und jeden Übergang über eine Segmentkette durch eine Sequenz von neueingeführten Übergängen über die entsprechenden Einzelsegmente ersetzt. Der Automat in Abbildung 3.24 wäre damit einfach eine andere Notation für den in Abbildung 3.23. Da Automaten mit Segmentsequenzen übersichtlicher sind, werden auch im Folgenden Segmentautomaten in dieser Weise geschrieben. Aufgrund der Äquivalenz von endlichen Automaten und regulären Ausdrücken kann man auch einen Schritt weiter gehen und Automaten ganz ohne Zustände notieren. Der folgende reguläre Ausdruck in ERA-Notation (siehe Unterkapitel 2.2, S. 72) ist ebenfalls äquivalent zu dem Automaten in Abbildung 3.24:

$$(3.9) \text{ wate}(t \mid n \mid st)?$$

An diesem Punkt sind die benutzten Automaten relativ trivial. Die Anzahl der abgedeckten Formen lässt sich aber durch einfache Erweiterungen steigern, etwa indem man weitere Verbstämme und das Imperfektmorphem *-et* einfügt wie in Abbildung 3.25.

Abbildung 3.25: Präsens- und Präteritumformen von *waten*, *beten* und *hasten*

Finite-State-Transducer

Eine zusätzliche Erweiterung wird notwendig, wenn man die Aufgabenstellung ausdehnt und Wortformen nicht nur auf ihre Wohlgeformtheit hin überprüfen, sondern auch analysieren will, also z. B. für *beteten* die Information erhalten möchte, dass es sich um eine Imperfekt-Form 1.pl. oder 3.pl. handelt. Dies kann man durch den Gebrauch eines **Finite-State-Transducers (FST)** erreichen, der nicht Mengen einzelner Ketten, sondern Abbildungen von Ketten auf andere Ketten kodiert (Abbildung 3.26).

Die rechten Symbole sind hier abstrakte Morpheme, die linken Segmentketten. Der Transducer bildet also Ketten abstrakter Morpheme auf Ketten von (phonologischen/graphemischen) Segmenten ab. Der Transducer ermöglicht sowohl die Analyse (z. B. *beteten* \Rightarrow V impf 1pl) als auch die Generierung von Formen (V impf 1pl \Rightarrow *beteten*). Außerdem kann er dazu verwendet werden, um zu prüfen, ob eine gegebene Abbildung wie V impf 1pl:*beteten* korrekt ist.

Dass der Transducer komplexer ist als der entsprechende Automat, liegt an den Synkretismen: Die 1.pl. fällt mit der 3.pl. und die 3.sg. mit der 2.pl. zusammen. Eine vereinfachte Notation könnte die Übergänge über *en* : 1pl und *en* : 3pl zusammenfassen, z. B. als *en* : {1pl, 3pl}.

Integration von Morphologie und Phonologie

Praktische computerlinguistische Anwendungen sind darauf angewiesen, Morphologie- und Phonologiekomponenten zu integrieren. Die Endungen für die Verbformen der 2.sg. sind beispielsweise weitgehend phonologisch, d.h. durch den letzten Laut des Stamms, bedingt:

Stamm auf s	Stamm auf t	andere Stämme
<i>hass -t</i>	<i>wat-est</i>	<i>hol-st</i>
<i>lös-t</i>	<i>hast-est</i>	<i>schau-st</i>

Wenn der Verbstamm auf *s* auslautet, ist die Endung *-t*. Endet der Stamm selbst auf *t* ist die Endung *-est*. Nach allen anderen Lauten erhält man *-st*. Wie man anhand anderer Verbstämme leicht nachprüfen kann, ist dies ein Fall

von weitgehend **phonologisch konditionierter Allomorphie**. Natürlich ist es möglich, für jeden Verbstamm festzulegen, ob er die 2.sg. mit *-t*, *-st* oder *-est* bildet, aber es ist weniger aufwendig, für das 2.sg.-Affix eine einheitliche Ausgangsrepräsentation anzunehmen, etwa *-st*, die dann durch phonologische Regeln auf die korrekte Form abgebildet wird. Für Stämme auf *t* lässt sich das durch die folgende Regel erreichen, die u.a. *watst* auf *watest* abbildet:

$$(3.10) \epsilon \rightarrow e : t __ st$$

Solche Regeln können wie in der *Zwei-Ebenen-Morphologie* (Koskenniemi 1983) ebenfalls durch Finite-State-Transducer umgesetzt werden (siehe auch Unterkapitel 3.1). Die Regel in (3.10) wird dann zum Transducer in Abbildung 3.27. *X:X* steht hier für Identitätsübergänge aller Buchstaben des Alphabets, für die vom jeweiligen Ausgangszustand kein anderer Übergang angegeben ist. Z. B. bildet der Übergang *X:X* von *T* nach *X* alle Buchstaben außer *t* und *s* auf sich selbst ab. Der Transducer ist so konstruiert, dass er beim Einlesen von *t* in jedem Fall in den Zustand *T* übergeht. Wenn jetzt im Eingabestring *s* und *t* folgen, ist die einzige mögliche Zustandsfolge *TS T*, die dazu führt, dass *tst* auf *test* abgebildet wird.

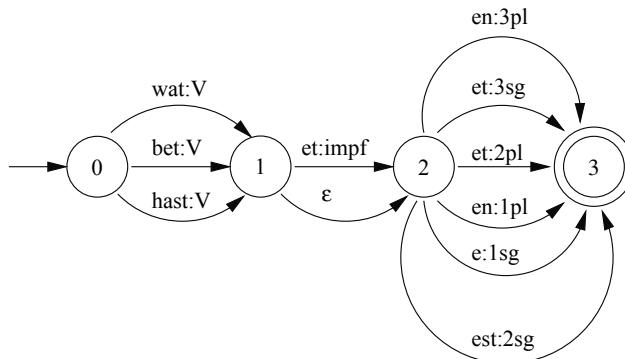


Abbildung 3.26: Verbformen in einem FST

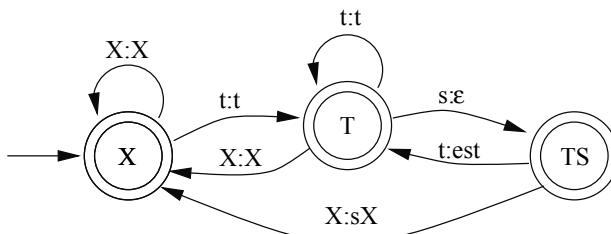


Abbildung 3.27: Die Regel in (3.10) als FST

Da die Komposition von Finite-State-Transducern (mit bestimmten Einschränkungen) wiederum einen FST ergibt, kann man Morphologie-Transducer wie in

Abbildung 3.26 mit phonologischen Transducern wie in Abbildung 3.27 zu FSTs komponieren, die Bedeutungen wie *V 2sg* direkt auf phonologische Oberflächenformen abbilden (z. B. *betest*).

Defaults und Neutralisierung

Während sich das Zusammenfallen von 1. und 3. Plural im Präsens (siehe den Transducer in Abbildung 3.26) durch eine entsprechende Notation entschärfen lässt, ist die Neutralisierung von 3.sg. und 2.pl. schwieriger. Diese fallen nämlich bei genauem Hinsehen nur im Präsens zusammen ((*er*) *betet*, (*ihr*) *betet*) im Imperfekt aber ist die 3.sg. identisch zur 1.sg. ((*ich/er*) *betete*). Um dies im Transducer korrekt wiederzugeben, muss die übersichtliche Struktur etwas aufgebrochen werden (Abbildung 3.28). Störend an diesem Transducer ist, dass

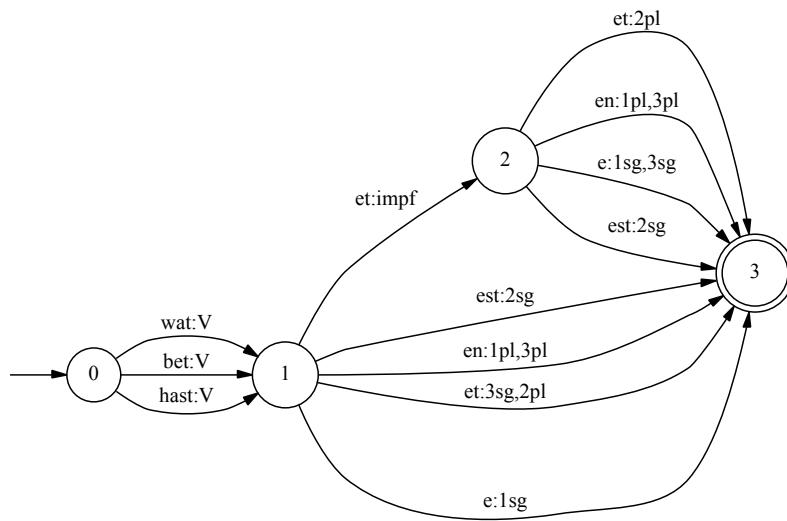


Abbildung 3.28: Alle Imperfekt- und Präsensformen in einem FST

identische Allomorphe zweimal dargestellt werden müssen, was einerseits theoretisch unbefriedigend ist und andererseits die Übersichtlichkeit und Modifizierbarkeit des Transducers beeinträchtigt. Hier wäre es sinnvoll, Neutralisierung durch einen Defaultmechanismus beschreiben zu können: -e lässt sich als die Defaultrealisierung von 1./3.sg. betrachten, während -t für die 3.sg. nur im Kontext von Präsensformen erscheint. Der aufgeblähte Automat in Abbildung 3.28 ließe sich vermeiden, wenn sich der Übergang *et:3sg* unabhängig vom Rest-Transducer auf Imperfektformen beschränken ließe, und man für *e:1sg,3sg* festlegen könnte, dass es nur im Default-Fall möglich ist, wenn *et:3sg* ausgeschlossen ist.

Nichtkonkatenative Morphologie: Starke Verben

Probleme ergeben sich auch mit nichtkonkatenativen morphologischen Prozessen wie dem Ablaut bei starken Verben, bei dem die Stammvokale in bestimmten morphologischen Kontexten wechseln, z. B. *raten*.

Person	Präsens		Imperfekt	
	sg.	pl.	sg.	pl.
1	rate	raten	riet	rieten
2	rätst	ratet	rietst	rietet
3	rät	raten	riet	rieten

Der Transducer in Abbildung 3.29 beschreibt wieder die möglichen Formen.

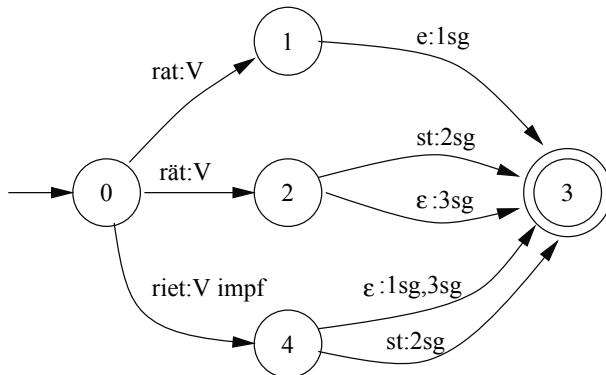


Abbildung 3.29: Präsens- und Imperfekt-Formen von *raten* in einem FST

Theoretisch unbefriedigend an dieser Darstellung ist, dass die weitgehende Identität der Stammformen, die sich nur durch den Vokalwechsel unterscheiden, nicht erfasst wird. Will man weitere Verben derselben Flexionsklasse hinzufügen, muss man wiederum drei verschiedene Übergänge einfügen, da der Vokalwechsel nicht explizit repräsentiert ist.

Eine Alternative dazu bietet die Behandlung von Ablaut durch phonologische Regeln. Während das die meisten Linguisten für den Ablaut ablehnen würden (siehe Abschnitt 3.3.2), wäre es für den produktiveren Umlaut eine plausible Analyse. In beiden Fällen stellt sich aber die Frage, was die Vokalverschiebung auslöst. In der Finite-State-Phonologie (z. B. Koskenniemi 1983) werden in solchen Fällen oft **diakritische Symbole** angenommen. In Anlehnung an die Umlautanalyse aus Beispiel 3.3.2 könnte man z. B. annehmen, dass *rät* zugrundeliegend als *rAt* repräsentiert wird und das 2/3.sg.-Morphem durch das Symbol @ realisiert wird, d.h. der Morphologie-Transducer bildet *V 3sg* auf *rAt@* ab:

Repräsentation	phonologische Regel
rAt@	
↓	A → ä / <u> </u> @ (A wird vor @ zu ä)
rät@	
↓	@ → Ø (@ wird gelöscht)
rät	
rAtε	
↓	A → a (A wird zu a)
rate	

A wird in dem Beispiel zu ä wenn ein @ folgt und andernfalls zu a. @ ist ausschließlich dadurch motiviert, dass es die Anwendung der Umlautungsregel in den erforderlichen Kontexten auslöst.

Nichtlokale Abhängigkeiten

Die Bildung des Partizips II (*gebetet*, *gewatet*, etc.) verlangt in den meisten Fällen sowohl ein Präfix als auch ein Suffix (Kombinationen von Präfixen und Suffixen heißen auch **Zirkumfixe**.) Es ist relativ einfach, einen Automaten zu konstruieren, der genau diese Formen abdeckt. Er entspricht dem regulären Ausdruck in (3.11).

(3.11) ge (bet | wat) et

Komplikationen ergeben sich, wenn man versucht, die Bildung des Partizips zusammen mit anderen Flexionsformen in einem Automaten oder Transducer zu erfassen.

Im Transducer von Abbildung 3.30 lässt sich auch die ungrammatische Form **ge-bet-est* ableiten, während im Transducer in Abbildung 3.31, der Übergang bet:V zweimal auftaucht:

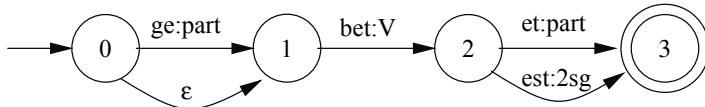


Abbildung 3.30: Verbflexion mit Partizipialformen: Übergenerierender FST

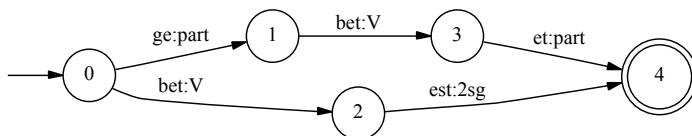


Abbildung 3.31: FST mit zwei Übergängen für bet:V

Da die meisten nicht-derivierten Verben im Deutschen auf dieselbe Weise gebildet werden wie *waten* und *beten* heißt das, dass jeder Verbstamm durch mindestens zwei Übergänge repräsentiert werden muss, was den Transducer weiter aufbläht. Dies lässt sich nicht durch eine geschicktere Konstruktion des Transducers vermeiden. Das zentrale Problem ist, dass in endlichen Automaten/Transducern Übergangsmöglichkeiten nur vom vorherigen Zustand abhängen. Das schließt eine direkte Modellierung **nichtlokaler Abhängigkeiten**, d.h. von Auftretensbeschränkungen über Elementen, die nicht unmittelbar nebeneinander liegen, aus.

Prosodische Morphologie

Während in der Derivationsmorphologie oft authentische Fälle von nichtlokaler Abhängigkeit vorkommen, ist *ge-* zumindest nicht eins zu eins an *-et* gekoppelt: Für viele Verbstämme wird das Partizip II mit *ge- -en* gebildet (*ge-hau-en gefahr-en*, etc.). Bei genauerem Hinsehen ist es fraglich, ob *ge-* überhaupt ein echtes Präfix ist. Oft reichen auch *-et* und *-en* alleine zur Bildung des Partizips (*verlor-en*, *überhol-t*, *abstrahier-t*). Das Präfix *ge-* tritt genau dann auf, wenn der Verbstamm mit einer betonten Silbe beginnt. Das Problem fällt also in den Bereich der prosodischen Morphologie.

Das löst die Frage, wie die Partizipialbildung zu behandeln ist, allerdings nicht, sondern führt zu dem Problem, wie sich prosodische Regularitäten in einem Transducer repräsentieren lassen. Silben sind nach einem weitverbreiteten Verständnis komplexe Gebilde mit einer internen Phrasenstruktur (siehe Abschnitt 3.3.6). Endliche Automaten sind hingegen über Segmenten definiert. Problematisch ist auch, dass sich die bisherige Aufgabenstellung auf orthographische Repräsentationen bezieht, in denen abgesehen von Silben auch Betonung nicht repräsentiert ist. Noch komplexer ist das Problem der Bildung von *i*-Wörtern. Die relevanten Daten aus Beispiel 3.3.3 sind hier noch einmal aufgeführt:

- a. [Ka][tha][ri][na] → [Ka][thi]
- b. [A][bi][tur] → [A][bi]
- c. [Stu][dent] → [Stu][di]

Wieder stellt sich das Problem, wie Silben in endlichen Automaten zu realisieren sind. Hinzu kommt, dass sich ein Prozess wie die Herauslösung von Wortteilen (*-arina*, *-itur* und *-ent*) nicht ohne weiteres durch endliche Automaten darstellen lässt.

3.3.5 Default-Vererbungsnetze: DATR

DATR (Evans und Gazdar 1996) gehört zu den sogenannten Vererbungs-Netzen und ist speziell für computerlinguistische Lexikonrepräsentationen entwickelt worden, also insbesondere für morphologische Anwendungen. Viele Probleme,

die sich aus einer naiven Anwendung von Finite-State-Methoden in der Morphologie ergeben, lassen sich in DATR durch dessen reichere Syntax und durch den Gebrauch von Defaultmechanismen elegant umgehen.

Wissensrepräsentation in DATR

Lexikalisches Wissen ist in DATR in Knoten organisiert, wobei Knoten Pfade Werte zuweisen. Einzelne Grammatiken bzw. Programme heißen in DATR Theorien. Der folgende Satz aus einer DATR-Theorie enthält Informationen über die Formen des Verbs *waten*, die sich über Anfragen an ein DATR-System abrufen lassen:

```
WATEN:
<form präs sg eins> == w a t e
<form präs pl zwei> == w a t e t
.
```

WATEN ist der Name eines Knotens, `<form präs sg eins>` ist ein Pfad, der für die erste Person Singular Präsens des Wortes steht und `wate` ist der Wert des Pfades, Entsprechendes gilt für die zweite Zeile. Der Doppelpunkt nach dem Knotennamen, `==` und der Punkt am Ende des Satzes sind Teil der DATR-Syntax, um Sätze, Pfade und Werte von einander abzugrenzen. Auf die Anfrage WATEN: `<form präs sg eins>` wird das System, wie zu erwarten, `w a t e` antworten. Das Ziel ist natürlich nicht, alle möglichen Formen aufzuzählen, sondern Regeln anzugeben, mit denen die Inferenzmaschine von DATR Formen ableiten kann:

```
WATEN:
<form> == <wurzel> <endung>
<wurzel> == w a t
<endung präs sg eins> == e
<endung präs pl zwei> == e t
.
```

Eine Ableitung der verschiedenen Formen wird ermöglicht durch die Option, auf der rechten Seite von Statements auf andere Pfadangaben zu referieren und die entsprechenden Werte zu verketten. `<wurzel>` aus dem unteren Abschnitt ist in diesem Kontext eine Abkürzung für WATEN: `<wurzel>`. Im Defaultfall wird ein Pfad immer dem aktuellen Knoten zugerechnet. Was geschieht nun bei der Anfrage WATEN: `<form präs sg eins>`., die nach einem Pfad fragt, der explizit nirgends definiert ist. Was definiert ist, ist ein Subpfad davon: WATEN: `<form>`. ($\langle A_1 \dots A_n \rangle$ ist ein Subpfad von $\langle A_1 \dots A_n A_{n+1} \dots A_m \rangle$, für $0 \leq n \leq m$, ein Subpfad ist also ein zusammenhängender initialer Teilstamm.) Ist ein Pfad nicht ausdrücklich definiert, hält sich DATR an den längsten definierten Subpfad der Anfrage, in diesem Fall also das erste Statement aus obigem Code. Automatisch wird dabei auf der rechten Seite der fehlende Teilstamm ergänzt, also nach `<endung präs sg eins>` gesucht. Der zweite Pfad ist tatsächlich explizit definiert und zwar mit dem Wert `e`. Für den ersten existiert wieder ein Subpfad,

nämlich <wurzel>, dessen Wert diesmal atomar ist (d.h. keine weiteren Pfadangaben enthält). Also wird direkt w a t übernommen. w a t und e werden zu w a t e verkettet.

Defaults und Neutralisierung

Die Wahl des längsten Subpfades ist einer von zwei grundlegenden Defaultmechanismen in DATR. Damit kann man z.B. die Tatsache ausdrücken, dass -en die Default-Endung für Verbformen ist, und Imperfekt-Endungen (impf) normalerweise identisch zu den Präsens-Endungen sind. Die einzige Ausnahme zu dieser Feststellung ist die 1.sg. Präsens, für die ein expliziter Wert angegeben ist, der aufgrund des ersten Kriteriums (längster Subpfad) wiederum die Default-Spezifikation überschreibt. Dies ist im Übrigen genau das Neutralisierungs-Phänomen, das in Abschnitt 3.3.4 zur Aufblähung des Transducers geführt hat.

WATEN:

<endung sg zwei>	== e s t
<endung sg>	== e
<endung pl zwei>	== e t
<endung>	== e n
<endung präs sg drei>	== e t
<endung präs>	== <endung>
<endung impf>	== <endung>

.

Unbefriedigend an der bisherigen Organisation ist die Tatsache, dass sehr generelle Informationen über die Bildung von Verb-Formen Teil des Knotens für ein spezielles Verb sind. Günstiger wäre es, einen eigenen Knoten festzulegen, der solche generellen Informationen über Verbformen enthält und diese Informationen für WATEN einfach zu übernehmen:

VERB:

<form>	== "<wurzel>" "<endung>"
<endung sg zwei>	== e s t
<endung sg>	== e
<endung pl zwei>	== e t
<endung pl>	== e n
<endung präs sg drei>	== e t
<endung präs>	== "<endung>"
<endung impf>	== "<endung>"

.

WATEN:

<>	== VERB
<wurzel>	== w a t

.

Das erste Statement von WATEN ist eine Abkürzung für `<> == VERB:<>`. Das Klammerpaar `<>` ist dabei der leere Pfad, der Subpfad für alle Pfade (einschließlich seiner selbst). Das heißt, dass WATEN alle Pfad-Werte, die nicht unter WATEN selbst definiert sind, von VERB übernimmt. Der Knoten VERB enthält jetzt genau die Information, die in WATEN fehlt. Einziger Unterschied sind die Pfadangaben auf der rechten Seite, die hier in Anführungszeichen stehen. Damit hat es folgende Bewandtnis: Eine Pfadangabe wie `<wurzel>` bezieht sich in VERB auf den Pfad `VERB:<wurzel>`. Dieser Pfad ist aber gar nicht definiert. Was eigentlich gemeint ist, ist die Wurzel des Verb-Knotens, für den eine Anfrage gestartet wird, also z. B. für WATEN. Die Anführungszeichen bewirken, dass "`<wurzel>`" global ausgewertet wird, sich also für die Anfrage `WATEN:<form präs sg eins>`. auf `WATEN:<wurzel>` bezieht.

Nichtlokale Abhängigkeiten

Das Fragment lässt sich jetzt in naheliegender Weise erweitern, um auch die Imperfekt- und Partizipial-Formen abzudecken:

```

VERB:
<form>                ==  "<stamm>" "<endung>"
<form part>            ==  "g e "<wurzel>" "<endung part>""
<stamm präs>           ==  "<wurzel präs>""
<stamm impf>           ==  "<wurzel impf>" e t
<endung sg zwei>      ==  e s t
<endung sg>             ==  e
<endung pl zwei>      ==  e t
<endung pl>             ==  e n
<endung präs sg drei> ==  e t
<endung präs>           ==  "<endung>""
<endung impf>           ==  "<endung>""
<endung part>          ==  e t
.
.
```

Die Verbwurzel wird mit einem eventuellen Imperfekt-Affix zu `stamm` zusammengefasst. Der Präsensstamm ist identisch zur Wurzel. Während Verbformen generell aus `stamm` und `endung` bestehen, wird dies für das Partizip überschrieben, das die entsprechende dreiteilige Struktur erhält.

Nichtkonkatenative Morphologie: Starke Verben

Um starke Verben wie *raten* ebenfalls zu erfassen, muss die Repräsentation der Verb-Wurzel etwas modifiziert werden:

```

VERBKLASSE1:
<>                  ==  VERB
<wurzel>             ==  "<wurzelanf>" "<wurzelvok>" "<wurzelend>""
<wurzel impf>         ==  "<wurzel impf>""
.
.
```

```
RATEN:
<>          == VERBKLASSE1
<wurzelanf> == r
<wurzelend> == t
.
```

Wurzeln werden in Wurzelvokale (jene Vokale, die sich bei der Ablautung ändern), den Wurzel-Anteil davor und den Wurzelanteil danach eingeteilt, um explizit auf diesen Vokal zugreifen und seine jeweilige Form beschreiben zu können. VERBKLASSE1: muss dann noch durch folgende Statements ergänzt werden, um die korrekte Ablautung zu erhalten:

```
<wurzelvok präs sg zwei> == ä
<wurzelvok präs sg drei> == ä
<wurzelvok impf>          == i e
<wurzelvok>               == a
```

Anders als mit endlichen Automaten ist diese Darstellung relativ minimal und erlaubt, den Vokalwandelprozess explizit darzustellen. Weitere Feststellungen sind natürlich nötig, um die Endungs-Allomorphie darzustellen, z. B. die Tatsache, dass *raten* bei der Bildung des Partizip II *-en* statt *-et* fordert, dass die 3.sg. Präsens und die 1/3.sg. Imperfekt mit dem Nullaffix gebildet werden und die 2.sg. Präsens *-st* statt *-est* aufweist:

```
<endung impf sg eins>    ==
<endung impf sg drei>    ==
<endung präs sg drei>    ==
<endung präs sg zwei>    == s t
<endung part>            == e n
```

Um weitere Verben derselben Flexionsklasse einzufügen, reicht es, einen minimalen weiteren Knoten einzufügen, etwa:

```
SCHLAF:
<>          == VERBKLASSE1
<wurzelanf> == s c h l
<wurzelend> == f
.
```

Prosodische Morphologie: *i*-Wörter

Anhand der *i*-Wörter soll jetzt gezeigt werden, dass es DATR auch erlaubt, morphologische Prozesse, die auf die Silbenstruktur Bezug nehmen, zu modellieren. Um die Diskussion zu vereinfachen, wird in den Formalisierungen von *i*-Wörtern hier und in Abschnitt 3.3.6 nur ein Teilbereich der Daten behandelt, nämlich nur Ausgangsstämme mit einer relativ einheitlichen Silbenstruktur. Als erster Schritt dahin wird die phonologische Repräsentation im Lexikon weiter modifiziert, indem jeder Laut einer Silbenposition zugewiesen wird (**ons** → Onset, **nuk**

→ Nukleus, **kod** → Koda). Nukleus ist der unverzichtbare Kern einer Silbe, in den meisten Fällen ein Vokal. Onset und Koda sind die Konsonanten, die vor bzw. hinter dem Nukleus stehen (siehe auch Unterkapitel 3.1):

```
STUDENT:
<>           == WORT
<silbe ons 1> == s t
<silbe nuk 1> == u
<silbe ons 2> == d
<silbe nuk 2> == e
<silbe kod 2> == n t
.
```

Die Nummern hinter den Silbenpositionen markieren dabei die Position der einzelnen Silben, z. B. ist **<silbe nuk 2>** der Nukleus der zweiten Silbe. Der generische Knoten **WORT** erlaubt es, daraus vollständige Formen für Silben und Wörter abzuleiten. Die letzte Zeile legt fest, dass im Default-Fall alle Silbenpositionen leer sind:

```
WORT:
<form>        == "<silbe form 1>" "<silbe form 2>"
<silbe form>   == "<silbe ons>" "<silbe nuk>" "<silbe kod>"
<silbe>        ==
.
```

Z. B. erhält man für die Anfrage **STUDENT:<silbe form 1>** die Antwort **s t u** und für **STUDENT:<form> s t u d e n t**. Die minimale Spezifikation für eine Bildung wie *Studi* ist nun, dass sie als Ausgangs-Basis *Student* hat – dies wird durch die Zuweisung der Silbenpositionen von **STUDENT** an die entsprechende Stelle nach **<basis>** erreicht – und ein entsprechendes *i*-Wort ist:

```
STUDI:
<>           == IWORT
<basis silbe> == STUDENT:<silbe>
.
```

Der Knoten **IWORT** beschreibt jetzt die generelle Bildungsweise von *i*-Wörtern: Onset und Nukleus der ersten Silbe (**s t u**) + Onset der zweiten Silbe (**d**) + *i*:

```
IWORT:
<>           == NOMEN
<silbe form 1> == "<basis silbe ons 1>" 
                  "<basis silbe nuk 1>" 
<silbe form 2> == "<basis silbe ons 2>" i
.
```

Anwendungsaspekte

Ein Aspekt, der hier nicht angesprochen wurde, ist die Integration von Phono- logie. In DATR lassen sich allerdings auch FSTs problemlos implementieren und integrieren. Dies weist bereits darauf hin, dass DATR im Sinn der Theorie der formalen Sprachen ein sehr mächtiger Formalismus ist (siehe Abschnitt 3.3.7), womit sich auch die Frage stellt, wie effizient DATR-Theorien in der praktischen Anwendung sind.

Tatsächlich sind existierende DATR-Implementationen problematisch, weil sie anders als Finite-State-Transducer nicht grundsätzlich in beide Richtungen arbeiten. Man kann mit DATR-Theorien wie den oben skizzierten zwar `w a r t e` ausgehend von einer Spezifikation wie `WARTEN:<sg eins>` ableiten, aber nicht umgekehrt `w a r t e` den Pfad `WARTEN:<sg eins>` zuweisen. Strategien für solche sogenannten „Reverse Queries“ (siehe Langer 1996) existieren bis jetzt nur für eine Teilmenge von DATR und mit Vorbehalten für deren praktische Nutzung. Für Generierung wie für Analyse sind Finite-State-Modelle bei weitem effizienter als jede existierende DATR-Implementation.

Dennoch setzt DATR für die kompakte Modellierung morphologischer Phäno- mene einen hilfreichen Standard, an dem sich andere Ansätze, wie die erweiterten Finite-State-Ansätze, die im nächsten Abschnitt vorgestellt werden, messen lassen.

3.3.6 Erweiterte Finite-State-Ansätze

Bei der Diskussion von endlichen Automaten (Abschnitt 3.3.4) sind Probleme v.a. bei der Modellierung von Neutralisierung und nichtkonkatenativer Mor- phologie deutlich geworden. In diesem Abschnitt werden Möglichkeiten bespro- chen, die Effizienz von endlichen Automaten zu nutzen, ohne diese Nachteile mit zu übernehmen. Zentral ist dabei die Möglichkeit, endliche Automaten durch Schnitt und Komposition zu neuen Automaten zu kombinieren. Dadurch können verschiedene Aspekte morphologischer Struktur in kompakte Automa- ten/Transducer kodiert werden, während die Kombination dieser Bestandteile durch Standardoperationen der Automatentheorie gewährleistet ist. Die folgen- den Ausführungen orientieren sich lose an den Arbeiten von Ken Beesley zum Arabischen (siehe Beesley 1998 und die Literaturangaben dort).

Nichtlokale Abhängigkeiten

Wie in Abschnitt 3.3.4 gezeigt worden ist, lässt sich die wechselseitige Abhän- gigkeit von *ge-* und *-et* nur schwer in einem endlichen Automaten repräsentie- ren. Z. B. akzeptiert der Transducer in Abbildung 3.32 neben *ge-wat-et* auch die falsche Form **wat-et* (o.k. als 3.sg. aber falsch als Partizip):

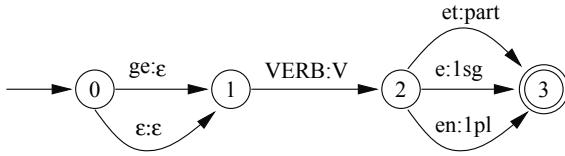


Abbildung 3.32: FST ohne Beschränkungen über die Kookurrenz von *ge-* ohne *-et*

VERB:V steht hier abkürzend für alle Übergänge, die Verben auf V abbilden. Wie in Abschnitt 3.3.4 angesprochen wurde, sind die Bedingungen für das Auftreten von *ge-* und *-et* relativ komplex, aber in der folgenden Modellierung wird vereinfachend davon ausgegangen, dass beide Affixe immer zusammen auftauchen. Der wichtigste Schritt, um **ge-wat* und **wat-et* auszuschließen, ist die Bedingung explizit zu machen, dass *ge-* und *-et* nur gemeinsam vorkommen. Das leistet der Transducer in Abbildung 3.33.

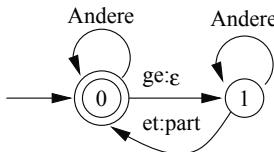


Abbildung 3.33: Kookurrenz-Beschränkung über *ge-* und *-et* als FST

Der Ausdruck *Andere* bezeichnet dabei alle Übergänge, die Morphemen außer *ge-* und Partizipial-*-et* entsprechen. Auf Grund der Übergänge von Zustand 0 zu sich selbst akzeptiert der Transducer alle Abbildungen, die weder *ge-* noch *-et* enthalten. *ge-* und *-et* können andererseits nur paarweise auftreten, da der einzige Übergang mit *-et* von Zustand 1 ausgeht, der wiederum nur über *ge-* zu erreichen ist, und der einzige Übergang mit *ge-* nach Zustand 1 führt, von dem aus ein Endzustand nur über *-et* erreichbar ist.

Für sich genommen akzeptieren die beiden Transducer inkorrekte Abbildungen, da der Transducer in Abbildung 3.33 nur die Beziehung von *ge-* und *-et* festlegt und der Transducer in Abbildung 3.32 alle anderen relevanten Beschränkungen. Hingegen sind die Abbildungen, die von beiden akzeptiert werden, wohlgeformt.

Eliminiert man die ϵ -Übergänge in geeigneter Weise – denn die Schnittbildung von Transducern mit solchen Übergängen führt u. U. nicht zu Finite-State-Transducern (siehe Unterkapitel 3.1) –, so kann die Kombination der beiden Transducer auf zwei Arten erfolgen: Zum einen können beide Transducer parallel verwendet werden. Z. B. lässt sich die Zulässigkeit der Abbildung $ge \text{ } bet \text{ } et \rightarrow \epsilon \text{ } V \text{ } part$ dadurch prüfen, dass beide Transducer gleichzeitig durchlaufen werden, wobei ein Zustandsübergang für ein Symbolpaar (wie *et:part*) nur dann möglich ist, wenn in beiden Transducern ein entsprechender Übergang vorhanden ist. Die

zweite Möglichkeit besteht darin, die Transducer zu schneiden (siehe Unterkapitel 2.2). Man erhält dann einen Gesamt-Transducer, der genau die Abbildungen akzeptiert, die von beiden Teil-Transducern akzeptiert werden.

Der Gesamttransducer wird in diesem Fall aus den in Abschnitt 3.3.4 genannten Gründen wie im Automaten aus Abbildung 3.31 zu einer Verdoppelung von Übergängen führen, aber durch die Aufspaltung ist die Repräsentation plausibler und praktisch handhabbarer geworden.

Nichtkonkatenative Morphologie: Starke Verben

Ähnlich wie mit nichtlokalen Abhängigkeiten kann man mit nichtkonkatenativen Prozessen wie dem Ablaut umgehen. Die Grundidee ist, den variablen Vokal in ablautenden Wurzeln wie *schlafen* und *raten* nicht näher zu spezifizieren, d.h. durch eine Disjunktion aller möglichen Vokale darzustellen (mittels „|“ angegeben), hier in Form eines regulären Ausdrucks:

$$(3.12) \quad r (a | e | i | o | u | ä | ie) t$$

$$(3.13) \quad s c h l (a | e | i | o | u | ä | ie) f$$

Ebenfalls durch reguläre Ausdrücke darstellen lassen sich die Vokalmuster für bestimmte Verbklassen. So hat die Klasse von *schlafen* und *raten* in der 1.sg. Präsens ein -*a*, in der 2.sg. Präsens ein -*ae* und in allen Präteritumformen -*ie*.

$$(3.14) \quad X^* a K^* \text{ (Präsens 1.sg.)}$$

$$(3.15) \quad X^* ä K^* \text{ (Präsens 3.sg.)}$$

$$(3.16) \quad X^* ie K^* \text{ (Imperfekt)}$$

K ist hier eine Abkürzung für die Disjunktion aller Konsonanten (*s | f | l | c | h | r | t | n*) und *X* für alle Symbole des Alphabets. (3.14) beschreibt also eine Kette, in der einem *a* beliebig viele Konsonanten folgen und beliebig viele Segmente (Vokale oder Konsonanten) vorausgehen, in anderen Worten eine Form, in der der letzte Vokal ein *a* ist.

Schneidet man (3.12) und (3.14) (bzw. die Automaten, die diesen regulären Ausdrücken entsprechen), so erhält man die Form der Wurzel von *schlafen* in der 1. Pers. Präsens: *schlaf*. Natürlich müssten diese Automaten wiederum in größere Automaten eingebettet werden, die auch die entsprechenden Endungen berücksichtigen, was hier aus Platzgründen unterbleibt.

Noch näher an linguistischen Beschreibungen sind solche Automaten, wenn Automaten statt atomarer Symbole Merkmalsstrukturen verwenden. (3.16) kann man dann durch (3.17) ersetzen (vgl. hierzu auch Unterkapitel 3.1):

$$(3.17) \quad [\quad]^* \left[\begin{array}{c} -\text{konsontant} \\ +\text{hoch} \\ +\text{front} \\ -\text{rund} \end{array} \right] [\begin{array}{c} +\text{konsontant} \end{array}]^*$$

K entspricht der Merkmalsstruktur [+konsonant] und X der leeren Merkmalsstruktur. Der Gebrauch von phonologischen Merkmalsstrukturen macht somit Disjunktionen in vielen Fällen überflüssig. Er spielt auch bei der Umsetzung von prosodischer Phonologie im nächsten Abschnitt eine entscheidende Rolle.

Prosodische Morphologie: *i*-Wörter

Die bisher vorgestellten Methoden reichen noch nicht aus für morphologische Bildungen, bei denen existierendes Material gelöscht oder kopiert wird, wie bei den schon behandelten *i*-Wörtern. Die folgende Analyse ist von Walther 1999 inspiriert, weicht aber in essentiellen Einzelheiten von seiner Behandlung derselben Daten ab.

Ein erster Schritt zur Darstellung von prosodischer Morphologie durch endliche Automaten ist die Interpretation von Silbenpositionen (vgl. Abschnitt 3.3.5) durch segmentale Merkmale: [+O(nset)], [+K(oda)] und [-O(nset) -K(oda)] (Silbennukleus). Die hierarchische Struktur in Abbildung 3.34 wird also zu (3.18):

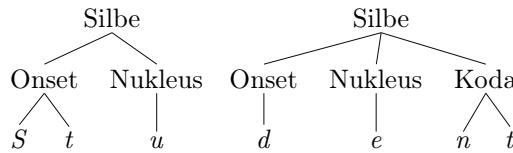


Abbildung 3.34: Silbenstruktur von *Student*

$$(3.18) \quad \begin{matrix} [+O] & [+O] & [-O-K] & [+O] & [-O-K] & [+K] & [+K] \\ s & t & u & d & e & n & t \end{matrix}$$

Die zweite Grundidee ist, endliche Automaten durch diakritische Übergänge zu erweitern, die die Möglichkeit partieller Realisierung kodieren. Diese erhalten das ausgezeichnete Label *skip* und erlauben, unter bestimmten Umständen einzelne Segmente in einem Automaten wegzulassen (Abbildung 3.35).

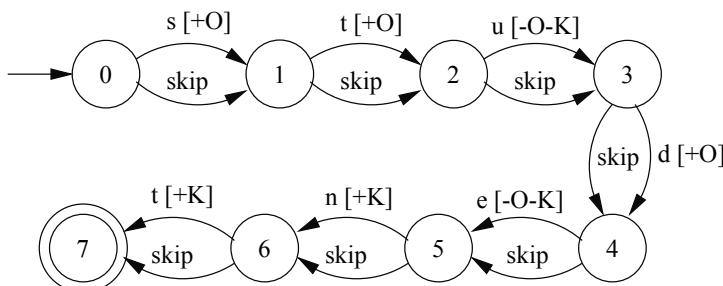


Abbildung 3.35: *Student* als endlicher Automat mit *skip*-Übergängen

Anders als ϵ -Übergänge sind sie nur dann relevant, wenn der Automat mit einem anderen Automaten geschnitten wird. Die Verkürzung bei der Bildung von *i*-Wörtern lässt sich jetzt durch den Automaten in Abbildung 3.36 darstellen.

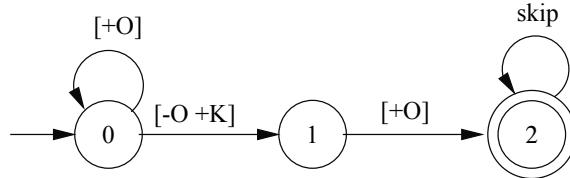


Abbildung 3.36: Die Stammverkürzung bei *i*-Wörtern als endlicher Automat

Dieser Automat akzeptiert nur Ketten mit einer Silbe gefolgt von einem einfachen Onset einer zweiten Silbe, dem beliebig viele *skip*-Symbole folgen können. Der Schnitt der beiden Automaten ergibt:

$$(3.19) \quad \begin{matrix} [+O] & [+O] & [-O - K] & [+O] & \text{skip} & \text{skip} & \text{skip} \\ s & t & u & d & & & \end{matrix}$$

Hier kommen jetzt die *skip*-Übergänge zum Tragen. Aufgrund der Struktur von Abbildung 3.36 werden nur die Anfangssemente von Abbildung 3.35 übernommen. Die drei Vorkommen von *skip* entsprechen der Folge $e /[-O -K] n /+K/ t /+K/$ in Abbildung 3.35, also den weggelassenen Segmenten. Um die korrekte Form zu erhalten, müssen jetzt nur noch die verbliebenen *skip*-Symbole (bzw. ϵ -Übergänge) entfernt und $-i$ suffigiert werden.

Defaults und Neutralisierung

Die Formalisierung von Defaults in Trommer (1999) beruht im Wesentlichen auf der Idee, dass Allomorphie durch die Anwendung von geordneten Ersetzungsregeln auf Ketten abstrakter Morpheme dargestellt werden kann. Als Beispiel soll noch einmal die Neutralisierung des Konstrasts zwischen 1. und 3. Person sg. im Imperfekt dienen (vgl. Abschnitt 3.3.4). Auf der Bedeutungsseite sollen folgende Repräsentationen für diese Formen angenommen werden:

$$(3.20) \quad \text{V 3sg } (er \text{ wat-}et)$$

$$(3.21) \quad \text{V impf 3sg } (er \text{ wat-}et-e)$$

Die nachstehende Regel-Folge leitet die korrekten Formen für die Person-/Numerus-Endungen ab:

$$(3.22) \quad 3\text{sg} \rightarrow et / V \underline{\quad}$$

$$(3.23) \quad (1\text{sg} | 3\text{sg}) \rightarrow e$$

Das Format und die Interpretation der Regeln ist wie bei phonologischen Ersetzungsregeln. Regel (3.22) muss wie bei phonologischen Regeln immer vor (3.23) angewendet werden. Da in der Präsens-Form *3.sg* nach V durch *-et* ersetzt (also gelöscht) wird, bevor (3.23) angewendet werden kann, entfällt in diesem Fall die Anwendung von (3.23). Dies ergibt die erforderliche Blockierung von *-et*. Für die Formen der 1.sg. und die 3.sg. Imperfekt ist aufgrund der Kontextbeschränkung von (3.22) nur die Default-Regel anwendbar.

Regelhierarchien dieser Art können ihrerseits wie in der computerlinguistischen Phonologie als Finite-State-Transducer implementiert werden. Dabei ist die Komposition von FSTs zu neuen FSTs von entscheidender Bedeutung (vgl. Unterkapitel 3.1).

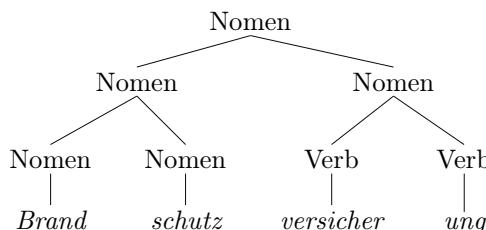
Anwendungsaspekte

Die erweiterten Finite-State-Methoden, die in diesem Abschnitt eingeführt wurden, beruhen vor allem auf Schnitt und Komposition von endlichen Automaten. Obwohl diese Operationen im Sinn der Komplexitätstheorie nicht trivial und entschieden komplexer sind als etwa die Erkennung von Formen mit endlichen Automaten, sind in den letzten Jahrzehnten dafür effiziente Verfahren entwickelt worden.

3.3.7 Morphologie und generative Kapazität

Die Formalismen, die in diesem Unterkapitel diskutiert wurden, sind von sehr unterschiedlicher Mächtigkeit. Ausgehend von der Chomsky-Hierarchie (siehe Unterkapitel 2.2) sind endliche Automaten aufgrund der Äquivalenz zu Typ-3-Grammatiken minimal komplex. DATR hingegen erlaubt die Simulation einer Turing-Maschine, ist also mindestens so mächtig wie die Klasse der Typ-0-Grammatiken (Moser 1992). Vor diesem Hintergrund stellt sich die Frage, wie mächtig Grammatikformalismen grundsätzlich sein müssen, um morphologische Phänomene adäquat formalisieren zu können.

Auf der Ebene der starken generativen Kapazität stellt sich die Frage, welche Strukturen morphologische Repräsentationen benötigen. Die meisten Linguisten gehen davon aus, dass zumindest für eine angemessene Repräsentation von Wortbildung Konstituentenstrukturen wie in der Syntax (siehe Unterkapitel 3.5) nötig sind, z. B. für *Brandschutzversicherung*:



Durch endliche Automaten erhält man keine entsprechenden Strukturen. Eine angemessene Struktur kann aber auch nicht durch Typ-3-Grammatiken erzeugt

werden, die zu endlichen Automaten schwach äquivalent sind, da der Baum weder links- noch rechtslinear ist. Dazu sind mindestens Typ-2-Grammatiken nötig. Auf der Ebene der schwachen generativen Kapazität zeigt ein klassisches Beispiel von Bar-Hillel und Shamir 1960, dass Wortbildungsmorphologie nicht regulär sein kann. Die Autoren gehen von dem englischen Wort für Rakete aus: *missile*. Gegen ein *missile* lässt sich ein *anti missile missile* konstruieren. Als gegnerische Antwort auf dieses ist ein *anti anti missile missile missile* zu erwarten, dem nur durch ein *anti anti anti missile missile missile missile* beizukommen ist. Während diese Konstruktionen schnell unverständlich werden, lässt sich argumentieren, dass sie, egal wie lange man das Spiel weitertreibt, grundsätzlich grammatisch bleiben, im Gegensatz zu Formen wie **anti missile* oder **anti missile missile missile*. Die Menge der wohlgeformten Ableitungen entspricht der Sprache $\{anti^n missile^{n+1} | n \in \mathbb{N}\}$, die, wie sich zeigen lässt, nicht regulär ist, aber sich durch die folgende kontextfreie Grammatik aufzählen lässt:

$$\begin{array}{l} N \rightarrow missile \\ N \rightarrow anti N missile \end{array}$$

Obwohl diese Beispiele zeigen, dass es Fälle gibt, in denen endliche Automaten oder Typ-3-Grammatiken nicht hinreichen, handelt es sich hier um Grenzfälle, die in der Praxis keine große Rolle zu spielen scheinen. Ein Grund dafür ist wohl, dass Rekursion in der Morphologie nur sehr eingeschränkt möglich ist.

3.3.8 Zusammenfassung und Ausblick

In diesem Unterkapitel wurden grundlegende Probleme vorgestellt, die sich bei der Formalisierung morphologischer Phänomene ergeben. Schwerpunkte waren dabei die Integration von Morphologie und Phonologie, Neutralisierung und nichtkonkatenative Morphologie. Diese Problemfelder sind auch für andere Ansätze, z. B. aus dem Umfeld der Merkmal-Wert-Formalismen relevant, die hier nicht behandelt wurden. Eine Fülle von weiteren Problemen, etwa im Bereich der Wortsemantik und der Produktivität morphologischer Bildungen, ergeben sich insbesondere in der Wortbildungsmorphologie.

3.3.9 Literaturhinweise

Begriffe und Modelle aus der linguistischen Morphologie werden ausführlich in Spencer (1991) eingeführt. Einen guten Überblick über die computerlinguistische Morphologie bietet Sproat (1992). Er geht insbesondere auf Finite-State-Modelle ein und bietet einen Überblick über viele weitere Ansätze, die hier nicht besprochen werden konnten. Nicht vertreten ist DATR. Die beste Einführung dafür ist immer noch Evans und Gazdar (1996).

3.4 Flache Satzverarbeitung

André Hagenbruch

Elektronisch verfügbare Textkorpora bilden das Ausgangsmaterial verschiedenartigster linguistischer Analysen. Damit diese Dokumente mit informationstechnologischen Werkzeugen bearbeitet werden können, müssen sie derart aufbereitet werden, dass zunächst aus dem Strom der Zeichen sprachlich relevante Einheiten, wie z. B. Wörter, Phrasen oder Sätze, extrahiert werden. Einer solchen Segmentierung schließt sich die Annotation des linguistischen Materials an: Wörter werden gemäß ihrer Wortarten von einem **Tagger** ausgezeichnet und anschließend von einem **Chunk-Parser** zu phrasalen Strukturen zusammengefasst. Derart bearbeitete Dokumente stellen eine wertvolle Ressource sowohl für höhergeordnete Anwendungen wie der Lexikographie (s. Unterkapitel 5.2) als auch für grundlegende Analysen wie der Kollokationsextraktion oder dem Syntaxparsing (s. Unterkapitel 3.5) dar.

Der Schwerpunkt dieses Unterkapitels soll auf der Vorstellung der jeweiligen Problemstellungen liegen, der man sich bei der Implementierung oben genannter Systeme gegenüber sieht. Lösungsansätze für die verschiedenen Vorverarbeitungsschritte werden anhand einschlägiger Implementationen skizziert und mögen dem Leser als Richtschnur dienen.

3.4.1 Tokenisierung

Problemstellung

Vor der eigentlichen Analyse und Verarbeitung in elektronischer Form vorliegender Texte segmentiert man Dokumente in linguistische Einheiten, wie z. B. Wörter, Phrasen, Sätze, Absätze oder Diskursabschnitte. Die Frage, was ein Wort ist, erscheint trivial: Ein Wort ist eine Einheit aus alphanumerischen Zeichen, die zu ihrer Rechten und Linken durch Leerraumzeichen (engl. *white space*) oder Interpunktionszeichen begrenzt wird. Ein solches Segmentierungsverfahren, das jedes Wort eines Textes erfasst, nennt man **Tokenisierung**.

Das oben eingeführte Kriterium zur Definition eines Tokens trifft für alle **segmentierten Schriftsysteme** zu, wie z. B. den aus lateinischen, kyrillischen oder griechischen Zeichen bestehenden. In **nicht-segmentierten Schriftsystemen**, wie sie beispielsweise im Chinesischen oder Japanischen verwendet werden, fehlen Leerraumzeichen und Interpunktionszeichen gänzlich. Die dort verwendeten Piktogramme werden in keiner Form explizit voneinander abgegrenzt, sondern adjazent aneinander geschrieben. Die Problematik besteht in diesen Sprachen nun darin, dass fast alle Piktogramme einerseits sowohl aus einem Zeichen bestehende Wörter darstellen können, andererseits aber auch im Zusammenhang mit anderen Zeichen ein neues Wort ergeben (vgl. Mikheev 2003).

Doch auch segmentierte Schriftsysteme sind in Bezug auf die Tokenisierung nicht unproblematisch: Während Buchstaben, Ziffern und Satzzeichen bedeutungstragende Elemente sind, zählen Leerzeichen, Tabulatoreinschübe und Zei-

lenumbrüche zu denjenigen Zeichen, die rein typographische Funktionen erfüllen; Ambiguitäten ergeben sich daher vor allem durch Satzzeichen. Ausrufe- und Fragezeichen werden zwar selten in Markennamen wie *Joop!* oder *Guess?* gebraucht, markieren aber für gewöhnlich genau so wie das Semikolon und der Doppelpunkt ein Satzende. Der Punkt wird allerdings nicht nur in dieser Funktion verwendet, sondern auch, um Abkürzungen oder Datumsangaben zu kennzeichnen oder als Trennzeichen in Zahlen.

Betrachten wir zunächst den zuletzt genannten Fall. Hier erschweren gerade sprachspezifische Unterschiede die Segmentierung: Während man den Punkt im englischsprachigen Raum zur Abtrennung von Dezimalstellen verwendet, geschieht dies im Deutschen durch ein Komma. Umgekehrt verwenden Briten und Amerikaner das Komma zur Gliederung von Zahlen mit mehr als drei Ziffern, während im deutschsprachigen Raum dazu der Punkt benutzt wird. Im Französischen hingegen kommt zwar ein Dezimalkomma zum Einsatz, man unterteilt große Zahlen allerdings durch Leerzeichen. Dementsprechend repräsentieren *12,345,678.99*, *12.345.678,99* und *12 345 678,99* die selbe Zahl.

Ein dem englischen Zahlenformat ähnliches Verhalten zeigen numerische Datumsangaben. Dort stehen für gewöhnlich ein- oder zweistellige Zahlen für den Tag und den Monat, während Jahreszahlen zwei- oder vierstellig repräsentiert werden; zwischen diesen Einheiten steht jeweils ein Punkt, z. B. *30.4.03* oder *30.04.2003*. Aufgabe des Tokenizers ist es, aus all diesen Ziffern- und Interpunktionsfolgen jeweils ein Token zu machen.

Die mit Abstand signifikanteste Ambiguität, die ein Punkt verursachen kann, manifestiert sich allerdings in der Frage, ob der Punkt Teil einer Abkürzung ist, oder ob er ein Satzende markiert. Obwohl der Punkt in beiden Fällen rechtsadjazent zu der betrachteten Zeichenkette auftritt, ist er nur im ersten Fall integraler Bestandteil des Wortes und sollte somit auch zusammen mit der Zeichenkette tokenisiert werden, während er als Satzendezeichen von dieser abgetrennt als einzelnes Token betrachtet werden muss.

Zunächst sollte man zwischen **Abkürzungen** (z. B., *vgl.*, *usw.*, *Dr.*, *Abt.*) und **Akryonymen** (*JArbSchG*, *Kfz*, *ADAC*, *FCKW*) unterscheiden. Während in ersten nach jedem zu verkürzenden Glied ein Punkt steht, bestehen Akryonyme aus adjazent zueinander gesetzten Anfangsbuchstaben der zu verkürzenden Wörter (weshalb sie auch **Initialworte** genannt werden). Dementsprechend können sie wie ganz normale Wörter behandelt werden. Ebenfalls keine Abkürzungen stellen Wörter oder Phrasen dar, bei denen sich ausschließlich wortinterne Punkte wie z. B. in *Dot.Com-Blase*, *Ruhr.2010* oder *Web 2.0* finden.

Welche Kriterien können wir nun angeben, um einen Satz von einer Abkürzung zu unterscheiden? Versuchen wir zunächst zu erfassen, was ein Satz ist: Das erste Wort eines Satzes beginnt mit einem Großbuchstaben, danach folgen weitere durch Leerzeichen oder Satzzeichen begrenzte groß- oder kleingeschriebene Wörter; am Ende des Satzes findet sich ein Satzendezeichen. Das Kriterium der initialen Großschreibung trifft aber auch auf Abkürzungen wie z. B. zu. Ähnliches gilt für *vgl.*, nach dem häufig ein großgeschriebenes Wort oder eine Abkürzung steht, die mit einem Großbuchstaben beginnt. Es erscheint dementsprechend ergebniger, eine Definition für Abkürzungen zu finden, die deren Schriftstruktur

berücksichtigt: Grundsätzlich kann jede aus wenigen Zeichen bestehende Zeichenkette eine Abkürzung darstellen, wenn sie durch einen Punkt abgeschlossen wird. Findet man diese Einheit darüber hinaus nicht oder nur sehr selten in anderen Kontexten als eigenständiges Wort der betrachteten Sprache ohne Punkt, kann man mit hoher Zuverlässigkeit davon ausgehen, dass es sich um eine Abkürzung handelt. Man muss also abhängig von der Vorkommenshäufigkeit entscheiden, ob *Abt* ein Wort oder eine Abkürzung darstellt. Als weiteres (wenn auch nicht so starkes) Kriterium dient die Beobachtung, dass Abkürzungen oftmals keine Vokale enthalten.

Dennoch gilt es, einige weitere Fälle im Auge zu behalten, die problematisch sind: Abkürzungen wie *usw.* treten häufig am Satzende auf, weshalb in einem solchen Fall der Abkürzungspunkt zugleich der Schlusspunkt des Satzes ist. Eine Abkürzung wie *Dr.* kann z. B. im Englischen die unterschiedlichen Bedeutungen *Doctor* und *Drive* besitzen (vgl. Mikheev 2003):

(3.24) He stopped to see Dr. White.

(3.25) He stopped at Meadows Dr. White Falcon was still open.

Neben dem Punkt können auch Interpunktionszeichen wie Bindestriche oder Apostrophe bei der Tokenisierung Probleme bereiten. In beiden Fällen stellt sich die Frage, ob es sich bei der betrachteten Einheit um ein oder mehrere Wörter handelt. So wird man den Bindestrich in *Blaue Reiter-Ausstellung* nicht als integralen Bestandteil des Wortsegments betrachten, während er es in einer Zeichenkette wie *Hochschul-Strukturkommission* sicherlich ist. Nähme man die umgekehrte Sichtweise ein und würde *Reiter-Ausstellung* als ein Token betrachten, während *Hochschul* und *Strukturkommission* als zwei Token zählen würden, wird dies in nachgeordneten Analyseschritten zu Fehlern führen. Eine weitere Fehlerquelle stellt der Bindestrich als Trennzeichen eines Wortes am Zeilenende dar. Da er hier rein typographische Funktion besitzt, sollte er einfach entfernt werden. Tritt allerdings der Fall ein, dass ein Trennstrich gleichzeitig ein Bindestrich ist, kann eine solche Tilgungsoperation zu falschen Ergebnissen führen, sodass z. B. aus *Streik-Ende Streikende* wird.

Der Apostroph ist im Deutschen nicht so häufig anzutreffen wie im Englischen oder Französischen, in denen er Bestandteil von **Klitika** ist. Da es selbst innerhalb einer Sprache keine festen Regeln für die Abtrennung gibt, ist eine richtige Tokenisierung oftmals problematisch: Während im Englischen *they are* zu *they're* zusammengezogen wird und man die Tokenisierungsregel „trenne am Apostroph“ formulieren könnte, ergibt sich bei der Kontraktion von Negationen, wie z. B. *does not* zu *doesn't*, die Schwierigkeit, dass diese Regel zur falschen Tokenisierung der Elemente führen würde. Im Fall des Deutschen breitet sich die Schreibung des Genitiv-s mit Apostroph genauso aus wie die Abtrennung des Plural-s: *Harry's Hardwarehölle: Tiefpreise für PC's*. Hier muss ein System erkennen, dass es sich tatsächlich um jeweils ein Token handelt.

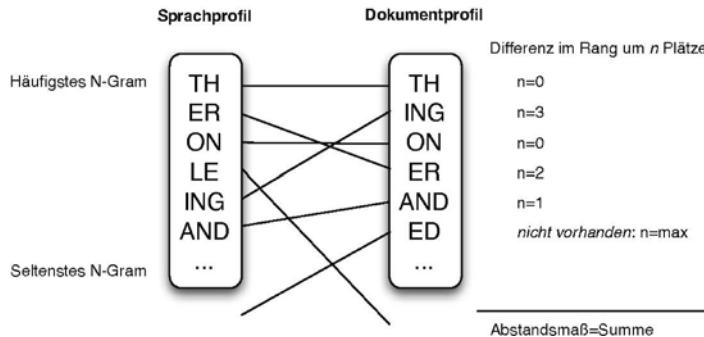
Verfahren

Ansätze zur Tokenisierung lassen sich in symbolische und statistische Verfahren unterteilen. Erstere beruhen auf Heuristiken, die man darüber aufstellt, wie ein Wort, eine Abkürzung und ein Satz der Zielsprache aussehen. Darüber lassen sich reguläre Ausdrücke (s. Unterkapitel 2.2) konstruieren, die man zu einem Regelwerk zusammenfasst, anhand dessen man das betrachtete Korpus vorverarbeitet. Zusätzlich werden häufig Abkürzungslisten manuell erstellt oder durch einfache statistische Verfahren aus einem Korpus generiert. Die Vor- und Nachteile einer solchen Vorgehensweise liegen auf der Hand: Zwar lassen sich schnell und auf einfache Art manuell Regeln erstellen, doch erfordert ein wirklich robustes System einen hohen Arbeitsaufwand. Darüber hinaus ist ein auf diese Weise erstellter Tokenizer meist auf ein bestimmtes Korpus und somit auf eine Domäne und eine Sprache zugeschnitten, lässt sich also nicht ohne Weiteres auf andere Dokumente portieren, zumal Abkürzungen oftmals nicht geschlossenen Klassen angehören und dementsprechend niemals vollständig in einem Lexikon erfasst werden können. Demgegenüber stehen statistische Anwendungen: Sie sind zwar meist schwieriger zu implementieren und in der Entwicklung zeitaufwändiger, lassen sich aber einfacher an neue Korpora, andere Domänen und Sprachen anpassen.

Ein wichtiges Kriterium, welches der Qualitätsbeurteilung dieser Verfahrensarten zugrunde liegt, ist das der **Baseline**: Damit bezeichnet man denjenigen Grad an Genauigkeit, der erreicht wird, wenn man den einfachsten möglichen Algorithmus auf das Problem anwendet. Palmer und Hearst (1997) geben abhängig von der jeweiligen Anwendung und dem Korpus eine Untergrenze an, die ungefähr zwischen 50 und 90% richtig erkannter Interpunktionszeichen liegt, wenn man beispielsweise alle potentiellen Satzendezeichen einschließlich des Punkts, die von einem Leerzeichen und einem Großbuchstaben gefolgt werden, als tatsächliche Satzendezeichen markiert. Mikheev (2003) berichtet Ähnliches für Bindestriche, die am Zeilenende stehen: Verbindet man alle durch Zeilenende-Bindestriche getrennten Wortsegmente miteinander, kommt man mit dieser einfachen Strategie auf eine Fehlrate von ungefähr 5%. An dieser Untergrenze müssen sich weitergehende Verfahren messen lassen.

Exkurs: Automatisierte Sprachenerkennung Bevor wir nun symbolische Verfahren zur Tokenisierung und Satzendeerkennung betrachten, widmen wir uns zunächst einem Ansatz zur automatischen Erkennung von Sprachen, damit ein System unbeaufsichtigt dasjenige Abkürzungslexikon und die für die jeweilige Sprache adäquaten Regeln auswählen kann. Eines der prominentesten, effizientesten und am einfachsten zu implementierenden Verfahren stellt Cavnar und Trenkle (1994) dar. In ihrem Ansatz werden im ersten Schritt Trainingskorpora der zu betrachtenden Sprachen aus Uni- bis Pentagrammen (s. Unterkapitel 2.4) auf Graphemebene in einem Hash gezählt (vgl. Unterkapitel 3.9); Wortanfänge und -enden werden durch einen Unterstrich gekennzeichnet. Danach wird dieser Hash gemäß der absteigenden Häufigkeiten sortiert und die N-Gramme dann in eine Liste geschrieben. Ebenso geht man dann für ein zu klassifizierendes Doku-

ment vor, sodass man letztendlich eine Menge an Sprachprofilen hat, gegen die man das Dokumentprofil vergleichen kann. Dabei wird nun ermittelt an welcher Position sich ein N-Gramm jeweils im Dokumentprofil und in den Sprachenprofilen befindet. Summiert man die Differenz der unterschiedlichen Positionen auf, ergibt sich ein Abstandsmaß, anhand dessen sich ein Ranking der wahrscheinlichsten Sprachen für das gegebene Dokument erzeugen lässt.



Der große Vorteil dieses Verfahrens besteht darin, dass sowohl für die Sprachenprofile als auch für die zu testenden Dokumente geringe Datenmengen ausreichen, um zu einer sicheren Klassifikation zu gelangen; in ihrem Aufsatz erzielen Cavnar und Trenkle mit 400 Trainings-N-Grammen eine durchschnittliche Genauigkeit von 99,8%. Bedenken sollte man allerdings, dass es keinerlei Hilfestellungen bezüglich der Erkennung der Zeichenkodierung eines elektronischen Dokuments oder der Bereinigung von nicht-sprachlichen Zeichen, wie z. B. unterschiedlichster Leerraum oder HTML-Tags oder -Entities bei Web-Dokumenten leistet. Normalisierungsschritte zur Aufbereitung eines Korpus werden in Unterkapitel 4.1 näher beschrieben.

Symbolische Verfahren Reguläre Ausdrücke bilden den Kern von Tokenizern wie sie Anfang der 90er Jahre z. B. von Grefenstette und Tapanainen (1994) entwickelt wurden und auch heute noch häufig eingesetzt werden. Im Unterschied zur Notation regulärer Ausdrücke in Unterkapitel 2.2 verwenden wir hier die in Anwendungen wie Perl, grep oder awk gebräuchliche Schreibweise (s. Unterkapitel 3.9): Zwischen zwei Schrägstriche, dem sogenannten **Vergleichsoperator**, schreiben wir ein **Muster**, das auf eine oder mehrere (Teil-)Zeichenketten im zu betrachtenden Korpus passen soll. Solch ein Muster kann aus literalem Text oder aus einer Abstraktion über Zeichenketten, die man **Zeichenklasse** nennt, bestehen. In einer Zeichenklasse, die durch eckige Klammern notiert wird, stehen einzelne Zeichen, die durch Alternation miteinander verknüpft sind. So bezeichnet z. B. `[ptk]` die Menge der stimmlosen Plosive des Deutschen. Gebräuchliche Zeichenklassen, wie z. B. die der Ziffern, werden durch einen umgekehrten Schrägstrich `\` und einen Buchstaben abgekürzt. So bezeichnet bspw. `\d` alle Ziffern, die Schreibweise mit einem Großbuchstaben ihr Komplement. Eine besondere Bedeutung kommt dem Punkt zu: Er steht in einem regulären Ausdruck für alle Zeichen außer dem Zeilenendezeichen. Will man ihn dennoch als Literal

in einem Ausdruck verwenden, muss man ihn durch einen umgekehrten Schrägstrich maskieren. Dies gilt ebenso für alle anderen Zeichen, denen besondere Funktionen zukommen.

Um die Anzahl der zu findenden Zeichen in einer Zeichenkette anzugeben, verwendet man **Quantoren**: Sucht man null oder mehrere Vorkommen, verwendet man den Kleene-Stern $*$, bei mindestens einem oder mehreren Vorkommen das Kleene-Plus $^+$ und bei Optionalität eines Elements ein Fragezeichen $?$. Darüber hinaus lassen sich in geschweiften Klammern auch einzelne numerische Angaben oder Bereiche für die Häufigkeit eines Elements spezifizieren.

Ein weiteres Element regulärer Ausdrücke stellen die sogenannten Anker dar. Mit ihrer Hilfe lässt sich sicherstellen, dass eine Zeichenkette nur am Anfang oder am Ende einer Zeile gefunden wird: Im ersten Fall verwendet man dazu das Karet \wedge , für das Zeilenende das Dollarzeichen $\$$. Zusätzlich kann man mittels des Ankers $\backslash b$ jenen Punkt bezeichnen, an dem eine Grenze zwischen einem Wortzeichen und einem Nicht-Wortzeichen verläuft.

Zur Unterscheidung zwischen Satzgrenzen und Abkürzungen erfasst beispielsweise der folgende Ausdruck Sequenzen, die aus Buchstabe-Punkt-Buchstabe/Ziffer-Punkt bestehen: $/[A-ZÄÖÜa-zAöü] \cdot [A-ZÄÖÜa-zAöü] 0-9]^+ \cdot /$. In Kombination mit weiteren einfachen Ausdrücken erkennt dieses Verfahren 97,7% der Satzgrenzen richtig. Dieses Ergebnis lässt sich marginal durch den Einsatz eines **Korpusfilters** verbessern, der aufgrund der Vorkommenshäufigkeit eines Tokens im Korpus mit und ohne Punkt entscheidet, ob es sich dabei um eine Abkürzung oder ein Satzende handelt. Substantielle Verbesserungen lassen sich durch den Einsatz eines Vollform-Lexikons in Kombination mit kaskadierenden Heuristiken zur Erkennung von Abkürzungen erreichen. Für das Englische definieren Grefenstette und Tapanainen (1994) z. B. folgende Regeln, die sie auf alle Zeichenketten anwenden, die auf einen Punkt enden:

1. Folgt dieser Sequenz ein kleingeschriebener Buchstabe, ein Komma oder ein Semikolon, ist es eine Abkürzung.
2. Ist das Wort eine kleingeschriebene Zeichenkette und existiert das gleiche Wort im Lexikon ohne Punkt, ist es keine Abkürzung; andernfalls wird es als Abkürzung klassifiziert.
3. Beginnt die Sequenz mit einem Großbuchstaben, ist keine bekannte Abkürzung, findet sich im Korpus ohne abschließenden Punkt und besitzt nur eine sehr geringe Häufigkeit, handelt es sich nicht um eine Abkürzung, sondern wahrscheinlich um einen Eigennamen.
4. Andernfalls handelt es sich um eine Abkürzung.

Ergänzt man einen solchen Ansatz durch eine Liste häufig verwendeter Abkürzungen, lässt sich eine Genauigkeit von über 99% erzielen.

Statistische Verfahren Ähnliche Genauigkeit erreichen Ansätze mit statistischen Verfahren. Dabei muss zwischen Systemen unterschieden werden, die ein Referenzkorpus als Trainingsmaterial erfordern und solchen, die unbeaufsichtigt

lernen. Letzteren wohnt die Einsicht inne, dass nur ein geringer Anteil von Interpunktionszeichen ambig ist, und sich Regeln aus denjenigen Fällen ableiten lassen, in denen Interpunktionszeichen eindeutig verwendet wird. All diesen Ansätzen gemein ist die Sichtweise, dass die Erkennung von Satzgrenzen ein Klassifikationsproblem darstellt, das sich durch Analyse morphosyntaktischer Merkmale eines Wortes am potentiellen Satzende und dessen Kontext lösen lässt. Derartige Kontexte werden in ***n*-gramm-Modellen** beschrieben (s. Unterkapitel 2.4), die über Vorkommenshäufigkeiten einzelner Wörter (**Unigramme**) hinaus mit bedingten Wahrscheinlichkeiten operieren: In einem **Bigramm-Modell** betrachtet man die Wahrscheinlichkeit dafür, dass ein Wort abhängig von einem vorhergehenden Wort auftritt, in einem **Trigramm-Modell** dementsprechend die Wahrscheinlichkeit, dass das Vorkommen eines Tokens von den beiden vorherigen abhängig ist. Restringiert wird die Auswahl des *n*-gramm-Modells durch die Größe des Korpus: Je kleiner dieses ist, desto geringer ist die Wahrscheinlichkeit, eine Sequenz der Länge *n* zu finden; dies bezeichnet man auch als **Sparse-Data-Problem**. Für gewöhnlich beschränkt man sich daher auf Bi- oder Trigramm-Modelle.

Schmid (2000) stellt einen Ansatz des *unsupervised learnings* dar. In seinem System unterscheidet er verschiedene Abkürzungsarten, denen er jeweils eigene Wahrscheinlichkeitsmodelle zuordnet. In einem ersten Durchlauf ermittelt der Tokenizer statistische Informationen über potentielle Abkürzungen und Namen, kleingeschriebene Wörter (als Indikatoren für Satzgrenzen, wenn sie nach einem Punkt großgeschrieben werden) und Wörter, die vor und hinter Zahlen auftreten (zur Disambiguierung von Zahlenformaten). In einem zweiten Durchlauf werden dann aufgrund dieser Zahlen Wahrscheinlichkeiten dafür berechnet, ob ein Token eine Abkürzung darstellt oder am Satzende steht.

Das in Kiss und Strunk (2006) vorgestellte *Punkt*-System stellt eine Weiterentwicklung des in Kiss und Strunk (2002) vorgestellten Lösungsansatzes zur Unterscheidung zwischen Abkürzungen und Satzendepunkten dar. Zentral ist dabei die Betrachtung des Abkürzungspunkts als Teil einer Kollokation: Während sich für einen Satzendepunkt in Bezug auf den Satz keine Merkmale finden lassen, die ihn als solchen charakterisieren würden, bildet der Abkürzungspunkt nahezu immer mit dem abzukürzenden Wort eine Kollokation – die Autoren reservieren ein Prozent Unsicherheit für Fälle, in denen die Abkürzung fälschlicherweise ohne Punkt in einem Korpus vorkommt. Als weitere Charakteristika einer Abkürzung verwenden sie die geringe Länge eines solchen Worts sowie das relativ häufige Auftreten eines wortinternen Punkts. In einem zweistufigen Verfahren bestimmen sie zunächst auf Typen-Ebene mögliche Abkürzungen, Ellipsen und Satzendepunkte, um dann auf Token-Ebene anhand dieser Befunde und weiterer Heuristiken sowohl eine qualitative Verbesserung bzw. Korrektur dieser Zwischenergebnisse zu erzielen, aber auch um satzfinale Abkürzungen und Ellipsen zu bestimmen. Dazu verwenden sie die in Dunning (1993) entwickelte *log-likelihood ratio*, die allerdings auf der Typen-Ebene in einigen Punkten modifiziert und durch obengenannte Faktoren wie die Wortlänge und die Anzahl der wortinternen Punkte skaliert wird. Auf Token-Ebene sorgen Heuristiken wie z. B. orthographische Hinweise über korpusinterne Häufigkeiten satzinterner Groß-

und Kleinschreibung bestimmter Wortklassen dafür, Satzenden bestimmen zu können, die mit Abkürzungen oder Ellipsen zusammenfallen, während die Kollokationsheuristik aufgrund der Kollokationswahrscheinlichkeit zweier um einen Punkt herum gruppierter Wörter als Hinweis gegen die Verwendung als Satzendmarkierung verstanden wird. Daraüber hinaus soll ein Histogramm Wörter identifizieren, die typischerweise am Satzanfang stehen, um so einen weiteren Hinweis auf einen Satzendpunkt zu bekommen. Kiss und Strunk haben dieses Verfahren sowohl mit elf verschiedenen Sprachen als auch auf unterschiedlichen Textarten getestet, wobei es sich als sehr genau und robust erwiesen hat. Eine Implementierung im NLTK findet sich unter <http://code.google.com/p/nltk/source/browse/trunk/nltk/tokenize/punkt.py>.

Palmer und Hearst (1997) verwenden in ihrem *Satz* genannten System sowohl Heuristiken als auch Algorithmen des maschinellen Lernens zur Tokenisierung. Schätzungen über die Verteilung von Wortarten derjenigen Token, die einen Trigramm-Kontext um einen Punkt herum bilden, werden als Vektoren repräsentiert und dienen einem Lernalgorithmus zur Disambiguierung von Satzgrenzen als Ausgangsmaterial. Dazu werden zunächst in einem Trainingsdurchlauf anhand eines kleinen manuell disambiguierten Korpus Regeln zur Erkennung von Satzgrenzen erlernt. Bei der eigentlichen Tokenisierung eines unbekannten Korpus wird dann mittels Heuristiken eine erste Auswahl an Token getroffen. Im zweiten Schritt wird dem jeweiligen Token aufgrund im Lexikon abgelegter Häufigkeiten über das Auftreten eines Wortes mit einer bestimmten Wortart der wahrscheinlichste Wortarttag zugewiesen, wodurch sich der Parameterraum extrem reduziert. Daraufhin findet eine Klassifikation durch den Lernalgorithmus statt: Zur Disambiguierung kommen sowohl neuronale Netze als auch Entscheidungsbäume zum Einsatz, wobei beide Verfahren ähnlich gute Ergebnisse liefern.

3.4.2 Wortart-Tagging

Problemstellung

Der Segmentierung eines Korpus folgt für gewöhnlich seine Anreicherung um grammatische Informationen. Zunächst soll jedes Token gemäß seiner Wortart klassifiziert werden. Ein System, das eine solche Analyse und Generierung leistet, bezeichnet man als **Part-of-Speech Tagger** (POS-Tagger). Auch hier unterscheidet man wiederum stochastische von regelbasierten Verfahren: Während erstere über Wahrscheinlichkeiten, dass ein Wort im betrachteten Kontext einer bestimmten Wortart angehört, operieren, ordnen regelbasierte Tagger dem Token in einem ersten Schritt denjenigen Tag zu, mit dem es in einem Trainingskorpus am häufigsten annotiert wurde. Dieses Vorgehen stellt wiederum die Baseline dar; die Genauigkeit beträgt in diesem Fall ungefähr 90%. Anschließend werden iterativ Transformationen gelernt und angewandt, die zur Verbesserung der Genauigkeit beitragen.

Ein wichtiges Kriterium für die Performanz des Taggings stellt die Auswahl des Inventars an Wortarten und Klassifikationen für Interpunktion, numerische Angaben oder Daten, das sogenannte **Tagset**, dar. Bei dieser Entscheidung muss

zwischen der Granularität der Beschreibung und der Genauigkeit des Taggers abgewägt werden: Je detailreicher ein Tagset ist, desto feinere Unterscheidungen müssen vom Tagger getroffen werden; in Sprachen mit reichhaltiger Morphologie ergeben sich durchschnittlich weniger Ambiguitäten als in Sprachen, deren Token kaum morphologische Merkmale aufweisen. Im (Vollform-)Lexikon eines Taggers findet man Abbildungen der möglichen Wortarten auf Wortformen: Während beispielsweise ein Personalpronomen wie *er* nur dieser einen Kategorie zugeordnet werden kann, lässt sich *dem* sowohl als Demonstrativpronomen, Relativpronomen oder Artikel klassifizieren. Gleichwohl kann man anhand des wohl populärsten Tagsets für das Deutsche, dem **Stuttgart-Tübingen-Tagset** (STTS), zeigen, dass Ergänzungen des Inventars für die linguistische Analyse sinnvoll sein können, sie allerdings das Potential zur Ambiguität erhöhen (vgl. Rudolph et al. 2001):

- (3.26) Dies versteht er *schwerlich*.
- (3.27) Der Laster wiegt *schwer*.
- (3.28) Er ist *schwer*.
- (3.29) Der *schwere* Mann kam herein.

Laut STTS-Guidelines (Schiller et al. 1999) werden Adjektive, die formähnlich als attributive Adjektive auftreten auch in einer Adverbposition als prädiktative Adjektive klassifiziert. Das heißt, in den Beispielen (3.26)–(3.28) bekämen sie den Tag ADJD (prädiktives Adjektiv) und in (3.29) den Tag ADJA (attributives Adjektiv). Es ließe sich nun argumentieren, dass diese Beispiele ein Kontinuum bezüglich des Grades der Beteiligung des Nomens an der Modifikation der verbalen Relation markieren: Während in (3.26) kein Bezug auf das Nomen besteht, modifiziert das Adjektiv in (3.27) sowohl das Nomen als auch die Verbalphrase. In (3.28) hingegen wird nur das Nomen modifiziert, wohingegen in (3.29) kein Bezug auf die Relation besteht. Dementsprechend könnte man dafür plädieren, Token wie *schwerlich* unabhängig davon, ob sie de-adjektivisch abgeleitet sind oder nicht, als Adverb zu taggen, während man ansonsten den Empfehlungen der Entwickler des STTS folgt. Problematisch an diesem Ansatz ist die Tatsache, dass Verben, deren Adverbien auch in gewissem Umfang die beteiligten Nomina modifizieren, zu einer größeren semantischen Gruppe gehören (*sein*, *werden*, *bleiben*, *sich ausnehmen*, *gelten als*, ...) und theoretisch auch jedes andere Verb mit einem adjektivischen Prädikat versehen sein kann, das sich in erster Linie auf ein Nomen bezieht:

- (3.30) Er bat bescheiden (ADJD) darum, dass...

Tabelle 3.3 zeigt einige Tags einer leicht modifizierten Version des STTS samt Beispielen. Darüber hinaus benötigen die meisten Tagger ein manuell ausgezeichnetes Trainingskorpus, das einerseits korrekt und ohne Ambiguitäten getaggt ist, andererseits möglichst umfangreich ist. Denn eines der Hauptprobleme eines jeden Taggers stellen unbekannte Wörter dar. Um auch für diese Fälle eindeutige

ADJA	attributives Adjektiv	[das] große [Haus]
ADJD	adverbiales oder prädikatives Adjektiv	[er fährt] schnell [er ist] schnell
ADV	Adverb	schon, bald, doch
APPR	Präposition; Zirkumposition links	in [der Stadt], ohne [mich]
ARTDEF	bestimmter Artikel	der, die, das
KON	nebenordnende Konjunktion	und, oder, aber
NN	normales Nomen	Tisch, Herr, [das] Reisen
NE	Eigennamen	Hans, Hamburg, HSV
PRELS	substituierendes Relativpronomen	[der Hund ,] der
VVFIN	finites Verb, voll	[du] gehst, [wir] kommen [an]
VVPP	Partizip Perfekt, voll	gegangen, angekommen
\$.	Satzbeendende Interpunktionszeichen	. ? ! ; :

Tabelle 3.3: Tags einer leicht modifizierten Version des STTS

Lösungen anbieten zu können, bestünde eine sehr einfache, aber schlechte Strategie darin, dem jeweiligen Token alle oder keine Kandidaten des Tagsets oder den Tag UNBEKANNT zuzuweisen. Eine erste Näherung an eine praktikable Lösung stellt der Ansatz dar, den unbekannten Wörtern ausschließlich Tags der offenen Wortklassen zuzuweisen, was aber im Fall von Eigennamen, die einen Großteil der noch nicht gesehnen Token ausmachen, oftmals zum falschen Ergebnis führt. Die beste Lösung besteht allerdings darin, einerseits den linken und rechten Kontext eines Tokens in Betracht zu ziehen, andererseits aufgrund morphologischer Informationen zu einer Entscheidung über einen geeigneten Tag zu gelangen.

Verfahren

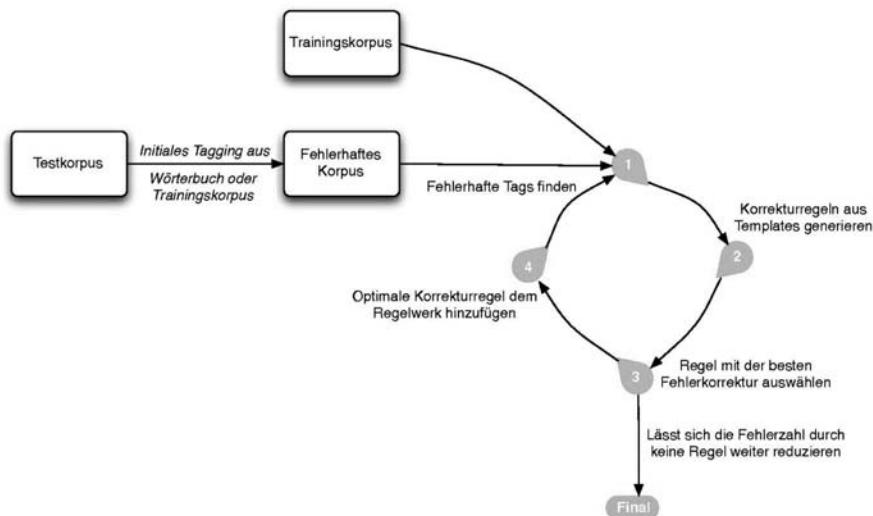
Zur Illustration der Tagging-Verfahren dient der folgende Satz mitsamt der Ambiguitätsklassen seiner einzelnen Wörter aus dem Lexikon und ihren Häufigkeiten aus einem ca. 80.000 Token umfassenden Trainingskorpus der Neuen Zürcher Zeitung als Ausgangsmaterial.

- (3.31) *Sonderrechte für Minoritätenkollektive widersprechen*
 (NN 2) (APPR 407) (NN 1) (VVFIN 1)
tatsächlich der klassischen
 (ADV 15, ADJD 1) (ARTDEF 2619, PRELS 162) (ADJA 3)
demokratischen Regel, wonach
 (ADJA 25) (NN 5) (\$, 5215) (PWAV 12)
die Mehrheit
 (ARTDEF 2351, PRELS 448, PDS 4) (NN 40)
bestimmt
 (VVFIN 4, ADV 1, ADJD 1, VVPP 7) (\$. 3252)

Der in Brants (2000b) entwickelte Tagger *Trigrams'n Tags* (TnT) stellt einen Ansatz eines stochastischen Taggers dar, der auf einem Markov-Modell zweiter Ordnung basiert. Aufgrund der Bayes-Formel (s. Unterkapitel 2.4) entsprechen die Zustände in diesem Modell den Tags, deren maximierte Wahrscheinlichkeiten die Wörter emittieren. Diese Wahrscheinlichkeiten sind als Trigramme modelliert, d.h. die Übergänge von einem Zustand zum nächsten hängen von den beiden vorherigen Tags ab. In einem ersten Schritt werden maximum likelihood-Wahrscheinlichkeiten für den Tag-Kandidaten im Verhältnis zur Gesamtzahl an Token im Trainingskorpus (relative Häufigkeit des Unigramms), im Verhältnis zum Tag davor (Bigramm) und der beiden vorherigen Tags (Trigramm) sowie zum zu emittierenden Wort im Lexikon berechnet. Bevor maximum likelihood-Wahrscheinlichkeiten aufgrund des Sparse-Data-Problems 0 werden können, wird ein sogenanntes **Smoothing** vorgenommen, d.h. man interpoliert aus den Uni-, Bi- und Trigramm-Häufigkeiten minimale Wahrscheinlichkeiten. Ebenso werden unbekannten Wörtern anhand von Suffix-Analysen Wahrscheinlichkeiten zugewiesen, die sich aus der Verteilung des jeweiligen Suffixes im Trainingskorpus errechnen. So lässt sich beispielsweise einem englischen Wort mit der Endung *-able* in 98% der Fälle die Wortart Adjektiv (*fashionable, variable*) zuweisen, während es sich nur in 2% der Fälle um ein Nomen (*cable, variable*) handelt.

Lässt man TnT den Beispielsatz taggen, weist er allen Wörtern mit Ausnahme von *bestimmt* den richtigen Tag zu. Anstatt eines finiten Vollverbs wird dieses Token als Vollverb im Partizip Perfekt ausgezeichnet. Dies liegt in der Tatsache begründet, dass die Tagsequenzen ARTDEF NN (*bestimmt VVFIN*) und ARTDEF NN (*bestimmt VVPP*) jeweils einmal im Trainingskorpus vorkommen. Aufgrund der höheren Frequenz von (*bestimmt VVPP*) im Lexikon annotiert das System hier allerdings falsch. Insgesamt erreicht TnT aber eine Genauigkeit von ungefähr 97%, was den Durchschnitt sowohl statistischer als auch regelbasierter Tagger markiert. Quellcode zu diesem Verfahren findet man im NLTK als <http://code.google.com/p/nltk/source/browse/trunk/nltk/nltk/tag/tnt.py>.

Brill (1995) stellt einen transformationsbasierten Ansatz dar. Im initialen Evaluationsdurchlauf bekommt jedes Token denjenigen Tag, mit dem es am häufigsten im Trainingskorpus auftritt. Unbekannten Wörtern wird pauschal der insgesamt frequenterste Tag im Trainingskorpus zugewiesen; dies ist oftmals ein Nomen. Danach durchläuft dieses rudimentär getaggte Korpus eine geordnete Liste von Transformationen, die ihrerseits wiederum aus Ersetzungsregeln und kontextuellen Bedingungen bestehen. In jedem Durchlauf wird jede mögliche Transformation auf jedes Wort-/Tag-Paar angewandt, die Anzahl der Tagging-Fehler wird gegen die korrekten Sequenzen des Trainingskorpus verrechnet und diejenige Transformation wird ausgewählt, die die wenigsten Fehler produziert. Dieser Zyklus endet, wenn es keine weiteren Transformationen gibt, welche die Fehlerrate unter einen vordefinierten Schwellwert senken. Um eine hohe Genauigkeit zu erreichen, benötigt man 100 bis 200 Regeln. Den Workflow illustriert folgende Grafik:



Brills Tagger produziert für den Beispielsatz das gleiche Ergebnis wie TnT; *bestimmt* wird in diesem Fall als VVPP kategorisiert, da dieser Tag im initialen Durchlauf der häufigste für dieses Token war. Zwar gibt es Regeln, die VVPP zu VVFIN ändern, jedoch keinen Kontext, in dem sie das gewünschte Ergebnis lieferten. Die Zahlen markieren die Rangfolge der Transformationen; für diesen Test bestand das Regelwerk aus 148 Transformationen:

48: VVPP VVFIN NEXTTAG ARTDEF

49: VVPP VVFIN NEXTTAG ADV

107: VVPP VVFIN NEXT1OR2TAG PRF

Im Gegensatz zu stochastischen Taggern besteht in diesem Framework die Möglichkeit, durch einfaches Hinzufügen einer Regel, das Ergebnis zu verbessern: Durch geschickte Anordnung bezüglich von Regeln, die das Vorhandensein von Auxiliaren berücksichtigen, könnte z. B. VVPP VVFIN PREVTAG NN zum richtigen Resultat führen.

Insgesamt erreicht auch Brill durchschnittlich 97% Genauigkeit, die sich durch Hinzufügen von Transformationen weiter verbessern lässt. Allerdings muss man auch bei dieser Entscheidung zwischen dem manuellen Aufwand, der oftmals auf ein bestimmtes Korpus beschränkt bleibt, und dem zu erwartenden Grad der Verbesserung abwägen. Eine Implementierung dieses Taggers im NLTK ist unter <http://code.google.com/p/nltk/source/browse/trunk/nltk/nltk/tag/brill.py> zu finden.

3.4.3 Chunk-Parsing

Problemstellung

Anders als beim traditionellen globalen Parsen eines Satzes – wie es im folgenden Unterkapitel 3.5 vorgestellt wird – liefert das Verfahren des **Chunk-Parsings**

(auch **partielles** oder **shallow Parsing** genannt) keine vollständigen syntaktischen Strukturen, die hierarchische Beziehungen zwischen Konstituenten repräsentieren, sondern identifizieren nebengeordnete Teilstrukturen (engl. *chunks*), denen phrasale Tags zugeordnet werden. Die Motivationen für derartige Verfahren speisen sich einerseits aus psycholinguistischen Beobachtungen, nach denen beim Sprachverstehen keine kompletten Sätze verarbeitet werden, sondern Teilstrukturen, andererseits aus der eher praktischen Tatsache, dass vollständige Parser weder eine hohe Genauigkeit ihrer Analysen aufweisen, noch sonderlich performant sind. Der signifikanteste Unterschied zwischen diesen beiden Formen des Parsings liegt darin, dass Chunk-Parser einerseits nur lokale syntaktische Abhängigkeiten erkennen, andererseits keine rekursiven Strukturen aufbauen, weshalb sie performanter sind und genauere Analysen liefern. Der sich daraus ergebende Verlust an Informationen im Vergleich zum vollständigen Parsen ist für viele Anwendungen irrelevant: Häufig dient der Output eines solchen Systems dem Information Retrieval oder der Text-Zusammenfassung als Input, kann aber auch einem vollständigen Parser als Ausgangsmaterial dienen.

Der Definition eines **Chunks** liegt Abney (1991) den Begriff des signifikanten Kopfes (engl. *major head*) zugrunde: Jedes inhaltstragende Wort, das nicht zwischen einem Funktionswort und dem von ihm selegierten Inhaltswort steht, stellt einen signifikanten Kopf dar. Eine Ausnahme bilden Pronomina, die wie lexisalisierte Nominalphrasen behandelt werden und den Status verwaister Wörter tragen:

(3.32) [PP in [NP those big houses]]

Zwar ist *in* hier syntaktischer Kopf der PP, gleichwohl ist es aber ein Funktionswort, sodass es nicht der signifikante Kopf des Chunks sein kann. In der von *in* selegierten NP finden sich die beiden inhaltstragenden Wörter *big* und *houses*; da letzteres zusätzlich den syntaktischen Kopf der Phrase darstellt, ist es gleichzeitig signifikanter Kopf der NP als auch der PP.

Das folgende Beispiel illustriert die obengenannten Unterschiede zwischen der Analyse von Teilstrukturen und dem globalen Parsing:

(3.33) [ADVP [ADJD Weltweit]][NP [NN Überraschung]]
 [VPFIN [VVFIN löste] [PP [APPRART im] [NN August] [CARD 1984]]]
 [NP [ARTDEF die] [NN Gründung]][NP [ARTIND einer] [NN Union]]
 [PP [APPR zwischen] [NP [ARTDEF dem] [NN König]]]
 [PP [APPR aus] [NE Marokko]]
 [KON und][NP [ARTDEF dem] [NN Revolutionär]]
 [PP [APPR aus] [NP [NE Libyen]]][PTKVZ aus] [\$..]

Während in vollständigen syntaktischen Analysen dieses Satzes die eingebetteten NPs auch als solche gepräst würden, sind hier alle Nominalphrasen ab *die Gründung* auf einer Ebene aneinander gereiht. Grammatische Informationen wie z. B. Subjekt oder Objekt fehlen genauso wie syntaktische Abhängigkeiten, beispielsweise die Relation zwischen Verb und Objekt (vgl. Lemnitzer und Naumann 2002).

Verfahren

Anhand eines Satzes aus Heinrich Bölls *Ansichten eines Clowns* soll der im Deutschen Referenzkorpus (DEREKO) verwendete Chunk-Parser als Beispiel eines Systems vorgestellt werden, das getagten Text als Ausgangsmaterial nimmt, aus dem es Teilstrukturen analysiert:

Input des Chunk-Parsers	Output des Chunk-Parsers
[KON und]	[KON und]
[PROAV außerdem]	[PROAV außerdem]
[VVFIN sorgt]	[VVFX [VVFIN sorgt]]
[PPOSAT mein]	[NX [PPOSAT mein]]
[NN Agent]	[NN Agent]]
[\$, ,]	[\$, ,]
[PRELS der]	[NXP [PRELS der]]
[PPOSAT meine]	[NX [PPOSAT meine]]
[NN Eigenheiten]	[NN Eigenheiten]]
[VVFIN kennt]	[VVXF [VVFIN kennt]]
[\$, ,]	[\$, ,]
[APPR für]	[PX [APPR für]]
[ARTIND eine]	[NX [ARTIND eine]]
[ADJA gewisse]	[AJXatt [ADJA gewisse]]
[NN Reibungslosigkeit]	[NN Reibungslosigkeit]]
[\$. .]	[\$. .]

Während Chunks, die linear aufeinander folgen, durch die wiederholte Anwendung des Chunk-Parsers erkannt werden, werden eingebettete Chunks vom System anhand kaskadierter Parser-Module analysiert, wobei der Output jeweils als Input eines weiteren Chunk-Parsers dient:

Ebene 0	[APPR für]	[ARTIND eine]	[ADJA gewisse]	
Ebene 1	[APPR für]	[ARTIND eine]	AJXatt	[ADJA gewisse]
Ebene 2	[APPR für]	NX	[ARTIND eine]	AJXatt
Ebene 3	PX	[APPR für]	NX	[ARTIND eine]
				[AJXatt [ADJA gewisse]]

Zuerst wird der adjektivische Chunk analysiert, da er nicht sonderlich komplex ist. Dies gehorcht dem in Abney (1996) beschriebenen Prinzip des *easy-first parsings*: Diejenigen Strukturen, die mit dem geringsten Aufwand erkannt werden können, werden zuerst ausgezeichnet. Danach werden komplexere Strukturen auf höheren Ebenen identifiziert, was zu weiteren *islands of certainty* führt. Die Anzahl der zu behandelnden Einheiten reduziert sich auf jeder Ebene, wodurch auch der Grad an Ambiguität im Text abnimmt, bis eine *containment of ambiguity* erreicht ist.

Technisch betrachtet basiert der hier beschriebene Chunk-Parser ähnlich wie Brills Tagger auf Regeln. Zur Identifikation einer Adjektivphrase findet sich beispielsweise u.a. folgende Regel:

```
SEQ
T[POS='PTKNEG'] QUEST
T[POS='ADV'] QUEST constrain to 'ADVLEX' with tag 'ADVMOD'
T[POS='ADJD'] STAR
CH[C='ADJTRUNC'] QUEST
T[POS='ADJA'] PLUS
```

Diese Regel besagt, dass zur Identifikation eines attributiven Adjektiv-Chunks auf der zweiten Ebene eines kaskadierten Parser-Systems eine Sequenz aus drei Token (T), einem Chunk (CH) und einem weiteren Token notwendig ist. Die Attribute QUEST, STAR und PLUS entsprechen dabei den aus regulären Ausdrücken bekannten Kleeneschen Operatoren der Quantität *, +, ?. Ausformulieren ließe sich diese Regel ungefähr so: Weise die Markierung AJXatt zu, wenn sich eine optionale Negationspartikel vor einem optionalen Adverb, das seinerseits durch die Eigenschaft, ein modifizierendes Adverb zu sein eingeschränkt wird, findet. Folgt dieser Konfiguration ein optionales prädikatives Adjektiv, gefolgt von einem Chunk, der Adjektivstämme (wie z. B. *anglo-irische*) identifiziert und einem oder mehreren attributiven Adjektiven, handelt es sich um einen AJXatt-Chunk. Existiert mehr als ein möglicher Parse, wird die Ambiguität durch den längsten Match aufgelöst.

3.4.4 Literaturhinweise

In diesem Kapitel wurden theoretische Ansätze und praktische Verfahren zur Tokenisierung, zum Wortart-Tagging und zum Chunk-Parsing skizziert. Einige Aspekte der Vorverarbeitung elektronischer Korpora konnten hier aus Platzgründen nicht berücksichtigt werden. So bleibt z. B. die Frage offen, ob es nicht Aufgabenstellungen gibt, die man nicht eindeutig der Tokenisierung oder dem Chunk-Parsing zuordnen kann. Betrachtet man beispielsweise die Analyse von Nominalkomposita, wird man feststellen, dass Sprachen bezüglich der Nutzung von Leerzeichen nicht regelhafte Unterschiede aufweisen: Eine *Schreibmaschine* ist im Englischen ein *typewriter*, französisch jedoch eine *machine à écrire*, genauso wie eine *Holztür* eine *wooden door* und eine *porte en bois* ist. Hingegen nennt man *Wiener Würstchen* auf Englisch *frankfurter* oder einfach *wiener*, während es im Französischen *saucisses à l'ail* sind. Die Problematik besteht also darin, mehrere Worte auf ein Token abzubilden, das in einem höher geordneten Analyseschritt verarbeitet werden kann. Der umgekehrte Schritt ist genauso denkbar: Besitzt eine Sprache reichhaltige Morphologie, kann es im speziellen Anwendungsfall angezeigt sein, allein die Stämme eines Tokens zu betrachten und die Affixmorphologie zu eliminieren. Offen blieb auch die Frage, welchen

Einfluss fehlerhafte Vorverarbeitung von Korpora auf weitergehende Analysen hat. Beispielsweise beschreiben Kiss und Strunk (2002) die Auswirkungen ihrer Tokenisierung auf das Tagging mit TnT.

Einen umfassenden Überblick über alle drei Teilgebiete bieten Lemnitzer und Naumann (2002) und Bird et al. (2009). In die Probleme der Tokenisierung führt Mikheev (2003) ein, während sich eine tiefergehende Darstellung über statistische und symbolische Verfahren beim Tagging in den Kapiteln 9 und 10 von Manning und Schütze (2003) und Kapitel 5 von Jurafsky und Martin (2009) finden. Reguläre Ausdrücke werden erschöpfend in Friedl und Oram (2006) behandelt. Einen soliden Einblick sowohl in das Tagging als auch in das Chunk-Parsing bietet Abney (1996).

3.5 Syntax und Parsing

Hagen Langer

In diesem Unterkapitel wird es darum gehen, wie syntaktische Strukturen repräsentiert und verarbeitet werden können. Der erste Abschnitt dient dem Zweck, in diejenigen Grundideen der Syntaxforschung einzuführen, die für den praktischen Zweck, computerlinguistische Systeme mit Syntax-Komponenten zu entwickeln, von besonderer Bedeutung sind. Dabei wird bewusst darauf verzichtet, die Vielfalt der gegenwärtig verfügbaren grammatischen Repräsentationsmodelle in ihrer ganzen Breite darzustellen, stattdessen werden einige der – z.T. sehr alten – Grundprobleme und Grundideen der Syntaxforschung skizziert, mit denen sich wohl fast jeder, der den Versuch macht, einen Teilbereich der Syntax einer natürlichen Sprache mit den Mitteln der Computerlinguistik zu modellieren, auseinandersetzen muss.

Während sich der erste Abschnitt dieses Kapitels den deklarativen Aspekten der Repräsentation von syntaktischen Repräsentationen, Regelsystemen usw. widmet, werden im zweiten Abschnitt die Probleme der prozeduralen Umsetzung fokussiert: Wie können **formale Grammatiken** verwendet werden, um natürlichsprachliche Ausdrücke syntaktisch zu analysieren?

Neben dem Basisinstrument der syntaktischen Beschreibung schlechthin, der kontextfreien Grammatik, werden im ersten Abschnitt Elemente des unifikationsbasierten Grammatikmodells PATR-II sowie einzelne Aspekte anderer unifikationsbasierter Formalismen erläutert. Abschließend werden Kategorialgrammatiken und statistische Erweiterungen angerissen.

Der zweite Abschnitt behandelt nach einer kurzen Einführung in die Grundlagen und Grundprobleme der Parsing-Theorie den wohl bekanntesten Parsingalgorithmus für die Analyse natürlichsprachlicher Ausdrücke, den Earley-Algorithmus.

Die syntaktische Analyse von natürlichsprachlichen Ausdrücken ist in verschiedenen Anwendungskontexten der Computerlinguistik relevant. In Systemen, die eine umfassende Interpretation von natürlichsprachlichen Eingaben anstreben (z. B. in den Bereichen Natürlichsprachlicher Auskunftsysteme und **Maschinelle Übersetzung**, vgl. Unterkapitel 5.7), spielt die syntaktische Analyse eine wichtige Rolle bei der **Disambiguierung** mehrdeutiger (**ambiger**) Ausdrücke und sie liefert eine strukturierte Eingabe, wie sie von vielen semantischen und pragmatischen Analysekomponenten vorausgesetzt wird. In Spracherkennungssystemen für gesprochene Sprache werden (zumeist sehr einfache) syntaktische Analysemodelle für die Disambiguierung mehrdeutiger Worthypothesen verwendet. Für statistische Untersuchungen von größeren Korpora werden syntaktische Komponenten zur Annotation mit Wortarteninformationen verwendet. Für die beiden letztgenannten Anwendungsbereiche werden häufig **Hidden-Markov-Modelle** eingesetzt (vgl. auch die Unterkapitel 2.4, 5.4 und 3.4). Schließlich kann die syntaktische Analyse auch von zentraler Bedeutung sein, wenn z. B. in einem Textverarbeitungssystem für eine intelligente Rechtschreib- und Interpunktions-

korrektur (vgl. Unterkapitel 5.1) syntaktische Informationen verwendet werden sollen.

3.5.1 Syntax

Syntaktische Strukturen

In der linguistischen Tradition lassen sich hinsichtlich der Frage, was syntaktische Strukturen eigentlich sind, zwei wesentliche Entwicklungsstränge unterscheiden: Die erste Hauptrichtung fasst syntaktische Strukturen als Relationen zwischen Wörtern auf; es handelt sich um die Tradition der **Dependenz- und Determinationssyntax**. In der anderen Hauptrichtung, der **Konstituentenstruktursyntax**, werden neben Wörtern auch komplexere Einheiten, die so genannten **Konstituenten** oder **Phrasen**, angenommen; zu den syntaktischen Strukturen zählen dort auch Relationen zwischen Konstituenten. In beiden Traditionen werden **Baumgraphen** bzw. **Strukturbäume** zur Notation von syntaktischen Strukturen verwendet. Abbildung 3.37 zeigt einen Konstituentenstrukturbaum aus dem Jahre 1902 (Dittrich 1902). Ein Dependenzbaum aus dem Jahre 1883 findet sich in Abb. 3.38 (Kern 1883).

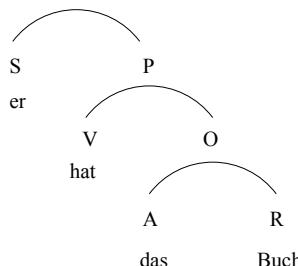


Abbildung 3.37: Konstituentenstruktur-Baumgraph nach Dittrich

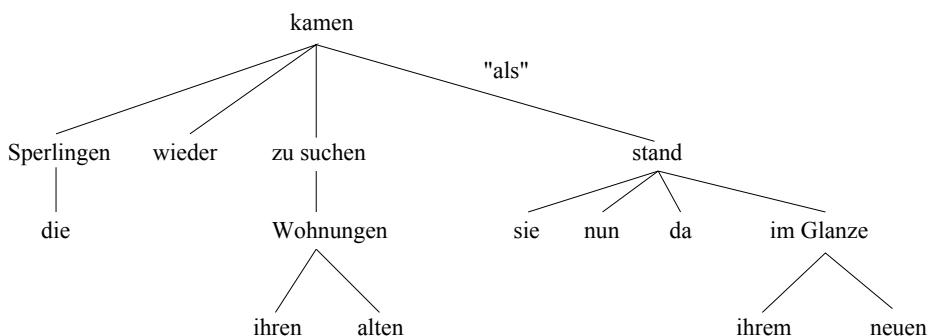


Abbildung 3.38: Dependenz-Baumgraph nach Kern

Hier wurden bewusst zwei ältere Beispiele gewählt, um dem weitverbreiteten Irrtum entgegenzuwirken, diese beiden Modelle und ihre jeweiligen graphischen

Notationsweisen stammten aus der Mitte des 20. Jahrhunderts³. Der Dependenzgraph gibt die Struktur des folgenden Satzes wieder.

- (3.34) *Als sie nun in ihrem alten Glanze da stand, kamen die Sperlinge wieder, ihre alten Wohnungen zu suchen.*

In der **Dependenzgrammatik** wird zumeist angenommen, dass das Verb die Struktur eines Satzes in entscheidendem Maße festlegt, alle anderen Wörter sind unmittelbar oder mittelbar vom Verb abhängig. Die Relation der Abhängigkeit, die der Dependenzgrammatik ihren Namen gegeben hat, wird in dem Baumgraphen durch die Kanten wiedergegeben: Unmittelbar von dem Verb *kamen* abhängig sind diejenigen Wörter, die mit ihm durch direkte Kanten verbunden sind (*Sperlingen*, *wieder*, *zu suchen* und *stand*). Vom Verb des durch *als* eingeleiteten Nebensatzes (*stand*) sind wiederum die Ausdrücke *sie*, *nun*, *da* und *im Glanze* abhängig usw.

Strukturbäume aus der Tradition der Konstituentenstrukturgrammatik wie der in Abb. 3.37 wiedergegebene enthalten folgende Informationen:

- Segmentierung: Zerlegung eines komplexen Ausdrucks in Teile, die wiederum komplex sein können. In dem oben angegebenen Strukturbau wird der Satz z. B. zunächst in die Hauptbestandteile *er* und *hat das Buch* zerlegt, anschließend wird *hat das Buch* in *hat* und *das Buch* segmentiert.
- Kategorisierung der komplexen Teilausdrücke: z. B. S (Subjekt), P (Prädi-kat) usw.
- Lineare Abfolge: In diesem Baum entspricht die lineare Abfolge der Blätter des Baums der Wortstellung.

Im Kontext computerlinguistischer Untersuchungen werden sowohl dependenzorientierte als auch konstituentenorientierte syntaktische Beschreibungsmodelle verwendet, vor allem aber auch Systeme, die Elemente aus beiden Modellen miteinander verbinden. Da die konstituentenorientierte Tradition jedoch – vor allem aufgrund der richtungweisenden Arbeiten von Noam Chomsky (Chomsky 1957, Chomsky 1981) – insgesamt einen wesentlich stärkeren Einfluss auf die Computerlinguistik der vergangenen Jahrzehnte hatte, werden wir uns im Folgenden vor allem auf diesen Ansatz konzentrieren.

Kontextfreie Grammatiken

Kontextfreie Grammatiken (Typ-2-Grammatiken, vgl. Unterkapitel 2.2) sind – zumindest innerhalb der Tradition der konstituentenstrukturorientierten Grammatikmodelle – nach wie vor das Basisinstrument für syntaktische Analysen, wenngleich sie heutzutage nur noch selten in reiner Form verwendet werden. Zumeist bildet eine kontextfreie Grammatik das Grundgerüst (oder Skelett) eines

³Die beiden Baumgraphen sind Thümmel (1993), einem Beitrag zum Internationalen Syntax-Handbuch (Jacobs et al. 1993), entnommen.

Systems, das auch andere Elemente, z. B. statistische Bewertungen oder komplexe Kategorien (etwa in Form von Merkmalsstrukturen, vgl. Unterkapitel 2.3) enthält. Die formale Definition von kontextfreien Grammatiken wurde bereits in Unterkapitel 2.2 gegeben. In diesem Abschnitt soll es um die Möglichkeiten und Grenzen bei der Verwendung kontextfreier Systeme bei der Definition von Grammatiken für natürliche Sprachen gehen. Deshalb sollen die Komponenten einer kontextfreien Grammatik nochmals kurz im Hinblick auf die Verwendung für natürliche Sprache wiederholt werden. Eine kontextfreie Grammatik $G = \langle \Phi, \Sigma, R, S \rangle$ besteht aus

1. einer Menge von Nichtterminalsymbolen Φ . Sie enthält typischerweise syntaktische Kategorien wie S (Satz), NP (Nominalphrase), VP (Verbalphrase), PP (Präpositionalphrase) sowie Wortartenkategorien wie V (Verb) und N (Nomen). Letztere werden **präterminale Kategorien** genannt, da sie ausschließlich terminale Symbole dominieren. Regeln, die präterminale Kategorien zu einem Element des terminalen Vokabulars expandieren, werden **lexikalische Regeln** bzw. **Lexikonregeln** genannt.
2. einer Menge von Terminalsymbolen Σ . Diese enthält sämtliche atomaren, d.h. nicht weiter zerlegbaren Bestandteile derjenigen Ausdrücke, die durch die Grammatik definiert werden. Typischerweise sind diese atomaren – zumindest syntaktisch – nicht weiter zerlegbaren Einheiten die Wörter der zu beschreibenden Sprache.
3. einer Regelmenge R , die endlich viele Regeln der Form

$$A \rightarrow \alpha$$

enthält, wobei A ein Nichtterminalsymbol ist (z. B. die syntaktische Kategorie V) und α eine Kette von Symbolen aus Φ und Σ ist (z. B. die Kette, die aus dem deutschen Wort *liest* besteht). Es ist zulässig, dass die rechte Regelseite aus der leeren Kette besteht, d.h. gar keine Symbole enthält.

4. einem **Startsymbol** S . Das Startsymbol ist Element des nichtterminalen Vokabulars und etikettiert die Wurzeln der Bäume, die sich auf der Basis einer gegebenen Grammatik generieren lassen. Sofern die durch die Grammatik zu beschreibenden Ausdrücke Sätze (und nicht etwa Wörter oder Texte) sind, verwendet man das Startsymbol S, das die syntaktische Kategorie *Satz* bezeichnet.

$$\begin{aligned}
 G = & \langle \{ & S, NP, VP, DET, N, V \}, \\
 & \{ & der, Hund, bellt, sieht, die, Katze \}, \\
 & \{ & S & \rightarrow NP\ VP, & (1) \\
 & NP & \rightarrow DET\ N, & (2) \\
 & VP & \rightarrow V, & (3) \\
 & VP & \rightarrow V\ NP, & (4) \\
 & DET & \rightarrow der, & (5) \\
 & DET & \rightarrow die, & (6) \\
 & N & \rightarrow Hund, & (7) \\
 & N & \rightarrow Katze, & (8) \\
 & V & \rightarrow bellt, & (9) \\
 & V & \rightarrow sieht \}, & (10) \\
 & S \rangle
 \end{aligned}$$

Abbildung 3.39: Eine kontextfreie Grammatik für ein Fragment des Deutschen

Beispiel 3.5.1

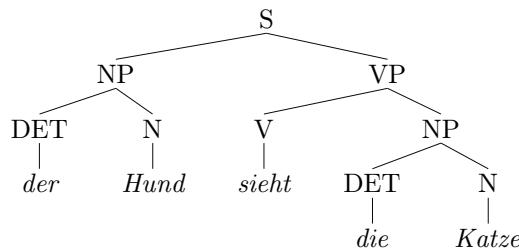
Die kontextfreie Grammatik in Abbildung 3.39 beschreibt ein kleines Fragment des Deutschen. Zu den mit Hilfe dieser Grammatik ableitbaren Ketten gehören:

(3.35) *der Hund bellt*

(3.36) *der Hund sieht die Katze*

(3.37) **der Hund bellt die Hund*

Für den Satz *der Hund sieht die Katze* ergibt sich z. B. die folgende Strukturbeschreibung:



Die Wurzel des Baumgraphen ist mit dem Startsymbol S (für Satz) etikettiert. Der Satz besteht aus zwei Teilstrukturen, der Nominalphrase (NP) *der Hund* und der Verbalphrase (VP) *sieht die Katze*. Die NP *der Hund* besteht aus dem Determinierer (DET) *der* und dem Nomen (N) *Hund*. Die VP besteht aus dem Verb (V) *sieht* und der NP *die Katze*, die dieselbe interne Struktur aufweist wie die Subjekts-NP. □

Um ungrammatische Ausdrücke wie **der Hund bellt die Hund* ausschließen zu können, müsste das Kategorieninventar der Beispielgrammatik in den folgenden Punkten ausdifferenziert werden:

- Statt einer einheitlichen Kategorie V für Verben wird eine Differenzierung in transitive und intransitive Verben benötigt; die neuen Kategorien könnten z. B. V_t und V_i heißen. Nur transitive Verben (wie *sehen*) können ein Akkusativ-Objekt zu sich nehmen, intransitive (wie *bellen*) hingegen nicht.
- Man benötigt Genus-Informationen, um nicht-wohlgeformte Ausdrücke wie **die Hund* ausschließen zu können. Diese Genus-Informationen betreffen sowohl die Artikel (Kategorie DET) als auch die Nomina (Kategorie N). Wenn wir zunächst nur die Unterscheidung zwischen Maskulina und Feminina einführen und Neutra vernachlässigen, ergeben sich die neuen Kategorien DET_m und DET_f sowie N_m und N_f .
- Schließlich benötigen wir noch Kasus-Informationen. Für dieses kleine Fragment des Deutschen können wir uns auf die Unterscheidung zwischen Nominativ und Akkusativ beschränken. Diese Unterscheidung muss bei Artikeln (Kategorie DET), Nomina (Kategorie N) und Nominalphrasen (Kategorie NP) vorgesehen sein.

Wir erhalten somit insgesamt die folgenden neuen Kategorien:

- DET_{mn} , DET_{ma} , DET_{fn} , DET_{fa}
- N_{mn} , N_{ma} , N_{fn} , N_{fa}
- NP_n , NP_a
- V_t , V_i

Mit dieser Ausdifferenzierung des Kategorieninventars ist allerdings nur der erste Schritt zur Verbesserung der Grammatik getan: Nun sind auch noch sämtliche Regeln zu korrigieren, in denen die modifizierten neuen Kategorien vorkommen. Aus der alten Regel $\text{NP} \rightarrow \text{DET N}$ entstehen z. B. die folgenden neuen Regeln:

$$\begin{array}{lcl} \text{NP}_{mn} & \rightarrow & \text{DET}_{mn} \text{ N}_{mn} \\ \text{NP}_{ma} & \rightarrow & \text{DET}_{ma} \text{ N}_{ma} \\ \text{NP}_{fn} & \rightarrow & \text{DET}_{fn} \text{ N}_{fn} \\ \text{NP}_{fa} & \rightarrow & \text{DET}_{fa} \text{ N}_{fa} \end{array}$$

Man kann sich nun leicht vorstellen, dass sich sowohl das Inventar von Kategorien als auch der Umfang der Regelmenge explosionsartig vergrößert, wenn man das durch die Grammatik definierte Fragment erweitern will, z. B. um die übrigen Kasus Dativ und Genitiv, um Singular und Plural, um Nominalphrasen mit zusätzlichen Adjektiven usw. Die Vergrößerung der Grammatik bei der Einführung differenzierterer Fallunterscheidungen mag per se noch kein schwerwiegender Nachteil sein, allerdings deuten sich doch bereits einige grundsätzliche Probleme kontextfreier Grammatiken an:

- Generalisierungen wie „Innerhalb der NP herrscht Kongruenz bzgl. Kasus, Numerus und Genus“ lassen sich nicht explizit als solche formulieren. Das Phänomen der Kongruenz wird stattdessen auf verschiedene Regeln und Kategorien verteilt.

- Auch dadurch, dass nun die Grundstruktur einer NP, die aus Artikel und Nomen besteht, auf mehrere Regeln verteilt wird, zersplittet eine Generalisierung in eine Aufzählung einzelner Fälle.
- Die Kategorien des nicht-terminalen Vokabulars einer kontextfreien Grammatik sind atomare Symbole. Auch wenn die oben verwendete Notation mit Indizes wie $_m$ für maskulin oder $_a$ für Akkusativ den Eindruck erwecken mag, dass es sich dabei um Werte für Parameter wie Genus und Kasus handelt: Formal sind diese Indizes Bestandteile eines Namens für ein atomares Symbol. In kontextfreien Grammatiken sind keine Parameter vorgesehen, d.h. die Symbole N_{mn} und N_{ma} sind sich auf formaler Ebene keinen Deut ähnlicher als die Symbole N_{mn} und VP. Mit den Mitteln einer kontextfreien Grammatik ist es also nicht möglich, Beziehungen zwischen Kategorien, z.B. die so genannte **Subkategorisierung**, d.h. die Zerlegung einer Grundkategorie (z.B. V) in Unterkategorien (z.B. V_t und V_i), explizit auszudrücken.

Man kann nun folgendes vorläufiges Fazit ziehen:

- Mit kontextfreien Grammatiken lassen sich Grundelemente der syntaktischen Struktur von natürlichsprachlichen Ausdrücken beschreiben, z.B.
 - die Segmentierung von komplexen Ausdrücken in (möglicherweise wiederum komplexe) Teilausdrücke,
 - die Zuordnung von Klassen von Ausdrücken zu Kategorien (z.B. NP, VP usw.)
- Andererseits weisen kontextfreie Grammatiken Defizite auf, wenn es um die explizite Repräsentation von bestimmten syntaktischen Phänomenen geht, zu denen unter anderem die folgenden zählen:
 - Kongruenz. Beispiel: die Übereinstimmung hinsichtlich Kasus, Numerus und Genus in der deutschen Nominalphrase.
 - Subkategorisierung. Beispiel: die Unterscheidung zwischen transitiven und intransitiven Verben in der deutschen Verbalphrase.

Im folgenden Abschnitt werden wir Erweiterungen von kontextfreien Grammatiken diskutieren, die Lösungsansätze für die genannten Probleme anbieten.

Unifikationsgrammatiken

Wenn in der Linguistik komplexe Objekte repräsentiert werden sollen, die durch mehrere Eigenschaften charakterisiert werden können, verwendet man zu diesem Zweck häufig Merkmalsstrukturen (s. Unterkapitel 2.3).

Im Standard-Modell der generativen Grammatik (Chomsky 1965) wurden Merkmale z.B. zu dem Zweck verwendet, Verben zu subkategorisieren. Um 1980 entstanden mehrere Grammatiktypen, in denen Merkmalsstrukturen eine zentrale Rolle spielen, die so genannten **Unifikationsgrammatiken**. Zu den

wichtigsten Vertretern zählen die **Generalized Phrase Structure Grammar** (GPSG; Gazdar, Klein, Pullum und Sag 1985), die **Lexical Functional Grammar** (LFG; Bresnan 1982), **PATR-II** (Shieber 1986) und die **Head-Driven Phrase Structure Grammar** (HPSG; Pollard und Sag 1994). Das einfachste der genannten Modelle ist der von Stuart Shieber entwickelte Grammatikformalismus PATR-II (PATR steht für *Parsing and Translation*, dem intendierten Anwendungsbereich des Grammatikmodells. PATR-II hatte einen Vorgänger PATR, der allerdings nicht unifikationsbasiert war).

Allen Unifikationsgrammatiken ist gemeinsam, dass sie für die Beschreibung syntaktischer Strukturen Merkmalsstrukturen (oder vergleichbare komplexe Datentypen) verwenden. Bei der Konstruktion von Strukturbeschreibungen werden diese Merkmalsstrukturen mit Hilfe der Unifikationsoperation (s. Unterkapitel 2.3) miteinander verknüpft.

PATR-II: PATR-II ist der einfachste der oben genannten unifikationsbasierten Grammatikformalismen. Einfachheit bezieht sich hier nicht auf beschränkte Ausdruckskraft oder geringe Komplexität im formalen Sinne – Generalisierte Phrasenstrukturgrammatiken (GPSG) sind beispielsweise formal eingeschränkter –, sondern auf die Anzahl und Komplexität der zugrunde liegenden linguistischen Annahmen. Während sich z. B. in der HPSG Strukturbeschreibungen aus dem Zusammenspiel verschiedener Prinzipien ergeben, beschränkt sich PATR-II auf einen grundlegenden Regeltyp. Stuart Shieber hat Systeme wie PATR-II oder **DCG** (Definite Clause Grammar) als **Werkzeugformalismen** (engl. *tool formalisms*) charakterisiert. Grammatikmodelle, die auf mehr oder weniger expliziten linguistischen Annahmen beruhen (z. B. GPSG, LFG und HPSG) bezeichnet er als **Theorieformalismen** (engl. *theory formalisms*).

Eine PATR-II-Grammatik besteht aus zwei Komponenten: einem Lexikon und einer endlichen Menge von Grammatikregeln. Jede Regel enthält als Kernbestandteil einen kontextfreien Kern (oder ein kontextfreies „Skelett“). Im Gegensatz zu einer traditionellen kontextfreien Grammatik sind die syntaktischen Kategorien einer PATR-II-Grammatik allerdings nicht als atomare Symbole konzipiert, sondern als Merkmalsstrukturen.

Beispiel 3.5.2

Das folgende Beispiel zeigt eine PATR-II-Regel, die man als Erweiterung der oben angegebenen kontextfreien Regel (2) in Beispiel 3.5.1 ansehen kann.

$$\begin{aligned} X_0 &\rightarrow X_1 X_2 \\ \langle X_0 \text{ CAT} \rangle &= \text{NP} \\ \langle X_1 \text{ CAT} \rangle &= \text{DET} \\ \langle X_2 \text{ CAT} \rangle &= \text{N} \end{aligned}$$

□

Anstelle der Symbole werden in der kontextfreien Regel Variablen (X_0 , X_1 und X_2) verwendet. Die darunter stehenden so genannten **Pfadgleichungen** weisen

diesen Variablen Restriktionen zu: So wird z. B. in der ersten Pfadgleichung festgelegt, dass das Attribut CAT bei der Kategorie X_0 den Wert NP hat. Analog dazu werden X_1 und X_2 die Kategoriennamen DET und N zugewiesen. Bis zu diesem Punkt enthält die PATR-II-Regel keine weitere Information, die nicht auch schon in der kontextfreien Phrasenstrukturregel (2) gegeben wäre. Nun lassen sich aber zusätzliche Informationen – z. B. über Kongruenz – mit Hilfe zusätzlicher Pfadgleichungen ergänzen:

$$X_0 \rightarrow X_1 X_2$$

$$\begin{aligned}\langle X_0 \text{ CAT} \rangle &= \text{NP} \\ \langle X_1 \text{ CAT} \rangle &= \text{DET} \\ \langle X_2 \text{ CAT} \rangle &= \text{N} \\ \langle X_1 \text{ AGR} \rangle &= \langle X_2 \text{ AGR} \rangle\end{aligned}$$

Die neue Pfadgleichung $\langle X_1 \text{ AGR} \rangle = \langle X_2 \text{ AGR} \rangle$ legt fest, dass die Kategorien X_1 und X_2 denselben Wert für das Attribut AGR tragen, oder genauer gesagt, dass diese Wert unifizierbar sein müssen. Wenn wir annehmen, dass das Attribut AGR Kasus-, Numerus- und Genus-Informationen betreffen soll, können wir für die Lexikoneinträge von Wörtern aus deklinierbaren Wortarten, also z. B. Determinierer und Nomina, Spezifikationen wie die folgenden konstruieren:

<i>Hund</i>			<i>Katze</i>		
CAT N			CAT N		
AGR $\left[\begin{array}{ll} \text{KASUS} & \text{nom} \\ \text{NUM} & \text{sg} \\ \text{GENUS} & \text{mask} \end{array} \right]$			AGR $\left[\begin{array}{ll} \text{NUM} & \text{sg} \\ \text{GENUS} & \text{fem} \end{array} \right]$		
<i>der</i>			<i>die</i>		
CAT DET			CAT DET		
AGR $\left[\begin{array}{ll} \text{KASUS} & \text{nom} \\ \text{NUM} & \text{sg} \\ \text{GENUS} & \text{mask} \end{array} \right]$			AGR $\left[\begin{array}{ll} \text{KASUS} & \text{nom} \\ \text{NUM} & \text{sg} \\ \text{GENUS} & \text{fem} \end{array} \right]$		

Kombiniert man nun diese Lexikoneinträge mit der oben angegebenen PATR-II-Regel, dann lassen sich die folgenden wohlgeformten Nominalphrasen ableiten:

(3.38) *der Hund*

(3.39) *die Katze*

Nicht-wohlgeformte Ausdrücke wie *die Hund*, die von der ursprünglichen kontextfreien Grammatik noch akzeptiert wurden, werden von dieser PATR-II-Grammatik abgelehnt, da die Forderung nach Unifizierbarkeit der AGR-Spezifikationen, die von der letzten Pfadgleichung gestellt wird, mit den entsprechenden

Werten in den beiden Lexikoneinträgen für *die* ($\langle \text{AGR GENUS} \rangle = \text{fem}$) und ($\langle \text{AGR GENUS} \rangle = \text{mask}$) nicht erfüllbar ist.

Ein wichtiges Element von PATR-II (aber auch anderen unifikationsbasierten Formalismen) ist die Möglichkeit, besonders generelle und kompakte Repräsentationen durch **Unterspezifikation** der Merkmalsstrukturen zu erzielen. So fehlt z. B. im Lexikoneintrag für Katze jegliche Kasusinformation. Dies bedeutet jedoch nicht, dass das Wort *Katze* keinen Kasus hat, sondern dass es mit beliebigen Kasusspezifikationen unifiziert.

Abschließend kommen wir auf das zweite Problem unserer ursprünglichen kontextfreien Grammatik zurück: Wie können Subkategorisierungen, z. B. die Differenzierung zwischen transitiven und intransitiven Verben, adäquater repräsentiert werden? Da diese Eigenschaften idiosynkratische Eigenschaften der jeweiligen Verben sind, ist das Lexikon der geeignete Ort für diese Spezifikationen. In einer PATR-II-Grammatik könnten die Lexikoneinträge für die beiden Verbformen *bellt* und *sieht* wie folgt aussehen:

	<i>bellt</i>	<i>sieht</i>
CAT	V	V
AGR	$\begin{bmatrix} \text{TEMP} & \text{pres} \\ \text{NUM} & \text{sg} \\ \text{PER} & 3 \end{bmatrix}$	$\begin{bmatrix} \text{TEMP} & \text{pres} \\ \text{NUM} & \text{sg} \\ \text{PER} & 3 \end{bmatrix}$
SUBCAT	<i>none</i>	$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{KASUS} & \text{akk} \end{bmatrix}$

Diese beiden Lexikoneinträge können nun mit folgenden beiden VP-Regeln kombiniert werden:

$$X_0 \rightarrow X_1$$

$$\langle X_0 \text{ CAT} \rangle = \text{VP}$$

$$\langle X_1 \text{ CAT} \rangle = \text{V}$$

$$\langle X_0 \text{ AGR} \rangle = \langle X_1 \text{ AGR} \rangle$$

$$X_0 \rightarrow X_1 X_2$$

$$\langle X_0 \text{ CAT} \rangle = \text{VP}$$

$$\langle X_1 \text{ CAT} \rangle = \text{V}$$

$$\langle X_0 \text{ AGR} \rangle = \langle X_1 \text{ AGR} \rangle$$

$$\langle X_1 \text{ SUBCAT} \rangle = \langle X_2 \rangle$$

Die rechte Regel könnte nun auch für alle anderen Verben verwendet werden, die genau ein Komplement haben (z. B. ein Dativ- oder Präpositionalobjekt) – in Schieber (1986) wird gezeigt, wie sich diese Grundidee weiterführen lässt, dort werden z. B. auch Subjekte subkategorisiert. Während bei einer kontextfreien Grammatik Lexikon und Regelapparat erweitert werden müssten, wenn Verben mit neuen Subkategorisierungseigenschaften aufgenommen werden sollen, beschränken sich die erforderlichen Änderungen in einer PATR-II-Grammatik auf das Lexikon. Insgesamt sind also die Regeln einer PATR-II-Grammatik im Vergleich zu einer kontextfreien genereller und unterspezifizierter, die Lexikoneinträge sind hingegen detaillierter und informativer. Diese Tendenz zu einer immer geringeren Anzahl von immer generelleren Syntaxregeln bei gleichzeitiger Präzisierung des Lexikons ist bei moderneren unifikationsbasierten Formalismen wie HPSG noch stärker ausgeprägt. Diese Schwerpunktverlagerung wird

gelegentlich (in Anlehnung an eine ältere Diskussion innerhalb der Chomsky-Tradition) **Lexikalismus** genannt.

GPSG: Generalisierte Phrasenstrukturgrammatiken (engl. *Generalized Phrase Structure Grammar*, GPSG, Gazdar et al. 1985) können als konservative Erweiterung von kontextfreien Grammatiken angesehen werden. Konservativ deshalb, weil die GPSG – zumindest in der Version von Gazdar et al. (1985) (eine frühere Version der GPSG (Gazdar und Pullum 1982) erfüllte dieses Kriterium nicht) – dieselbe **schwache generative Kapazität** wie kontextfreie Grammatiken haben und deshalb im Vergleich zu LFG- oder HPSG-basierten Grammatiken erheblich effizienter zu verarbeiten sind. Grammatiken im GPSG-spezifischen Format, dem sog. ID/LP-Format (s.u.), lassen sich systematisch in traditionelle kontextfreie Grammatiken umformen. Dies allerdings gilt nur, wenn die Kategorien entweder atomare Symbole sind oder eindeutig auf eine endliche Menge von atomaren Symbolen abgebildet werden können (in der GPSG ist letzteres der Fall). Für jede ID/LP-Grammatik G gibt es eine **stark äquivalente** kontextfreie Grammatik, d.h. eine kontextfreie Grammatik, die erstens dieselben Ketten akzeptiert und ablehnt wie G und die zweitens für jede in G ableitbare Kette dieselbe(n) Strukturbeschreibung(en) erzeugt wie G . Dieser Prozess der Übersetzung von ID/LP-Grammatiken in Typ-2-Grammatiken lässt sich auch für den allgemeinen Fall einer beliebigen ID/LP-Grammatik automatisieren. Es ist aber umgekehrt nicht für jede beliebige kontextfreie Grammatik möglich, eine stark äquivalente ID/LP-Grammatik zu konstruieren. Hier lässt sich allerdings zumindest eine schwach äquivalente ID/LP-Grammatik erzeugen (vgl. Unterkapitel 2.2).

In GPSG-Grammatiken werden verschiedene Restriktions- und Regel-Formate verwendet. Die Basis des Modells bildet eine Grammatik im so genannten **ID/LP-Format**. ID/LP-Grammatiken bestehen aus **ID-Regeln** (engl. *immediate dominance rules*) und **LP-Statements** (engl. *linear precedence statements*). Erstere ähneln in ihrer Notation den traditionellen kontextfreien Phrasenstrukturregeln. Im Unterschied zu diesen, werden die Kategorien auf der rechten Regelseite jedoch durch Kommata getrennt. Diese Anleihe bei der Notation von Mengen soll andeuten, dass die Elemente der rechten Regelseite nicht geordnet sind:

Beispiel 3.5.3

Die ID-Regel

$$\text{NP} \rightarrow \text{DET}, \text{N}$$

legt – im Gegensatz zu den Regeln einer kontextfreien Grammatik – die Reihenfolge der Symbole der rechten Regelseite (hier DET und N) nicht fest. Damit ist sie identisch mit der ID-Regel

$$\text{NP} \rightarrow \text{N}, \text{DET}$$

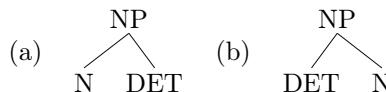


Die Definition der linearen Abfolge erfolgt durch LP-Statements. Ein LP-Statement, das festlegt, dass Nomina den Determinierern im Deutschen nachfolgen, hätte die Form:

$$\text{DET} \prec \text{N}$$

Beispiel 3.5.4

In Kombination mit der ID-Regel $\text{NP} \rightarrow \text{DET}, \text{N}$ lässt dieses LP-Statement den folgenden Baum (b) zu, nicht jedoch Baum (a):



□

Der Wirkungsbereich von LP-Statements umfasst die gesamte ID-Regelmenge. Falls es beispielsweise eine weitere ID-Regel gibt, in der N und DET vorkommen (z.B. $\text{NP} \rightarrow \text{DET}, \text{ADJ}, \text{N}$), so gilt auch dort, dass DET in allen ableitbaren Strukturbäumen vor – aber nicht notwendigerweise unmittelbar vor – N erscheint. Über die Position des Adjektivs (ADJ) wird zunächst keine Aussage gemacht. Ohne weitere LP-Statements könnte das Adjektiv dann vor, zwischen oder hinter DET und N erscheinen.

Die Kategorien einer GPSG-Grammatik sind zwar Merkmalsstrukturen, im Unterschied zu den anderen in diesem Abschnitt behandelten Formalismen sind diese jedoch nicht rekursiv, d.h. die Werte von Attributen sind grundsätzlich atomar und nicht wiederum Merkmalsstrukturen (vgl. Unterkapitel 2.3). Es gibt zwar insgesamt drei so genannte kategorienwertige Attribute, d.h. Attribute, die eine Merkmalsstruktur als Wert haben: dieser Prozess lässt sich aber nicht beliebig fortsetzen, da diese drei Attribute in ihren Werten nicht erneut vorkommen dürfen, so dass es für die Merkmalsstrukturen einer GPSG eine endliche Tiefe gibt. Aus diesem Grunde bleibt auch die Menge der Merkmalsstrukturen einer GPSG endlich – und das ist der entscheidende Punkt, der für die sehr eingeschränkte formale Mächtigkeit des Systems verantwortlich ist.

Im Kategorieninventar der GPSG wird – wie auch in vielen anderen Grammatikmodellen – zwischen Haupt- und Nebenkategorien unterschieden. Die vier Hauptkategorien sind Verb (V), Nomen (N), Adjektiv (A) und Präposition (P). Alle anderen Kategorien (z. B. Konjunktionen, Interjektionen und Gradpartikeln) sind Nebenkategorien. Die vier Grundkategorien werden durch die beiden binären Attribute N und V unterschieden, die jeweils die Werte + und – annehmen können:

	[+N]	[−N]
[+V]	A	V
[−V]	N	P

Nach dieser Definition ist z. B. das Verb durch die Merkmalsstruktur

$$\begin{bmatrix} \text{N} & - \\ \text{V} & + \end{bmatrix}$$

definiert. Für diese vier Hauptkategorien gilt, dass sie Phrasen bilden können und den so genannten lexikalischen **Kopf** (engl. *head*) der Phrase bilden. In vielen Grammatikmodellen (z. B. auch in der Rektions- und Bindungstheorie und in der HPSG) wird davon ausgegangen, dass der Kopf einer Phrase ein besonders relevantes Element ist. Zum einen werden Köpfe – im Gegensatz zu den anderen Elementen – als obligatorischer Bestandteil einer Phrase angesehen, zum anderen liegt vielen Grammatikmodellen die Annahme zugrunde, dass die besonders relevanten Merkmale einer Phrase von ihrem lexikalischen Kopf determiniert werden. So wird z. B. angenommen, dass der Kopf einer Verbalphrase das Verb (und nicht etwa eine Objekt-NP oder ein Adverb) ist. Merkmale wie Numerus und Person einer VP werden vom Verb und nicht von den etwaigen Objekten determiniert. Welche Teilkonstituente einer Phrase als ihr Kopf angesehen werden sollte, ist nicht immer unumstritten: Bei Nominalphrasen gibt es sowohl Vertreter der Annahme, dass das Nomen der lexikalische Kopf ist, als auch Linguisten, die Determinierer (DET) favorisieren und dementsprechend die Phrasenbezeichnung DP verwenden.

In der GPSG und der HPSG wird durch ein allgemeines Prinzip (die so genannte **head feature convention** bzw. das **head feature principle**) dafür Sorge getragen, dass eine Untermenge der in der Grammatik verwendeten Attribute (die unter anderem auch Person- und Numerus-Merkmale umfasst) in allen Strukturen bei einer Phrase und ihrem lexikalischen Kopf unifizierbare Werte tragen. Grammatiken, die die Idee systematisieren, dass Phrasen **Projektionen** lexikalischer Köpfe sind, werden **\overline{X} -Grammatiken (X-Bar-Grammatiken)** genannt. Das so genannte **\overline{X} -Schema (X-Bar-Schema)** besagt, dass alle Phrasen folgende Struktur haben:

$$X^n \rightarrow \dots X^m \dots \quad (m \leq n, X \in \{N, V, A, P\})$$

Nach dem \overline{X} -Schema enthalten alle Phrasen vom Grundtyp X mit einer Hierarchiestufe n (dem so genannten **Bar-Level**) mindestens eine syntaktische Kategorie des gleichen Grundtyps, deren Bar-Level m niedriger oder – in manchen Varianten des \overline{X} -Schemas – maximal gleich hoch ist. Häufig werden drei Bar-Level-Stufen unterschieden, wobei der lexikalische Kopf (z. B. N) den Bar-Level 0 hat und die Phrase (z. B. NP) den Bar-Level 2; Bar-Level 1 ist für eine Zwischenstufe reserviert, die für das Beispiel des Grundtyps N in manchen Grammatiken als „NP ohne Determinierer“ charakterisiert werden kann. Unter der Annahme dieses \overline{X} -Schemas sind Phrasenstrukturen wie

$$\text{NP} \rightarrow V \qquad \qquad \text{PP} \rightarrow V \text{ NP}$$

ausgeschlossen.

Für die Kategorien einer GPSG können Restriktionen definiert werden. Zu diesem Zweck werden so genannte **feature co-occurrence restrictions** (FCRs) verwendet. FCRs legen Restriktionen innerhalb von Merkmalsstrukturen fest, indem sie Merkmalsspezifikationen mit Hilfe logischer Junktoren verknüpfen:

$$[\text{VFORM}] \supset [+V, -N]$$

Diese feature co-occurrence restriction definiert, dass in jeder Merkmalsstruktur, die eine Spezifikation für das Attribut VFORM enthält, die Attribut-Wert-Paare V = + und N = – enthalten sein müssen. Informell ausgedrückt: Nur Verben (oder \overline{X} -Projektionen von Verben) können Spezifikationen für das Attribut VFORM tragen. Die VFORM-Spezifikation unterscheidet z. B. zwischen finiten und infiniten Verbformen und wäre deshalb in Merkmalsstrukturen für Nomina oder dergleichen fehl am Platze.

Neben den bereits dargestellten Komponenten enthält das GPSG-Modell eine Reihe weiterer Regel- bzw. Constraint-Typen, die hier aus Platzgründen nicht weiter diskutiert werden können.

In der gegenwärtigen wissenschaftlichen Diskussion spielt die GPSG keine wesentliche Rolle mehr. Dies liegt zum einen daran, dass sich Gerald Gazdar, einer der federführenden Entwickler dieses Modells, Ende der 80er Jahre anderen Themen zuwandte, zum anderen daran, dass die GPSG in der HPSG einen Nachfolge-Formalismus gefunden hat, in dem viele Ideen der GPSG aufgegriffen und weiterentwickelt wurden. Dennoch bleibt GPSG eine interessante Alternative zu anderen Formalismen (auch HPSG), da in diesem Modell die formale Beschränktheit kontextfreier Grammatiken (mit den entsprechenden Effizienzvorteilen bei Implementierungen) mit einem hohen deskriptiven Niveau kombiniert worden ist.

HPSG: Die HPSG (Head-Driven Phrase Structure Grammar, kopfgesteuerte/kopfgetriebene/kopforientierte Phrasenstrukturgrammatik, Pollard und Sag 1994) ist aus dem Kontext der GPSG entstanden, enthält aber auch diverse Elemente anderer unifikationsbasierter Grammatiken (unter anderem PATR-II, FUG und LFG) sowie Elemente der Rektions- und Bindungstheorie (Chomsky 1981). Im Gegensatz zur GPSG spielen in der HPSG Phrasenstrukturregeln keine wichtige Rolle mehr. An ihre Stelle treten einerseits generelle Prinzipien und andererseits detaillierte, hierarchisch organisierte lexikalische Strukturen. Auf eine einfache „Gleichung“ reduziert, ergibt sich die folgende Definition einer natürlichen Sprache in der HPSG:

$$S = P_1 \wedge \cdots \wedge P_n \wedge P_{n+1} \wedge \cdots \wedge P_{n+m} \wedge (L_1 \vee \cdots \vee L_k \vee R_1 \vee \cdots \vee R_l)$$

Eine Einzelsprache S erfüllt sämtliche Constraints der HPSG, wobei zwischen universellen, für alle Sprachen gültigen Constraints ($P_1 \dots P_n$) und einsprachlichen Constraints ($P_{n+1} \dots P_{n+m}$) unterschieden wird. Dabei gilt, dass alle wohlgeformten Ausdrücke aus S sämtliche Constraints *gleichzeitig* erfüllen müssen, was dadurch explizit gemacht wird, dass alle P_i eine Konjunktion bilden. Außerdem tragen die Lexikoneinträge ($L_1 \dots L_k$) und Grammatikregeln ($R_1 \dots R_l$) zur Spezifikation von S bei. Da aber nicht jeder Lexikoneintrag und jede Regel für jeden Ausdruck aus S relevant ist, bilden diese beiden Komponenten eine Disjunktion.

Im Rahmen der Darstellung des GPSG-Formalismus in vorigem Abschnitt wurden bereits einige Formalismus-Elemente genannt, die auch in der HPSG eine

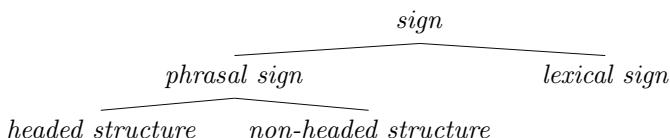
Rolle spielen. Dazu zählen z. B. der ID/LP-Formalismus, die Annahme eines \overline{X} -Schemas als zugrunde liegende Struktur für alle Phrasen und das in der HPSG besonders zentrale **head feature principle**.

Im Unterschied zu den anderen genannten unifikationsbasierten Grammatikformalismen werden die Merkmalsstrukturen in der HPSG typisiert (vgl. Unterkapitel 2.3). Der Typ einer Merkmalsstruktur kann durch die Merkmale definiert werden, die in ihr spezifiziert sein müssen. Die folgende Merkmalsstruktur ist vom allgemeinsten Typ der HPSG, dem Typ *sign*:

$$\begin{matrix} & \left[\begin{matrix} \text{PHON} & \dots \\ \text{SYNSEM} & \dots \end{matrix} \right] \\ \textit{sign} & \end{matrix}$$

Mit dem Typ *sign* sind die beiden Attribute PHON und SYNSEM verbunden, d.h., jede Merkmalsstruktur dieses Typs zeichnet sich dadurch aus, dass sie Spezifikationen für diese beiden Attribute trägt. PHON enthält Informationen über die phonologische Struktur des durch die Merkmalsstruktur beschriebenen sprachlichen Ausdrucks (diese wird aber häufig nur durch eine orthografische Repräsentation angedeutet) und SYNSEM enthält die Spezifikationen für seine syntaktischen und semantischen Eigenschaften. Dieser – sehr allgemeine – Typ wird in zwei Untertypen, *lexical-sign* und *phrasal-sign*, ausdifferenziert. Für alle Untertypen gilt grundsätzlich, dass sie (auch) alle Attribute ihres Obertyps, in diesem Falle also PHON und SYNSEM, enthalten müssen (vgl. Unterkapitel 2.3). Ein System von Typen und ihren Untertypen ergibt eine **Subsumptionshierarchie**, in der jeder Typ alle seiner Untertypen subsumiert. Man kann eine Typenhierarchie auch als Vererbungsnetzwerk auffassen, in dem Untertypen die Spezifikationen ihrer jeweiligen Obertypen erben.

Im Gegensatz zum *lexical sign* (bei lexikalischen Zeichen handelt es sich – grob gesagt – um Wörter) enthält jedes *phrasal sign* außerdem das Attribut DTRS (von engl. *daughters*), das Spezifikationen über die Struktur der Teilkonstituenten enthält. Der prototypische Fall eines phrasalen Zeichens ist eine Konstruktion, die genau einen Kopf und gegebenenfalls weitere Elemente (Komplemente, Adjunkte usw.) enthält. Solche Konstituenten werden in der HPSG dem Typ *headed-structure* zugeordnet. Die Struktur der Kopfkonstituente wird durch das Attribut HEAD-DTR gegeben, dessen Wert vom Typ *sign* ist. Komplemente (z. B. Objekte) werden unter dem Attribut COMP-DTRS aufgeführt. In der folgenden Abbildung ist der obere Bereich der HPSG-Typenhierarchie dargestellt:



Für HPSG-Beschreibungen gelten folgende Lizenziierungsbedingungen:

1. Alle Konstituenten einer Beschreibung erfüllen alle Prinzipien der HPSG.

2. Alle *phrasalen* Zeichen werden zusätzlich durch eine Regel lizenziert.
3. Alle *lexikalischen* Zeichen werden durch einen Lexikoneintrag lizenziert.

Wenn wir annehmen, dass der Satz

(3.40) *Der Hund bellt*

im Sinne der HPSG ein phrasales Zeichen (also vom Typ *phrasal-sign*) ist, dann lässt sich folgende Merkmalsstruktur konstruieren: Da jedes *phrasal-sign* auch zugleich ein *sign* ist, enthält die Strukturbeschreibung zumindest die Attribute PHON und SYNSEM, da diese für den Typ *sign* konstitutiv sind. Wie in der HPSG-Literatur üblich, enthält die Spezifikation von PHON keine detaillierte phonologische Repräsentation, sondern lediglich ein orthographisches Kürzel, wenn es eher um syntaktische oder semantische Fragestellungen geht. Es gibt allerdings auch Arbeiten, die sich explizit und schwerpunktmäßig mit der Behandlung phonologischer Fragestellungen innerhalb der HPSG auseinandersetzen (z. B. Bird (1992) und Bird und Klein (1993)).

Konstituenten vom Typ *phrasal sign* enthalten darüber hinaus das Attribut DTRS:

PHON	<i>Der Hund bellt</i>
SYNSEM	...
DTRS	...

phrasal-sign

Wenn wir nun weiter annehmen, dass es sich bei *bellt* um den Kopf der Konstruktion handelt und dass *der Hund* die Funktion eines Komplements hat, ergeben sich die folgenden Erweiterungen (wobei die Angabe des Typs *phrasal-sign* im Folgenden zur Platzersparnis weggelassen wird):

PHON	<i>Der Hund bellt</i>																
SYNSEM	...																
DTRS	<table border="0"> <tr> <td style="padding-right: 20px;">HEAD-DTR</td> <td><i>bellt</i></td> </tr> <tr> <td style="padding-right: 20px;">COMP-DTRS</td> <td> <table border="0"> <tr> <td style="padding-right: 20px;"><i>sign</i></td> <td> <table border="0"> <tr> <td style="padding-right: 20px;">PHON</td> <td><i>bellt</i></td> </tr> <tr> <td style="padding-right: 20px;">SYNSEM</td> <td>...</td> </tr> </table> </td> </tr> <tr> <td style="padding-right: 20px;"><i>sign</i></td> <td> <table border="0"> <tr> <td style="padding-right: 20px;">PHON</td> <td><i>Der Hund</i></td> </tr> <tr> <td style="padding-right: 20px;">SYNSEM</td> <td>...</td> </tr> </table> </td> </tr> </table> </td> </tr> </table> <p style="text-align: center;"><i>headed-structure</i></p>	HEAD-DTR	<i>bellt</i>	COMP-DTRS	<table border="0"> <tr> <td style="padding-right: 20px;"><i>sign</i></td> <td> <table border="0"> <tr> <td style="padding-right: 20px;">PHON</td> <td><i>bellt</i></td> </tr> <tr> <td style="padding-right: 20px;">SYNSEM</td> <td>...</td> </tr> </table> </td> </tr> <tr> <td style="padding-right: 20px;"><i>sign</i></td> <td> <table border="0"> <tr> <td style="padding-right: 20px;">PHON</td> <td><i>Der Hund</i></td> </tr> <tr> <td style="padding-right: 20px;">SYNSEM</td> <td>...</td> </tr> </table> </td> </tr> </table>	<i>sign</i>	<table border="0"> <tr> <td style="padding-right: 20px;">PHON</td> <td><i>bellt</i></td> </tr> <tr> <td style="padding-right: 20px;">SYNSEM</td> <td>...</td> </tr> </table>	PHON	<i>bellt</i>	SYNSEM	...	<i>sign</i>	<table border="0"> <tr> <td style="padding-right: 20px;">PHON</td> <td><i>Der Hund</i></td> </tr> <tr> <td style="padding-right: 20px;">SYNSEM</td> <td>...</td> </tr> </table>	PHON	<i>Der Hund</i>	SYNSEM	...
HEAD-DTR	<i>bellt</i>																
COMP-DTRS	<table border="0"> <tr> <td style="padding-right: 20px;"><i>sign</i></td> <td> <table border="0"> <tr> <td style="padding-right: 20px;">PHON</td> <td><i>bellt</i></td> </tr> <tr> <td style="padding-right: 20px;">SYNSEM</td> <td>...</td> </tr> </table> </td> </tr> <tr> <td style="padding-right: 20px;"><i>sign</i></td> <td> <table border="0"> <tr> <td style="padding-right: 20px;">PHON</td> <td><i>Der Hund</i></td> </tr> <tr> <td style="padding-right: 20px;">SYNSEM</td> <td>...</td> </tr> </table> </td> </tr> </table>	<i>sign</i>	<table border="0"> <tr> <td style="padding-right: 20px;">PHON</td> <td><i>bellt</i></td> </tr> <tr> <td style="padding-right: 20px;">SYNSEM</td> <td>...</td> </tr> </table>	PHON	<i>bellt</i>	SYNSEM	...	<i>sign</i>	<table border="0"> <tr> <td style="padding-right: 20px;">PHON</td> <td><i>Der Hund</i></td> </tr> <tr> <td style="padding-right: 20px;">SYNSEM</td> <td>...</td> </tr> </table>	PHON	<i>Der Hund</i>	SYNSEM	...				
<i>sign</i>	<table border="0"> <tr> <td style="padding-right: 20px;">PHON</td> <td><i>bellt</i></td> </tr> <tr> <td style="padding-right: 20px;">SYNSEM</td> <td>...</td> </tr> </table>	PHON	<i>bellt</i>	SYNSEM	...												
PHON	<i>bellt</i>																
SYNSEM	...																
<i>sign</i>	<table border="0"> <tr> <td style="padding-right: 20px;">PHON</td> <td><i>Der Hund</i></td> </tr> <tr> <td style="padding-right: 20px;">SYNSEM</td> <td>...</td> </tr> </table>	PHON	<i>Der Hund</i>	SYNSEM	...												
PHON	<i>Der Hund</i>																
SYNSEM	...																

Der Wert für SYNSEM ist wieder eine Merkmalsstruktur vom Typ *synsem*, die die beiden Attribute LOCAL und NON-LOCAL enthält. NON-LOCAL enthält nicht-lokale Information, also Information, die z. B. bei der Analyse von Abhängigkeiten weit voneinander entfernter Teilkonstituenten (long distance dependencies) Verwendung findet und hier nicht näher betrachtet werden soll. Innerhalb des Wertes von LOCAL gibt es eine weitere Unterteilung in das Attribut CAT (von engl. *category*), das syntaktische Informationen enthält, das Attribut CONT (von engl. *content*), das semantische Informationen enthält und das Attribut CTXT

(von engl. *context*), das pragmatische Informationen über den situativen Kontext (wie Informationen über Sprecher und Hörer, Zeitpunkt der Äußerung, etc.) bereitstellt. Hiervon interessieren uns wieder nur die syntaktischen Angaben in CAT. Dort sind die beiden wichtigen Attribute HEAD und SUBCAT zu finden. Das Attribut SYNSEM hat also die folgende interne Struktur:

$$\begin{array}{c} \text{LOCAL} \\ \text{NON-LOCAL} \end{array} \left[\begin{array}{c} \text{CAT} \quad \left[\begin{array}{c} \text{HEAD} \quad \dots \\ \text{SUBCAT} \quad \dots \end{array} \right] \\ \text{CONT} \quad \dots \\ \text{CTXT} \quad \dots \\ \dots \end{array} \right]$$

synsem

Im Wert von HEAD werden nun die Kopfinformationen repräsentiert, also die Informationen, die sich von einem Kopf auf die Phrase vererben. Der Wert des Attributs SUBCAT ist eine Liste, die Subkategorisierungsinformation – in ähnlicher Form wie bei PATR-II – enthält und weiter unter noch einmal etwas genauer erläutert wird. Der Lexikoneintrag des Wortes *bellt* stellt beispielsweise die Kopfinformation bereit, dass es sich um ein Verb im Präsens handelt. Der Wert des HEAD-Attributs ist damit vom Typ *verb* und könnte etwa das Attribut TEMP mit Wert *pres* enthalten:

$$\begin{array}{c} \text{TEMP} \quad \text{pres} \\ \text{verb} \end{array}$$

Das zentrale head feature principle lässt sich nun als folgende Bedingung formulieren: Jedes HPSG-Zeichen, das eine Kopf-Tochter (also ein Attribut HEAD-DTR) aufweist, muss den Wert des HEAD-Attributs von dieser übernehmen. Da ein solches Zeichen einen DTRS-Wert vom Typ *headed-structure* hat, lässt sich das Prinzip als folgende Bedingung formulieren:

Head Feature Principle:

$$\begin{bmatrix} \text{DTRS} & \text{headed-structure} \end{bmatrix} \implies \begin{bmatrix} \text{SYNSEM} \mid \text{LOCAL} \mid \text{CAT} \mid \text{HEAD } \boxed{1} \\ \text{DTRS} \mid \text{HEAD-DTR} \mid \text{SYNSEM} \mid \text{LOCAL} \mid \text{CAT} \mid \text{HEAD } \boxed{1} \end{bmatrix}$$

Die Schreibweise

$$\text{SYNSEM} \mid \text{LOCAL} \mid \text{CAT} \mid \text{HEAD } \boxed{1}$$

ist ein Abkürzung für

$$\left[\text{SYNSEM} \left[\text{LOCAL} \left[\text{CAT} \left[\text{HEAD } \boxed{1} \right] \right] \right] \right].$$

Der so genannte **Tag** $\boxed{1}$ zeigt dabei an, dass sich die beiden HEAD-Attribute der Phrase und der Kopf-Tochter denselben Wert teilen. Wenn zwei Attribute einer Merkmalsstruktur nicht nur auf zwei gleich etikettierte Knoten verweisen, sondern auf *denselben* Knoten, spricht man von einer **Koreferenz**. Man kann HPSG-Tags auch als Knotenvariablen auffassen. Wenn in einer Merkmalsstruktur mehrere Koreferenzen vorkommen, verwendet man Tags mit unterschiedlichen Ziffern ($\boxed{2}$, $\boxed{3}$ usw.).

Nun lässt sich das HPSG-Zeichen für Satz (3.40) wie folgt erweitern:

DTRS	$\left[\begin{array}{l} \text{PHON } \textit{Der Hund bellt} \\ \text{SYNSEM } \text{ LOCAL } \text{ CAT } \text{ HEAD } \boxed{1} \text{ } \verb verb \left[\begin{array}{l} \text{TEMP } \textit{pres} \end{array} \right] \\ \text{HEAD-DTR } \left[\begin{array}{l} \text{PHON } \textit{bellt} \\ \text{SYNSEM } \text{ LOCAL } \text{ CAT } \text{ HEAD } \boxed{1} \end{array} \right] \\ \text{COMP-DTRS } \left\langle \left[\begin{array}{l} \text{PHON } \textit{Der Hund} \\ \text{SYNSEM } \dots \end{array} \right] \right\rangle \end{array} \right]$
------	---

Damit trägt der ganze Satz *Der Hund bellt* die Kopfmerkmale des verbalen Kopfes *bellt* und ist damit im Sinne des \overline{X} -Schemas insbesondere als Projektion desselben ausgezeichnet. In dieser Merkmalsstruktur ist der Wert des Attributs COMP-DTRS eine Liste. In der HPSG-Literatur werden dafür zumeist spitze Klammern verwendet. Da es sich bei *bellt* um ein intransitives Verb handelt, enthält die Liste lediglich genau ein Element, das Subjekt. Bei Verben, die Objekte subkategorisieren, kommen aber weitere Elemente hinzu. Wie sich komplexere COMP-DTRS-Listen – z. B. als Bestandteil des Lexikoneintrags eines transitiven Verbs – auf die Struktur eines Zeichens auswirken, wird durch das **Subkategorisierungsprinzip** festgelegt:

Subcategorization Principle:

DTRS <i>headed-structure</i>	\Rightarrow	$\left[\begin{array}{l} \text{SYNSEM } \text{ LOCAL } \text{ SUBCAT } \boxed{2} \\ \text{DTRS } \left[\begin{array}{l} \text{HEAD-DTR } \text{ SYNSEM } \text{ LOCAL } \text{ SUBCAT } \textit{append}(\boxed{1}, \boxed{2}) \\ \text{COMP-DTRS } \boxed{1} \end{array} \right] \end{array} \right]$
------------------------------	---------------	--

Die Funktion *append* liefert als Wert die Verkettung zweier Listen und wird häufig auch als Infix-Operator \oplus notiert: Statt $\textit{append}(A, B)$ schreibt man dann $A \oplus B$. Durch das Subkategorisierungsprinzip, das gelegentlich auch *Valenzprinzip* genannt wird, wird für alle Zeichen vom Typ *headed-structure* festgelegt, dass das erste Element der Subcat-Liste der HEAD-DTR ($\boxed{1}$) dort lokal gesättigt wird und der Rest der Liste ($\boxed{2}$) als Valenz verbleibt. Wenn die Subcat-Liste – gegebenenfalls durch mehrfache Anwendung des Schemas – leer ist, erhält man eine vollständig gesättigte Konstituente, die keine weiteren Komplemente mehr

benötigt oder zulässt. Die Reihenfolge der Elemente in der Subcat-Liste entscheidet darüber, an welcher Position der syntaktischen Hierarchie das entsprechende Komplement erscheint. In der HPSG sind diese Elemente nach ihrer Abfolge in der **Obliqueness-Hierarchie** angeordnet. Die Unterscheidung zwischen dem *casus rectus*, dem Kasus des Subjekts, und den *obliquen* (ungeraden) Kasus geht auf stoische Grammatiker zurück. Der am wenigsten oblique Kasus ist der des Subjekts, dann folgt das direkte Objekt und schließlich das indirekte Objekt. In der HPSG-Obliqueness-Hierarchie sind ferner noch Genitive und Vergleichselemente als noch stärker oblique Konstituententypen vorgesehen.

FUG: Die von Martin Kay entwickelte Functional Unification Grammar (FUG, Kay 1979) ist der erste unifikationsbasierte Grammatikformalismus gewesen und diente den anderen in diesem Abschnitt vorgestellten Modellen als Ausgangspunkt. Zwar gibt es auch in noch früheren Arbeiten Ansätze, die die Grundidee der Unifikation durchaus schon in wesentlichen Aspekten vorwegnehmen; die Idee jedoch, die Unifikation von Merkmalsstrukturen zur Basis eines ganzen grammatischen Beschreibungssystems zu machen, geht auf Kay zurück. Dennoch ist der Formalismus Functional Unification Grammar recht schnell durch seine Nachfolger, zunächst GPSG und LFG, später vor allem durch HPSG ersetzt worden. Dies liegt vermutlich daran, dass die FUG in einem zentralen Aspekt nicht das einlöste, was unifikationsbasierte Modelle versprachen, nämlich in der Eigenschaft der **Deklarativität**. Unter Deklarativität versteht man die Eigenschaft einer Beschreibung oder eines Beschreibungsmodells, vollständig auf **prozedurale** (prozesshafte) Elemente zu verzichten. Zu solchen prozeduralen Elementen kann man z. B. die extrinsische Regelanordnung im SPE-Modell der Phonologie (vgl. Unterkapitel 3.1) zählen. Der Formalismus der FUG enthielt ein Element, das zumindest einen prozeduralen Beigeschmack hatte, nämlich die so genannte ANY-Variable. Eine ANY-Variable konnte während der Ableitung mit sämtlichen Werten unifizieren, musste aber in einer vollständigen Strukturbeschreibung gebunden sein.

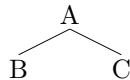
LFG: LFG (*Lexical Functional Grammar*, Lexikalisch-Funktionale Grammatik, Bresnan 1982) wurde Anfang der 80er Jahre von Joan Bresnan und Ronald Kaplan entwickelt.

Im Gegensatz zu PATR-II, GPSG oder HPSG ist LFG kein **monostrataler** Formalismus, in dem es nur eine Beschreibungsebene (ein **Stratum**) gibt: Jede Strukturbeschreibung einer LFG-Grammatik enthält zwei Komponenten, die **C-Struktur** und **F-Struktur** genannt werden. Die C-Struktur entspricht im Wesentlichen einer Beschreibung der Konstituentenstruktur durch eine kontextfreie Grammatik, verwendet aber auch Elemente des \overline{X} -Modells. Die F-Strukturen hingegen sind Merkmalsstrukturen, in denen **syntaktische Funktionen** wie Subjekt, Prädikat und Adjunkt verwendet werden.

LFG-Grammatiken haben eine höhere generative Kapazität als kontextfreie Grammatiken (so sind z. B. nicht-kontextfreie Sprachen wie $a^n b^n c^n$ mit LFG-Grammatiken definierbar). Für das Deutsche sind auf der Basis von LFG substantielle Grammatikfragmente implementiert worden.

Weitere Grammatikmodelle

Kategorialgrammatiken: Im Rahmen von Konstituentenstrukturgrammatiken werden Baumstrukturen wie



als Teil-von-Beziehungen betrachtet (B und C sind Teilstrukturen von A), die durch Produktionsregeln definiert werden können (wenn man ein A hat, kann man es durch B und C ersetzen). In der Kategorialgrammatik werden solche Strukturen weder als Teil-von-Beziehungen noch als Ersetzungsprozesse betrachtet, sondern als Funktor-Argument-Strukturen. Eine kontextfreie Regel wie

$$A \rightarrow B C$$

würde im Rahmen der Kategorialgrammatik als komplexe Kategorie repräsentiert werden. Solche komplexen Kategorien sind Funktoren, die durch Angabe ihrer Argumente und Werte notiert werden können. Wenn wir annehmen, dass B der Funktor ist, C das Argument und A der Wert der Anwendung von B auf C, dann ist B die komplexe Kategorie $\langle C, A \rangle$. Diese Schreibweise erinnert stark an die der Typen in der Typenlogik (vgl. Unterkapitel 2.1) – an der ersten Position steht das Argument, an der zweiten der Wert. Neben dieser Schreibweise, die keine Aussage über die lineare Abfolge von Funktor und Argument macht, ist auch die Notation mit / bzw. \ üblich, die diese Abfolge berücksichtigt: Wenn das Argument C links von einem Funktor mit dem Wert A steht, schreibt man $C \backslash A$, steht das Argument rechts vom Funktor, schreibt man A / C . Damit kann man aus einem Funktor A / C mit rechts daneben stehendem C als Ergebnis A ableiten; formaler geschrieben:

$$A / C \quad C \Rightarrow A \qquad C \quad C \backslash A \Rightarrow A$$

Als einfaches linguistisches Beispiel soll der Satz *Der Hund sieht die Katze* dienen. Im Lexikon werden den Wörtern folgende Kategorien zugeordnet:

Wort	Kategorie
der, die	(NP/N)
Hund, Katze	N
sieht	((NP\S)/NP)

Mit diesen Kategorien lässt sich nun die Kategorie S (für Satz) wie folgt ableiten. Ein Ableitungsschritt wird dabei durch einen waagrechten Strich unter dem Funktor und seinem Argument angedeutet.

$\begin{array}{cc} \text{der} & \text{Hund} \\ \text{NP/N} & \text{N} \\ \hline \text{NP} \end{array}$	$\begin{array}{cc} \text{sieht} & \text{die} \\ ((\text{NP}\backslash\text{S})/\text{NP}) & (\text{NP}/\text{N}) \\ \hline \text{NP}\backslash\text{S} \end{array}$	$\begin{array}{cc} \text{Katze} & \\ \text{N} & \\ \hline \text{NP} \end{array}$
--	---	--

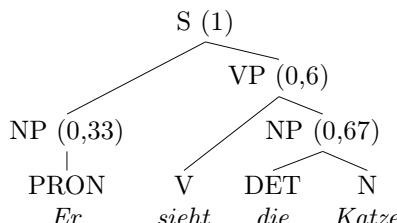
Kategorialgrammatiken gehen auf Adjukevicz (1935) zurück und wurden von Bar-Hillel (1953) für natürlichsprachliche Anwendungen weiterentwickelt. Sie liegen unter anderem der Montague-Semantik (vgl. Unterkapitel 3.6) und verschiedenen unifikationsbasierten Formalismen (Zeevat 1988, Uszkoreit 1986) zugrunde.

Probabilistische kontextfreie Grammatiken: Probabilistische Erweiterungen kontextfreier Grammatiken sind bereits Ende der 60er Jahre vorgeschlagen worden. Eine probabilistische kontextfreie Grammatik unterscheidet sich von einer traditionellen, rein symbolischen kontextfreien Grammatik dadurch, dass jede Regel als zusätzliches Element eine numerische Bewertung enthält – eine Wahrscheinlichkeit (vgl. Unterkapitel 2.4). Wie durch das erste Kolmogoroff-Axiom der Wahrscheinlichkeitstheorie festgelegt, nimmt diese numerische Bewertung einen Wert zwischen 0 und 1 an, wobei sich die Bewertungen aller Regeln mit identischer linker Regelseite zu dem Wert 1 aufsummieren. In dem folgenden Beispiel finden sich jeweils zwei Regeln, die die Symbole NP und VP expandieren, eine weitere Regel expandiert das Symbol S:

S	→	NP VP	1
NP	→	DET N	0,67
NP	→	PRON	0,33
VP	→	V	0,4
VP	→	V NP	0,6

Durch die Regelgewichte wird definiert, dass NPs, die aus DET und N bestehen, doppelt so wahrscheinlich sind wie NPs, die aus einem Pronomen aufgebaut sind. Da in dieser Grammatik keine alternativen Regeln für die Expansion des Startsymbols S vorgesehen sind, erhält die Regel $S \rightarrow NP\ VP$ das maximale Gewicht 1.

Die Wahrscheinlichkeit einer Ableitung relativ zu einer gegebenen Kette von terminalen Symbolen wird als Produkt der Regelbewertungen aller Regeln definiert, die in der Ableitung vorkommen. Bei diesem Bewertungsprinzip wird also die linke Regelseite als gegeben vorausgesetzt und die verschiedenen Expansionen eines Symbols werden relativ zu diesem gegebenen Symbol geschätzt, weshalb man von der **Expansionswahrscheinlichkeit** spricht. Für einen Satz wie *Er sieht die Katze* ergibt sich die unten stehende Ableitung. Die Expansionswahrscheinlichkeit eines Symbols mit der entsprechenden Regel ist dabei in Klammern hinter dem Symbol notiert. Die Ersetzung der präterminalen Symbole durch Terminalsymbole ist dabei nicht bewertet:



Das Produkt der vorkommenden Regelgewichte ergibt nun eine Gesamtwahrscheinlichkeit von $1 \cdot 0,6 \cdot 0,33 \cdot 0,67 = 0,13266$ für diese Ableitung.

Der beschriebene Bewertungsmechanismus ist allerdings nicht der einzige mögliche. In Manning und Carpenter (1997) wird z. B. ein alternativer Bewertungsmaßstab vorgeschlagen, der nicht an einer Top-down-Verarbeitungsstrategie (wie wahrscheinlich ist die Anwendung einer Regel, gegeben das Symbol der linken Regelseite) orientiert ist, sondern an einer Left-Corner-Verarbeitung (wie wahrscheinlich ist die Anwendung einer Regel, gegeben das erste Symbol der rechten Regelseite).

Eine wichtige Eigenschaft des oben angegebenen, klassischen Bewertungsschemas für kontextfreie Grammatiken ist, dass es – ebenso wie die Grammatik selbst – kontextfrei ist: Wie es unabhängig vom gegebenen Ableitungskontext möglich ist, ein Symbol durch eine beliebige Expansion dieses Symbols zu ersetzen, ist auch die probabilistische Bewertung der jeweils gewählten Regel von diesem Kontext gänzlich unabhängig. Wenn man diese Unabhängigkeitsannahme als eine linguistische Hypothese auffasst, dann besagt diese Hypothese beispielsweise, dass die verschiedenen Ersetzungsmöglichkeiten einer Nominalphrase in allen Kontexten dieselbe Vorkommenswahrscheinlichkeit haben. Das heißt, dass etwa reflexive NPs oder NPs, die aus einem Eigennamen bestehen, in der Subjektsposition ebenso häufig vorkommen wie als Genitivattribut oder als direktes Objekt. Dass diese Hypothese für das Englische falsch ist, zeigen die Untersuchungen, die Manning und Carpenter anhand der Penn-Treebank vorgenommen haben (Manning und Carpenter 1997). Für das Deutsche lassen sich leicht ähnliche Resultate finden: Die Wahrscheinlichkeit einer reflexiven Nominalphrase in Subjektfunktion ist z. B. 0, da diese Konstruktion im Deutschen nicht möglich ist.

Ein anderes grundsätzliches Problem probabilistischer kontextfreier Grammatiken besteht in ihrer Tendenz, kürzere Ableitungen gegenüber längeren zu bevorzugen, selbst wenn die Regelwahrscheinlichkeiten in der längeren Ableitung durchschnittlich höher liegen. Um dieses Problem zu vermeiden, werden Verfahren wie Normalisierung durch Bildung des geometrischen Mittels verwendet. Mit einer solchen Normalisierung verlässt man das Terrain der Wahrscheinlichkeitstheorie; streng genommen handelt es sich bei Systemen, in denen solche Verfahren angewandt werden, also nicht um probabilistische Grammatiken.

Die Verwendung von Statistik beim Parsing hat jedoch nicht die Ermittlung von Satzwahrscheinlichkeiten zum Zweck, sondern z. B. auch die korrekte Disambiguierung von Sätzen. Ein klassisches Beispiel hierfür liefert der Satz

(3.41) *Der Mann sieht die Frau mit dem Fernrohr,*

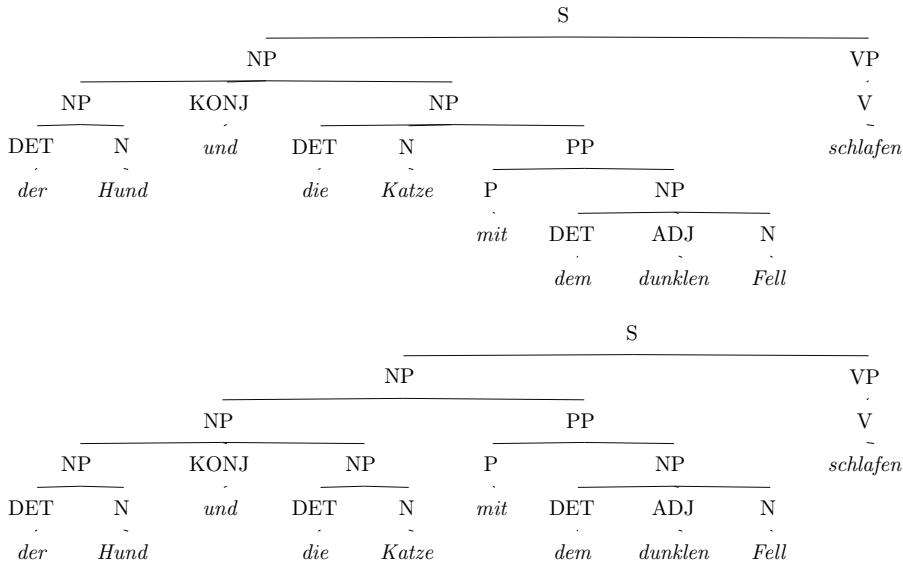
bei dem die Mehrdeutigkeit darin besteht, dass die Präpositionalphrase *mit dem Fernrohr* das Instrument sein kann, mit dem der Mann die Frau betrachtet, oder aber ein Attribut der Frau darstellt. Der Unterschied auf syntaktischer Seite liegt in der **PP-Anbindung** (engl. *PP-attachment*): Ist die PP ein Attribut der VP *sieht die Frau*, so erhält man die erste Lesart, ist sie ein Attribut der NP *die Frau*, erhält man letztere. Durch die Festlegung von Regelwahrscheinlichkeiten

lässt sich eine der beiden Lesarten bevorzugen. Enthält die Grammatik unter anderem beispielsweise die beiden Regeln

$$\begin{array}{lcl} \text{NP} & \rightarrow & \text{NP PP} \quad 0,4 \\ \text{VP} & \rightarrow & \text{VP PP} \quad 0,6 \end{array}$$

so wird die Ableitung, in der die PP an die VP angebunden wurde, höher bewertet – damit kann Satz (3.41) zugunsten dieser Lesart disambiguiert werden.

Da sich die Gesamtwahrscheinlichkeit einer Ableitung aus dem Produkt der einzelnen Regelwahrscheinlichkeiten ergibt, werden etwaige Reihenfolgeunterschiede durch die Kommutativität der Multiplikation wieder eingeebnet. So ergibt sich für die folgende PP-Anbindungsambiguität aus einer – wie auch immer gestalteten – Bewertung der Regeln im Sinne einer probabilistischen kontextfreien Grammatik keine Entscheidung zugunsten einer Lesart, weil sich die beiden Ableitungen nicht hinsichtlich der in ihnen vorkommenden Regeln unterscheiden, sondern nur hinsichtlich deren Reihenfolge:



Dass die Wahrscheinlichkeit einer PP-Anbindung in den allermeisten Fällen nicht nur strukturell determiniert ist, sondern auch stark vom lexikalischen Material abhängt, lässt sich mit folgenden Beispielsätzen illustrieren:

1. Er beobachtet das Mädchen mit dem Fernrohr.
2. Er kennt das Mädchen mit dem Fernrohr.

Die starke Tendenz zur adverbialen Lesart der PP in dem ersten Beispielsatz beruht darauf, dass ein *Fernrohr* ein geeignetes Instrument für die durch das Verb *beobachten* denotierte Handlung ist. Es liegt also an den semantischen Eigenschaften der beteiligten Lexeme, dass wir in diesem Beispielsatz der adverbialen

Lesart stärker zuneigen als in dem zweiten Satz - und nicht an deren syntaktisch-strukturellen Eigenschaften.

Auf die Verwendung statistischer Informationen beim Parsing gehen wir in Abschnitt 3.5.2 genauer ein.

3.5.2 Parsing

Das Ziel dieses Abschnitts ist es, einen Überblick über die Grundprobleme des Parsings und einige der wichtigsten Parsing-Methoden zu geben. Neben einem einfachen **Backtracking-Parser** stellen wir den **Earley-Algorithmus** vor. Abschließend geben wir einen Überblick über **probabilistische Parsing-Modelle**.

Grundlagen

Der Ausdruck **Parsing** leitet sich aus dem lateinischen *partes orationis*, den Teilen der Rede (d.h. den Wortarten), her. Im weitesten Sinne umfasst Parsing alle Formen der (automatischen) Analyse sprachlicher Ausdrücke, aber gerade für diesen ursprünglichen Sinn (grammatische Analyse als Feststellung der Wortarten) hat sich in den vergangenen Jahren der Begriff **Tagging** (Wortarten-Tagging, Part-of-speech tagging) etabliert; unter Parsing versteht man heute eher solche Analyseprozesse, die substantiell über das bloße Annotieren eines Textes mit Wortarten hinausgehen und die die grammatische Struktur einer Äußerung aufdecken. Die Grenzen sind jedoch inzwischen fließend, so wird der Begriff Tagging z.B. auch für die Annotation von Texten mit syntaktischen Funktionen wie Subjekt und Objekt verwendet.

Der Standardanwendungsbereich von Parsing-Methoden in der Computerlinguistik ist die (Satz-)Syntax und in diesem Sinne wird der Begriff Parsing im Folgenden auch verwendet. Es gibt aber auch zahlreiche Anwendungen in anderen Bereichen, z.B. prosodisches und phonologisches Parsing, morphologisches Parsing, semantisches Parsing usw. Außerhalb der Computerlinguistik findet der Begriff Parsing auch in der Informatik (Compilerbau) und in der Psycholinguistik Verwendung (dort als Bezeichnung für den kognitiven Prozess der syntaktischen Analyse beim Menschen).

Ebenso vielfältig wie die Phänomenbereiche, in denen Parsingmethoden zum Einsatz kommen, sind die Grammatikformalismen, für die Parsing-Algorithmen neu entwickelt oder angepasst wurden. Im Zentrum stehen jedoch nach wie vor und immer wieder Parsingalgorithmen für kontextfreie Grammatiken. Auch bei Formalismen, bei denen das kontextfreie Skelett relativ schwach ausgeprägt ist, z.B. in HPSG-orientierten Grammatiken, wird für Zwecke des Parsings zumeist auf Grundalgorithmen für kontextfreie Grammatiken zurückgegriffen.

Da natürlichsprachliche Ausdrücke in aller Regel zumindest partiell ambig sind, besteht ein syntaktischer Analyseprozess zu einem nicht unwesentlichen Anteil aus Suchprozessen. Ein solcher Suchprozess lässt sich graphentheoretisch als Durchlaufen eines **Suchraums** (Suchgraphen) charakterisieren, der einen Startzustand Z_0 hat und einen oder mehrere Endzustände E_1, E_2, \dots, E_n (vgl. Unterkapitel 2.3). Das Durchlaufen des Suchraums beginnt am Ausgangs- oder

Startzustand Z_0 , der sich als Tripel der Form

$$\langle \alpha, Kat, Struktur \rangle$$

definieren lässt. Dabei bezeichnet

- α den zu analysierenden Ausdruck (im Startzustand ist dies die vollständige Eingabekette),
- Kat die Zielkategorie (das Startsymbol der Grammatik) und
- $Struktur$ eine partielle (im Startzustand leere) Strukturbeschreibung.

Ein Zielzustand ist dadurch charakterisiert, dass er eine vollständige Strukturbeschreibung für die Eingabekette α enthält, deren Mutterknoten mit der Zielkategorie Kat etikettiert ist. Das Durchlaufen des Suchraums besteht darin, dass durch Anwendung einer Operation O_i aus einer gegebenen Menge $\Omega = \{O_1, \dots, O_n\}$ von einem Zustand Z_j in einen Zustand Z_k übergegangen wird. Eine Folge von Übergängen ist ein erfolgreicher Suchprozess, wenn sie mit dem Startzustand beginnt und mit einem Endzustand endet. Dies soll anhand eines Beispiels illustriert werden.

Beispiel 3.5.5

Gegeben sei die kontextfreie Grammatik aus Beispiel 3.5.1, analysiert werden soll der Satz:

(3.42) *Der Hund bellt*

Dann ist der Startzustand des Suchprozesses

$$\langle \text{Der Hund bellt}, S, [] \rangle$$

und der Endzustand

$$\langle [], S, S[NP[DET[Der] N[Hund]] VP[V[bellt]]] \rangle$$

Die Strukturbeschreibung wurde hier als Baum in der platzsparenden Klammerdarstellung angegeben. In dieser Darstellung bedeutet $N[Hund]$ beispielsweise, dass *Hund* ein Endknoten ist, dessen Mutterknoten mit N bezeichnet ist. Die Darstellung $NP[DET[\dots]N[\dots]]$ steht für einen Teilbaum, in dem der Knoten NP die Knoten DET und N als Töchter besitzt (vgl. Unterkapitel 2.3). \square

Ein Zielzustand kann über verschiedene Wege durch den Suchraum erreicht werden. Diese verschiedenen Wege lassen sich nach folgenden Kriterien klassifizieren:

Verarbeitungsrichtung: Beginnt die Analyse des Satzes am Satzanfang und schreitet sie Wort für Wort (**inkrementell**) von links nach rechts fort, spricht man von einer (unidirektionalen) **Links-rechts-Verarbeitung**. Diese Verarbeitungsrichtung ist der Standard, weil sie der Reihenfolge entspricht, in der Äußerungen – gemessen an den Schriftsystemen vieler europäischer Sprachen –

produziert werden. Im Prinzip sind jedoch auch ganz andere Verarbeitungsrichtungen möglich. So wird z. B. beim so genannten **Head-Corner-Parsing** (vgl. Bouma und van Noord 1993) von den lexikalischen Köpfen eines Satzes (d. h. in der Regel vom finiten Verb) ausgegangen und anschließend (bidirektional) nach links und rechts weiterverarbeitet.

Analyserichtung: Die beiden wichtigsten Analyserichtungen sind die **Top-down-Verarbeitung** und die **Bottom-up-Verarbeitung**. Bei der Top-down-Verarbeitung wird beim Aufbau der Struktur zunächst von der Start-Kategorie S ausgegangen und durch Expansion (d.h. durch die Ersetzung einer linken Regelseite durch die rechte) fortgefahren, bis die schrittweise Expansion zu den terminalen Symbolen führt. Diese Analyserichtung wird in Abbildung 3.40 anhand der Grammatik aus Beispiel 3.5.1 illustriert. Sie entspricht der Reihenfolge in einer Linkstableitung (vgl. Unterkapitel 2.2). Umgekehrt wird bei einer Bottom-up-Verarbeitung von der Eingabekette selbst ausgegangen, und die Anwendung der Regeln erfolgt als **Reduktion**, d.h. die Elemente der rechten Regelseite werden durch das Symbol der linken Regelseite ersetzt, bis eine vollständige Reduktion der gesamten Eingabekette auf das Startsymbol S erzielt wurde (die Darstellung eines bottom-up, links-rechts erfolgenden Strukturaufbaus findet sich in Abb. 3.41). Neben diesen beiden Grund-Analyse-Richtungen gibt es Mischformen, die Top-down- und Bottom-up-Elemente miteinander verbinden. Die wichtigste Variante ist die so genannte **Left-Corner-Strategie**, bei der von der linken Ecke einer Regel ausgegangen wird, d.h. dem ersten Element der rechten Regelseite.

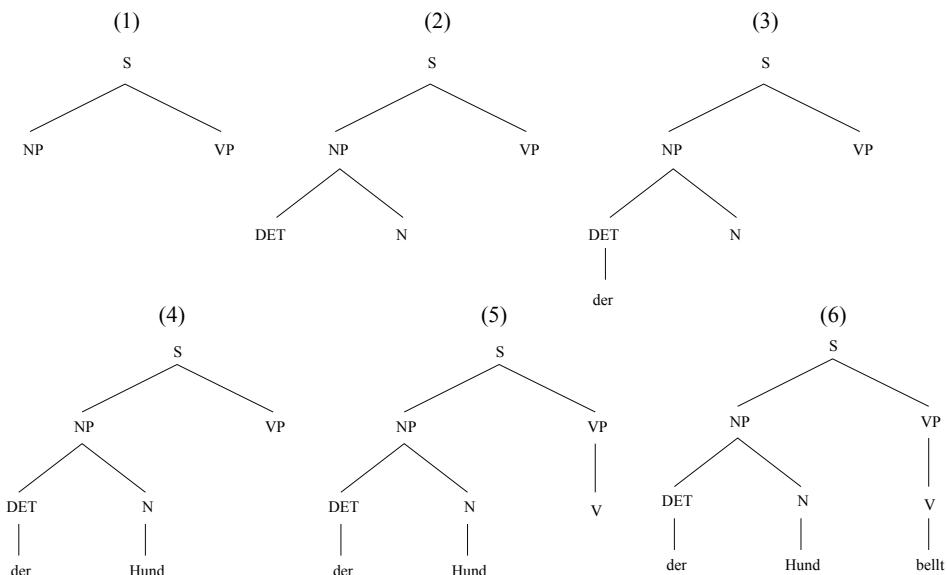


Abbildung 3.40: Strukturaufbau top-down, links-rechts

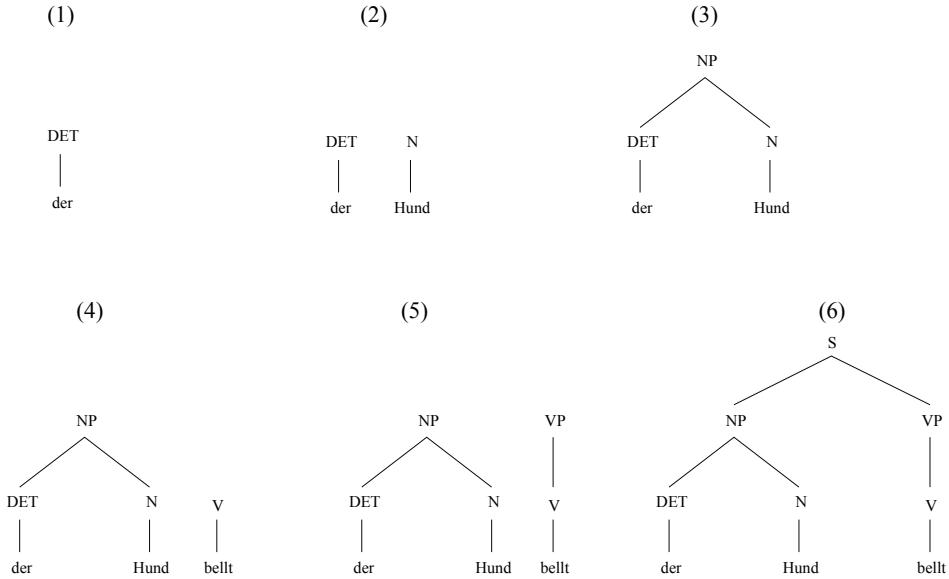


Abbildung 3.41: Strukturaufbau bottom-up, links-rechts

Suchstrategie: Die beiden wichtigsten Suchstrategien sind die **Tiefensuche** (engl. *depth-first search*) und die **Breitensuche** (engl. *breadth-first search*). Bei der Tiefensuche wird ein einmal eingeschlagener Pfad im Suchraum solange weiterverfolgt, bis er nicht mehr fortgesetzt werden kann; erst dann werden alternative Pfade durchlaufen. Bei einer Breitensuche werden alle Suchraumpfade um einen Schritt fortgesetzt, bevor einer von ihnen in der Tiefe weiterverfolgt wird.

Beispiel 3.5.6

Gegeben sei eine kontextfreie Grammatik mit folgenden Regeln:

1. $S \rightarrow NP\ VP$
2. $S \rightarrow S\ KONJ\ S$
3. $NP \rightarrow DET\ N$

Eine top-down und links-rechts verarbeitende Breitensuche würde zunächst die beiden S-expandierenden Regeln verwenden und erst danach die mit der ersten Regel erzeugte NP weiter expandieren. Eine top-down und links-rechts verarbeitende Tiefensuche, hingegen, würde nach Anwendung der Regel $S \rightarrow NP\ VP$ sofort damit fortfahren, die NP weiter zu expandieren. Wenn die beiden S-expandierenden Regeln für eine Tiefensuche in umgekehrter Reihenfolge vorliegen, terminiert die Tiefensuche wegen der **Linksrekursivität** der Regel $S \rightarrow S\ KONJ\ S$ nicht. Zur Erinnerung: Eine Regel heißt **linksrekursiv**, wenn das Symbol der linken Regelseite mit dem ersten Symbol der rechten Regelseite identisch ist. \square

Tiefensuche erfüllt deshalb das Kriterium der **Vollständigkeit** nicht, das fordert, dass eine Suchstrategie in jedem Fall in endlicher Zeit eine Lösung findet, sofern eine existiert. Man kann Tiefensuche allerdings so modifizieren, dass Vollständigkeit sichergestellt ist. Dazu verwendet man ein Tiefenlimit, das initial bei 1 liegt und iterativ erhöht wird, wenn die Suche bis zu einem gegebenen Limit nicht erfolgreich war. Der Vorteil, den Tiefensuche gegenüber Breitensuche hat, nämlich ein erheblich geringerer Speicherplatzbedarf, geht bei dieser Modifikation allerdings weitgehend verloren. Diese Variante der Tiefensuche wird **iterative Tiefensuche** (engl. iterative deepening) genannt.

Wenn für die Pfade des Suchraums ein Bewertungsmechanismus (eine Heuristik) vorliegt, können die Pfade in der Reihenfolge ihrer Güte durchlaufen werden; diese Strategie wird als **Best-first-Suche** bezeichnet. Eine Bewertung der Pfade eines Suchraums kann jedoch auch dafür verwendet werden, den Suchraum auf solche Pfade einzuzgrenzen, die gemäß der Bewertung besonders erfolgversprechend sind. Suchverfahren, die nur einen Teil des Suchraums betrachten, werden **lokale Suchverfahren** genannt. Wird eine solche Beschränkung auf erfolgversprechende Pfade verwendet, liegt eine **Beam-Suche** vor. Bei der Beamsuche werden schlecht bewertete Pfade verworfen (engl. *Pruning*) oder – bei der Variante der **stochastischen Beamsuche** – mit geringerer Wahrscheinlichkeit ausgewählt. Wenn Pruning verwendet wird, ist es nicht auszuschließen, dass auch die richtige Lösung unter ungünstigen Umständen verworfen wird und der Suchprozess gar keine oder nicht alle korrekten Lösungen für das Suchproblem findet. Unter bestimmten Umständen ist Beam-Suche jedoch unumgänglich, z. B. wenn der Suchraum für eine vollständige Suche zu groß ist. Eine wichtige Rolle spielt eine adäquate Einschränkung des Suchraums vor allem auch dann, wenn aufgrund der Problemstellung bereits bekannt ist, dass der Suchraum auch unerwünschte oder fehlerhafte Lösungen enthält. Diese Situation ist im Bereich Parsing vor allem dann gegeben, wenn anzunehmen ist, dass die Grammatik ungenau ist (das gilt vor allem für Grammatiken, die automatisch aus einer Treebank extrahiert wurden) oder wenn bekannt ist, dass die Eingabe des Parsers unbestimmt oder partiell fehlerhaft ist. Diese Situation liegt z. B. dann vor, wenn die Eingabe des Parsers nicht aus einer Kette von diskreten Symbolen besteht, sondern aus einem Worthypothesen-Graphen, wie er als Ausgabe einer Spracherkennungs- oder Handschrifterkennungskomponente erzeugt wird.

Probleme

Die enorme Schwierigkeit des Problems, ein praxistaugliches syntaktisches Analyseprogramm für eine natürliche Sprache zu entwickeln, hat sehr viele verschiedene Ursachen. Die folgenden drei Teilprobleme sind Gegenstand dieses Abschnitts:

1. Wie können syntaktische Ambiguitäten aufgelöst werden?
2. Wie kann eine Grammatik entwickelt werden, die einen sehr großen Ausschnitt einer natürlichen Sprache abdeckt?
3. Wie kann ein effizientes syntaktisches Analysesystem implementiert werden?

Diese drei Probleme sind eng miteinander verbunden, und sie sind allesamt entweder irrelevant oder bekanntmaßen lösbar, wenn es lediglich um die exemplarische Analyse sehr kleiner Satzmengen aus einer beschränkten Domäne geht. Sobald man jedoch den Versuch unternehmen will, einen syntaktischen Parser für Anwendungen wie allgemeine, nicht thematisch eingegrenzte, maschinelle Übersetzung zu entwickeln, stellen sich diese Probleme. Für Anwendungsfelder dieser Größenordnung existieren bislang weder Grammatiken, die hinreichend große Sprachfragmente präzise abdecken, noch existieren verlässliche Disambiguierungsstrategien oder hinreichend effiziente Parsingsysteme.

Ambiguität: Die Anzahl der syntaktischen Lesarten von ganz gewöhnlichen Sätzen, die von größeren Parsing-Systemen geliefert wird, ist zumeist erheblich höher als der Ambiguitätsgrad, den selbst geschulte Syntaktiker auf den ersten Blick erkennen. Als klassisches Beispiel für diesen Effekt wird in der Literatur gerne der Satz

(3.43) *Time flies like an arrow.*

angeführt, dessen Lesarten wie

(3.44) *Zeitfliegen mögen einen Pfeil.*

(3.45) *Bestimme die Geschwindigkeit von Fliegen so, wie es ein Pfeil tut!*

zwar (ohne einen entsprechenden Kontext) völlig abwegig, nichtsdestotrotz aber grammatisch sind und deshalb von einem syntaktischen Parser auch erkannt werden.

Nicht immer sind die von Parsing-Systemen gefundenen Lesarten so anekdotenreicher wie im Falle von Satz (3.43). Oftmals sind die gefundenen Lesarten im Prinzip – mit oder ohne Kontext – verständlich und semantisch interpretierbar. So liefert z. B. das Parsing-System des PARGRAM-Projekts (Kuhn und Rohrer 1997) für den Satz:

(3.46) *Hinter dem Betrug werden die gleichen Täter vermutet, die während der vergangenen Tage in Griechenland gefälschte Banknoten in Umlauf brachten.*

insgesamt 92 Lesarten, von denen eine beträchtliche Anzahl durchaus nicht semantisch abwegig sein dürfte. Ein Ambiguitätsgrad von 92 ist bei Parsing-Systemen, die auf der Basis von (rein syntaktischen) und somit domänenunabhängigen Grammatiken arbeiten, alles andere als ein extremer Sonderfall. Über Ambiguitätsgrade in der Größenordnung von einer Million und mehr ist in der Literatur bereits mehrfach berichtet worden und in Block (1995) wird sogar eine Lesartenanzahl von $6.4875e+22$ genannt (die sich allerdings auf eine Grammatik für spontansprachliche Äußerungen bezieht, die laut Block „keine harten Restriktionen über morphologische Kongruenz und Subkategorisierungen enthält“ und somit natürlich einen ganz anderen Status hat als eine Grammatik, die sich an schriftsprachlichen Standards orientiert).

Für den Umgang mit Mehrdeutigkeiten in sprachverarbeitenden Systemen sind verschiedene Strategien entwickelt worden. Eine dieser Strategien, die in den vergangenen Jahren relativ ausführlich untersucht worden ist, ist die frühzeitige enge Kopplung von syntaktischen und anderen Analysekomponenten, die semantische, pragmatische, kontextuelle und – im Falle gesprochener Sprache – prosodische Informationen miteinbeziehen. Eine solche enge und frühzeitige Kopplung wird z. B. im VerbMobil-System verwendet (Wahlster 1993). In diesem Projekt geht es vor allem auch um die Integration von akustischer Spracherkennung und linguistischer Analyse. Gerade im Kontext solcher Systeme sind Inkrementalität und möglichst frühzeitige Integration von Analyseresultaten verschiedener Komponenten besonders relevant, da die Eingabe nicht als Kette von diskreten Symbolen vorliegt.

Solche **integrativen Architekturen** sind ein Gegenentwurf zu dem klassischen **sequentiellen Architekturmodell**, in dem die einzelnen Komponenten wie morphologische Analyse, syntaktische Analyse, semantische Analyse usw. nicht miteinander interagieren, sondern das vollständige Analyseresultat der jeweils vorgeschalteten Komponente als Eingabe nehmen und anschließend das eigene Analyseresultat vollständig an die nachfolgende Komponente weitergeben. Diese sequentielle Architektur ist wohl nach wie vor die Standardarchitektur. Sie wurde in den meisten größeren Sprachverarbeitungssystemen verwendet, wie z. B. in der Core Language Engine (Alshawi 1992).

Theoretisch kann eine solche enge Kopplung die Anzahl der Lesarten beschränken, die ein syntaktischer Parser produziert, und somit zu einer Effizienz des Gesamtsystems beitragen. Mit einer integrativen Architektur ist jedoch auch ein höherer kommunikativer Aufwand verbunden, und es ist noch nicht völlig klar, ob und wie sich dieser Aufwand zu den erzielbaren Effizienzoptimierungen durch die Ambiguitätsreduktion verhält, welche Inkrementgröße (Wörter, abgeschlossene Phrasen, Teilsätze) sinnvoll ist und welche nicht-syntaktischen Informationen geeignet sind, um syntaktisch definierte Suchräume zu begrenzen. Zudem zeigen systematische Messungen, dass die gegenwärtigen Methoden zur Implementierung von Sprachverarbeitungssystemen auf echt-parallelen Hardware-Plattformen nicht die Ergebnisse erzielen können, die theoretisch erwartbar wären. Da integrative Architekturen ihre konzeptionellen Vorteile vermutlich erst dann in vollem Umfang in entsprechende Leistungssteigerungen umsetzen können, wenn es gelingt, effiziente echt-parallele Analysealgorithmen für Sprachverarbeitungsaufgaben zu implementieren, kann zum gegenwärtigen Zeitpunkt noch nicht entschieden werden, wie sich das Konzept der integrativen Architektur in der Praxis auswirken wird.

Zudem gibt es Fälle, bei denen eine Disambiguierung schlicht nicht möglich ist. Für die Eingabekette

- (3.47) *Es klappte gut weil Maria die Freundin von Anna aus Osnabrück mit dem Auto von Petra aus Bielefeld abgeholt hat.*

produziert der Gepard-Parser (vgl. Langer 2001) insgesamt 1732 Strukturbeschreibungen. Für diesen Analyseprozess benötigt das System etwa 20 Sekunden. Für einen menschlichen Rezipienten ist dieser Satz weitgehend unverständlich;

der kommunikative Gehalt dieses Satzes beschränkt sich darauf, dass irgendeine Frau (wahrscheinlich namens Maria) eine andere Frau mit einem Auto abgeholt hat. Auch semantische Plausibilitätsüberlegungen, allgemeines Weltwissen oder pragmatische Annahmen können dieses Ambiguitätsproblem nicht lösen. Auch ist die Frage der Architektur hier unerheblich: Die Komplexität lässt sich durch eine sequentielle Architektur ebensowenig reduzieren wie durch eine integrative. In diesem (erfundenen) Satz kommen mehrere Probleme des Deutschen zusammen:

1. Durch die fehlende Kasusmarkierung bei den Eigennamen kann deren syntaktische Funktion nicht bestimmt werden;
2. der Satz enthält mehrere PP-Anbindungsambiguitäten;
3. durch die Verbendstellung werden einige wichtige Informationen, die die Rolle der vorangehenden Nominal- und Präpositionalphrasen klären könnten (z. B. ob das Verb Präpositionalkomplemente zulässt, ob es sich um einen aktivischen oder passivischen Satz handelt usw.), erst spät gegeben.

Nimmt man nun an, dass der Satz im Passiv steht, ist das Ambiguitätsproblem um fast 90% reduziert: „Nur“ noch 192 Strukturbeschreibungen liefert der Gepard-Parser für den Satz

(3.48) *Es klappte gut weil Maria die Freundin von Anna aus Osnabrück mit dem Auto von Petra aus Bielefeld abgeholt wurde*.

Für die Analyse dieses Satzes benötigt der Parser etwas weniger als 7 Sekunden, d.h. etwa ein Drittel der Zeit, die zum Parsen der aktivischen Variante erforderlich war.

Für Sätze mit solchen Merkmalen ist also eine inkrementelle Links-rechts-Verarbeitung ausgesprochen ineffizient und die Verwendung nicht-syntaktischer Zusatzinformationen nicht effektiv; jedoch kann ein frühzeitiger **Lookahead**, der die Information des finiten Verbs berücksichtigt, den Suchraum massiv eingrenzen. Unter Lookahead (dt. *Vorausschau*) versteht man Techniken, bei denen Entscheidungen über den weiteren Verlauf eines Parsingprozesses – ohne Suchprozesse im engeren Sinn – allein durch Umsetzung von Informationen getroffen werden, die bei Betrachtung der nächsten Wörter der Eingabekette unmittelbar vorliegen. Das Resümee, das man aus den Beispielen dieses Abschnitts ziehen kann, ist das folgende:

1. Das Problem der syntaktischen Ambiguität ist groß und lässt sich bei umfassenden domänenunabhängigen Grammatiken nicht ausschließen.
2. Auch bei massiv mehrdeutigen Sätzen kann nicht immer eine Ambiguitätsreduktion erzielt werden, indem die merkwürdigen Lesarten frühzeitig durch einfache semantische oder pragmatische Constraints ausgeschlossen werden.

3. Die Ursachen für Ambiguität sind so vielschichtig und oftmals so konstruktionsspezifisch, dass es keine allgemeine Lösung geben kann. Stattdessen muss der Versuch unternommen werden, möglichst viele und möglichst verschiedene Strategien zur Ambiguitätsreduktion einzusetzen; solche Strategien können semantische und pragmatische Wohlgeformtheitsbedingungen einschließen, aber auch probabilistische Modelle, flexible Lookahead-Strategien usw.

Abdeckung: Von relativ vielen für das Deutsche entwickelten Parsing-Systemen ist nicht bekannt, wie groß der Anteil der Sätze in realen Texten oder Diskursen ist, der durch das jeweilige System abgedeckt wird. Dies gilt auch für einige der größeren und relativ ausführlich dokumentierten Systeme.

In Backofen et al. (1996) findet sich ein Überblick über die Abdeckung verschiedener implementierter Grammatiken, in dem zumindest die groben Phänomenbereiche angegeben sind, die von der jeweiligen Grammatik abgedeckt werden. In dieser Studie sind neben den in Saarbrücken entwickelten TAG- und HPSG-Fragmenten auch PARGRAM und LS-GRAM zu finden. Außerdem enthält die Arbeit auch Informationen über die existierenden Grammatikfragmente für andere europäische Sprachen. Zum Thema Evaluierung im Allgemeinen sei auf Kapitel 6 verwiesen.

Dass die Resultate zur Abdeckung von Parsing-Systemen nicht überprüft oder zumindest nicht publiziert werden, ist kein Phänomen, dass sich auf die Systeme beschränkt, die für die Analyse des Deutschen entwickelt wurden. Auch in den (ohnehin relativ wenigen) publizierten Darstellungen von tatsächlich implementierten Sprachverarbeitungssystemen für andere Sprachen fehlen nicht selten Angaben über den Anteil der erfolgreich analysierten Sprachdaten. So ist z.B. in der einflussreichen und vielzitierten Monographie über das Core Language Engine (Alshawi 1992) keine Angabe zu finden, die eine Einschätzung der tatsächlichen Performanz des Systems bei Realdaten erlaubt. Entsprechende Zahlen liegen zwar für diverse Parsingsysteme vor, deren Grammatiken vollautomatisch oder semiautomatisch aus der Penn-Treebank (Marcus et al. 1993) extrahiert wurden, es handelt sich bei diesen Grammatiken aber natürlich um Beschreibungen, die zum einen an Daten aus dem (thematisch begrenzten) Korpus getestet werden, aus dem die Grammatik auch erzeugt wurde, und zum anderen um Grammatiken, die übergenerieren, d.h. um Grammatiken, die keine systematische Unterscheidung zwischen wohlgeformter und nicht-wohlgeformter Eingabekette vornehmen können.

Die Entwicklung größerer Grammatikfragmente ist eine so komplexe Aufgabe, dass sie oftmals nur über mehrere Jahre hinweg von einer größeren Gruppe von Entwicklern bewerkstelligt werden kann. In den vergangenen Jahren wurden Methoden untersucht, wie dieser Entwicklungsprozess möglichst effektiv durch entsprechende Software-Werkzeuge unterstützt werden kann. Eine besonders wichtige Rolle spielen dabei die folgenden Komponenten:

1. Parser und Generatoren, die die automatische und halbautomatische Erzeugung und Überprüfung von Beispielsätzen erlauben,

2. Visualisierungswerzeuge, die die komplexen Strukturen größerer Grammatikfragmente visualisieren oder Verarbeitungsprozesse (z. B. die Analyse eines Beispielsatzes) graphisch aufbereiten und
3. die strukturierte Verwaltung von Testsatzmengen.

Einen guten Überblick über Kriterien für Grammatikentwicklungsumgebungen bietet Volk (1995).

Effizienz: Für algorithmische Lösungen hochkomplexer Probleme ist die Effizienz des jeweiligen Verfahrens von besonderer Bedeutung. So wäre z. B. ein Parser für eine natürliche Sprache selbst dann, wenn er alle anderen Adäquatheitskriterien wie Analysequalität und dergleichen perfekt erfüllte, für praktische Anwendungen nicht einsetzbar, wenn die Berechnung der Analyse eines Satzes mehrere Tage oder Wochen dauerte.

Probleme des erforderlichen Rechenzeitbedarfs und des erforderlichen Speicherplatzes werden mit unterschiedlichen Methoden untersucht. Im Rahmen der Komplexitätstheorie werden die Komplexitätseigenschaften von Problemen unter mathematischen Gesichtspunkten betrachtet. Unter einem Problem wird dabei nicht ein Einzelproblem verstanden (etwa die Analyse eines spezifischen Satzes mit einer spezifischen Grammatik), sondern eine Problemklasse. Eine solche Problemklasse wäre z. B. die Überprüfung eines beliebigen Satzes mit einer beliebigen Typ-2-Grammatik. Untersucht wird bei Arbeiten aus der Komplexitätstheorie in der Regel nicht die durchschnittliche oder praktische Komplexität eines Problems, sondern die obere Schranke für den schwierigsten Fall (engl. *worst case*). Für Typ-2-Grammatiken ist z. B. bekannt, dass die Komplexität des Problems, für eine Eingabekette zu entscheiden, ob sie zu der von der Grammatik definierten Sprache gehört, auch im worst case nicht schneller als mit kubischem Verlauf zur Basislänge der Eingabekette ansteigt (vgl. Unterkapitel 2.2). Das Problem der Definition sämtlicher Strukturbeschreibungen für einen beliebigen Satz und eine beliebige kontextfreie Syntax ist hingegen bekanntermaßen unlösbar (und somit streng genommen nicht Gegenstand der Komplexitätstheorie), da es Grammatiken gibt, die einem Satz unendlich viele Strukturbäume zuordnen.

Die Arbeiten aus der Komplexitätstheorie geben wichtige Anhaltspunkte dafür, ob eine Lösung für ein Problem im Rahmen der gewählten Formalisierung (z. B. das Problem der Grammatikalitätsentscheidung im Rahmen eines Grammatikformalismus) grundsätzlich implementierbar ist. Dabei wird meist die Dauermenregel akzeptiert, dass die Berechnung einer Lösung dann nicht effizient implementierbar ist, wenn ein variabler Parameter (z. B. die Satzlänge) in der Komplexitätsformel als Exponent erscheint. Natürlich können auch große konstante Exponenten bei großen Basen so große Zahlenwerte ergeben, dass die praktische Berechenbarkeit jenseits des Möglichen liegt.

In der computerlinguistischen Forschung wird gelegentlich über Komplexitätseigenschaften und manchmal sogar auch über Entscheidbarkeitseigenschaften der verwendeten formalen Systeme hinweggegangen. Der Grund dafür liegt nicht etwa darin, dass die mathematischen Beweise der Komplexitätstheorie nicht bekannt wären, dass ihnen nicht geglaubt würde oder dass deren grundsätzliche

Relevanz für den praktischen Einsatz in Zweifel gezogen würde, sondern vielmehr darin, dass ein Parser nicht als (uneingeschränkter) Interpreter des Formalismus angesehen wird, sondern als ein Performanzsystem, das das Kompetenzsystem Grammatik zusätzlich – auch hinsichtlich seiner Komplexitäts- und Entscheidbarkeitseigenschaften – einschränkt. Ein Modellfall für diese Strategie ist das Parsing-System vom Mitchell Marcus, das Ideen des Chomsky-Modells in einen Parser integriert, der **deterministisch** arbeitet (Marcus 1980).

Für die praktische Anwendung computerlinguistischer Technologien sind neben theoretischen Komplexitätsberechnungen für Problemklassen natürlich auch konkrete Messungen tatsächlich implementierter Systeme von Interesse. Solche Messungen entscheiden über die praktische Einsetzbarkeit eines Systems und sie geben dem Entwickler konkrete Anhaltspunkte, um die Performanz seines Systems zu optimieren.

Die vielleicht nahe liegendste Möglichkeit, die Effizienz eines Systems zu bestimmen, besteht darin, dass man mit einer Stoppuhr (oder mit den in Betriebssystemen wie Unix standardmäßig zur Verfügung stehenden Zeitmessfunktionen wie `time`) die Verarbeitungszeit des zu untersuchenden Systems auf der verfügbaren Hardware überprüft. Diese Messmethode ist zwar die einfachste und für praktische Anwendungen sicherlich auch die relevanteste (zumindest dann, wenn Entwicklungsplattform und intendierte Anwendungsplattform identisch sind), sie ist jedoch ungeeignet für den Vergleich von Effizienzresultaten, die unter verschiedenen Betriebssystemen oder auf unterschiedlichen Hardware-Plattformen erhoben wurden. Für derartige Vergleiche bieten sich andere Kriterien an, die von den konkreten Implementierungsdetails abstrahieren.

Backtrack-Parsing

Im Folgenden soll nun ein einfacher von links nach rechts voranschreitender top-down Algorithmus informell vorgestellt werden. Der Algorithmus erzeugt vom Startsymbol ausgehend Linkssatzformen (siehe Unterkapitel 2.2) und vergleicht diese nach und nach mit der Eingabekette. Bei der Erzeugung der Linkssatzformen kann es mehrere anwendbare Regeln, d.h. mehrere Regeln mit identischer linker Seite, geben, von denen der Algorithmus eine zur Expansion auswählen muss. Dabei kann es natürlich vorkommen, dass eine Regel gewählt wird, die nicht zur Ableitung der Eingabe führt. Diese Wahl der falschen Regel kann sich allerdings erst zu einem viel späteren Zeitpunkt – also nachdem schon weitere Expansionen durchgeführt wurden – bemerkbar machen. An einer solchen Stelle betreibt der Algorithmus nun Fehlerbehebung nach dem Prinzip des **Backtracking**: es werden alle Schritte, bis zu der letzten Stelle, an der eine Wahlmöglichkeit bestand, rückgängig gemacht. Anschließend wird eine andere Wahl getroffen und die Verarbeitung kann fortgesetzt werden.

Wir gehen zunächst davon aus, dass der Algorithmus lediglich prüfen soll, ob sich eine Eingabekette relativ zu einer gegebenen kontextfreien Grammatik ableiten lässt oder nicht. Der Algorithmus produziert also keine Strukturbeschreibungen, sondern prüft nur die Grammatikalität einer Eingabekette. Es handelt sich also nicht um einen Parsing-Algorithmus im engeren Sinne, sondern um einen **Erkennungsalgorithmus** (engl. *Recognizer*).

Der Algorithmus besteht aus drei Schritten: einem **EXPAND**-Schritt, der die anwendbaren Regeln wählt und die Linkssatzformen erzeugt, einem **SCAN**-Schritt, der die Linkssatzform mit der Eingabe vergleicht und einem **BACKTRACK**-Schritt, der im Falle eines Fehlers die vorangegangenen Schritte rückgängig macht. Dabei verzichten wir auf eine detaillierte Darstellung des Backtracking-Schrittes.

Algorithmus-Schema BACKTRACK-RECOGNIZER

DATEN: Eine kontextfreie Grammatik $G = \langle \Phi, \Sigma, S, R \rangle$. Die Grammatik darf keine linksrekursiven Regeln (also Regeln der Form $A \rightarrow A\alpha$) enthalten.

INPUT: Eine Eingabekette $w = w_1 w_2 \dots w_n$ ($0 \leq n$)

OUTPUT: Kette akzeptiert/Kette nicht akzeptiert

METHODE:

Beginne mit der Linkssatzform, die nur aus dem Startsymbol besteht und führe **EXPAND** und **SCAN** so oft wie möglich durch.

- **EXPAND:** Ist das erste Symbol in der Linkssatzform ein Nichtterminalsymbol, so wähle eine an dieser Stelle noch nicht angewandte Regel mit diesem Symbol als linke Regelseite und expandiere das Symbol.
- **SCAN:** Ist das erste Symbol der Linkssatzform ein Terminalsymbol, so vergleiche es mit der Eingabekette: Stimmen beide überein, so rücke in der Linkssatzform und in der Eingabekette ein Symbol weiter. Ist in beiden das Ende erreicht, so ist der Rückgabewert *Kette akzeptiert*. Ist nur in einer das Ende erreicht, oder stimmen die beiden Symbole nicht überein, so führe **BACKTRACK** durch.
- **BACKTRACK:** Mache alle Schritte bis zum letzten **EXPAND**-Schritt rückgängig, an dem eine alternative Regel angewendet werden kann. Gibt es keine solche Möglichkeit, so ist der Rückgabewert *Kette nicht akzeptiert*.

Beispiel 3.5.7

Ausgehend von der Grammatik in Beispiel 3.5.1 soll der Algorithmus anhand des Satzes

(3.49) *der Hund sieht die Katze.*

illustriert werden. Der Algorithmus beginnt mit der Linkssatzform

S.

Der erste **EXPAND**-Schritt expandiert dieses Symbol mittels der einzigen anwendbaren Regel $S \rightarrow NP VP$, womit die Linkssatzform also

NP VP

ist. Im nächsten Schritt produziert die Expansion von NP mittels der einzig anwendbaren Regel $\text{NP} \rightarrow \text{DET N}$ die Linkssatzform

Det N VP.

Bei der Expansion von Det gibt es eine Wahlmöglichkeit, denn die beiden Regeln $\text{Det} \rightarrow \text{der}$ und $\text{Det} \rightarrow \text{die}$ sind anwendbar. Nehmen wir an, dass der Algorithmus die Regeln in der Reihenfolge wählt, in der sie auch in Beispiel 3.5.1 angegeben sind, so ergibt sich die Linkssatzform

der N VP

und der anschließende SCAN-Schritt kann über *der* in der Linkssatzform und in der Eingabe weiterrücken. Die Expansion von N mittels der gewählten Regel $\text{N} \rightarrow \text{Hund}$ und der anschließende SCAN-Schritt erkennen erfolgreich das Wort *Hund*, womit die Linkssatzform nur noch

VP

ist. Auch an dieser Stelle sind zwei verschiedene Regeln anwendbar: $\text{VP} \rightarrow \text{V}$ und $\text{VP} \rightarrow \text{V NP}$, wobei sich der Algorithmus nach der angenommenen Strategie zunächst für erstere entscheidet. Damit ist die Linkssatzform also

V.

Im nächsten Expansionsschritt entscheidet sich der Algorithmus zwischen $\text{V} \rightarrow \text{bellt}$ und $\text{V} \rightarrow \text{sieht}$ für erstere Regel und die Linkssatzform ist somit

bellt.

An dieser Stelle stimmen die beiden Symbole in der Eingabe (*sieht*) und der Linkssatzform (*bellt*) nicht überein und Backtracking wird durchgeführt. Es werden nun also alle Schritte bis zur letzten Wahlmöglichkeit rückgängig gemacht: diese war bei der Expansion von V durch *bellt* gegeben. Nach diesem Backtracking wählt EXPAND also die alternative Regel $\text{V} \rightarrow \text{sieht}$ zur Expansion und die Linkssatzform lautet

sieht.

Im SCAN-Schritt stimmen nun zwar die beiden Symbole in der Linkssatzform und der Eingabekette überein. Die Linkssatzform ist aber danach zu Ende, obwohl in der Eingabekette noch die Worte *die Katze* zu finden sind. Damit muss wieder Backtracking durchgeführt werden. Die letzte Stelle, an der eine Wahlmöglichkeit bestand, liegt nun nicht mehr bei der Expansion von V, denn dort wurden inzwischen alle Wahlmöglichkeiten ausgeschöpft. Das Backtracking muss vielmehr noch einen Schritt mehr bis zur Expansion von VP zurückgehen. Dort gibt es als alternative Wahlmöglichkeit noch die Regel $\text{VP} \rightarrow \text{V NP}$, die nun von EXPAND richtigerweise verwendet wird und zur Linkssatzform

Nr.	Linkssatzform	Eingabe	Schritt
1	S	<i>der Hund sieht die Katze</i>	
2	NP VP	<i>der Hund sieht die Katze</i>	EXPAND
3	Det N VP	<i>der Hund sieht die Katze</i>	EXPAND
4	<i>der</i> N VP	<i>der Hund sieht die Katze</i>	EXPAND
5	N VP	<i>Hund sieht die Katze</i>	SCAN
6	<i>Hund</i> VP	<i>Hund sieht die Katze</i>	EXPAND
7	VP	<i>sieht die Katze</i>	SCAN
8	V	<i>sieht die Katze</i>	EXPAND
9	<i>bellt</i>	<i>sieht die Katze</i>	EXPAND
8'	V	<i>sieht die Katze</i>	BACKTRACK nach 8
9'	<i>sieht</i>	<i>sieht die Katze</i>	EXPAND
7''	VP	<i>sieht die Katze</i>	BACKTRACK nach 7
8''	V NP	<i>sieht die Katze</i>	EXPAND
9''	<i>bellt</i> NP	<i>sieht die Katze</i>	EXPAND
8'''	V NP	<i>sieht die Katze</i>	BACKTRACK nach 8''
9'''	<i>sieht</i> NP	<i>sieht die Katze</i>	EXPAND
10'''	NP	<i>die Katze</i>	SCAN
11'''	Det N	<i>die Katze</i>	EXPAND
12'''	<i>der</i> N	<i>die Katze</i>	EXPAND
11''''	Det N	<i>die Katze</i>	BACKTRACK nach 11'''
12''''	<i>die</i> N	<i>die Katze</i>	EXPAND
13''''	N	<i>Katze</i>	SCAN
14''''	<i>Hund</i>	<i>Katze</i>	EXPAND
13'''''	N	<i>Katze</i>	BACKTRACK nach 13'''''
14'''''	<i>Katze</i>	<i>Katze</i>	EXPAND
15'''''			SCAN, akzeptiere.

Tabelle 3.4: Ableitung des Satzes *der Hund sieht die Katze*

V NP

führt. Tabelle 3.4 zeigt die Ableitung im Ganzen. \square

In der Beschreibung des Algorithmus wurde gefordert, dass die Grammatik keine linksrekursiven Regeln enthalten darf. Das Ausklammern von linksrekursiven Regeln ist notwendig, weil diese zur Nichtterminierung des Algorithmus führen können. Angenommen, die Grammatik enthielte eine linksrekursive Regel der Form $A \rightarrow A\alpha$ und der EXPAND-Schritt würde diese dazu benutzen, das Nichtterminal A in einer Linkssatzform $A\beta$ zu expandieren. Die resultierende Linkssatzform wäre damit $A\alpha\beta$. An dieser Stelle könnte der EXPAND-Schritt dieselbe Regel nochmals benutzen, was zur Linkssatzform $A\alpha\alpha\beta$ führen würde. Offensichtlich könnte EXPAND abermals diese Regel benutzen, was sich so fortsetzen würde – der Algorithmus würde nur noch EXPAND-Schritte mit dieser Regel machen und nicht terminieren.

Ein die Effizienz betreffendes Problem von Backtracking-Algorithmen im Allgemeinen liegt in der Notwendigkeit von wiederholten Analysen. Ein Beispiel hierfür ist die Analyse von *sieht* als V: In Schritt 8' wird V mittels der Regel $V \rightarrow \text{sieht}$ expandiert. Dieser Expansionsschritt ist im Prinzip richtig – er wird aber durch das Backtracking in Schritt 9' wieder rückgängig gemacht und muss in Schritt 8''' wiederholt werden. Obwohl *sieht* also schon korrekt als V analysiert wurde, wird dieses Zwischenergebnis durch das Backtracking vergessen. In diesem Beispiel ist davon nur eine einzelne Regel betroffen. Es kann aber auch vorkommen, dass eine ganze Reihe von EXPAND- und SCAN-Schritten durch Backtracking rückgängig gemacht werden und so ganze Konstituenten mehrfach analysiert werden müssen.

Dieses Effizienzproblem wird durch die Speicherung solcher Teilresultate gelöst. Im folgenden Abschnitt stellen wir den Earley-Algorithmus vor, der dem Paradigma der **Dynamischen Programmierung** zugeordnet wird. Darunter werden nicht etwa Programmiermethoden verstanden, sondern Problemlösungsverfahren, die auf dem Prinzip beruhen, dass ein komplexes Gesamtproblem in gleichartige Teilprobleme zerlegt wird. Diese Teilprobleme können unabhängig voneinander gelöst und anschließend zur Gesamtlösung des ursprünglichen Problems kombiniert werden. Das Problem, ob eine Zeichenkette zu der Sprache einer kontextfreien Grammatik gehört, zählt zu den Problemen, die sich auf diese Weise effizienter lösen lassen.

Chart-Parsing

Die Familie der **Chart-Parsing**-Algorithmen ist nach der für sie charakteristischen Datenstruktur, der **Chart**, benannt. Eine Chart ist eine Tabelle, in der Teilresultate einer syntaktischen Strukturanalyse abgelegt werden können. Die Einträge in dieser Tabelle werden **Items** oder **Kanten** genannt.

Unabhängig von den Details, die die verschiedenen Varianten von Chart-Parsing-Algorithmen unterscheiden, enthält jedes Chart-Item zumindest die folgenden Informationen:

- den Satzabschnitt, auf den sich das Item bezieht,
- die Syntaxregel, die angewandt wurde.

Der Satzabschnitt wird üblicherweise durch ein Paar von Zahlen angegeben, wobei die erste Zahl den Anfang des Abschnitts angibt und die zweite das Ende. Die Null bezeichnet dabei den Satzanfang (die Position vor dem ersten Wort der Eingabekette), 1 bezeichnet die Position nach dem ersten Wort usw. Da jedes Chart-Item einen Schritt im Analyseprozess protokolliert, wird auch die Grammatikregel gespeichert, die angewandt wurde.

Beispiel 3.5.8

Im folgenden Satz sind die beim Chart-Parsing verwendeten Zahlenmarkierungen angegeben:

0 *Der* 1 *Hund* 2 *sieht* 3 *die* 4 *Katze* 5

Das folgende Item repräsentiert die Information, dass der Abschnitt der Eingabekette, der mit dem vierten Wort beginnt und dem fünften endet (der also zwischen den Markierungen 3 und 5 liegt), als Nominalphrase analysiert wurde, die aus einem Determinierer und einem Nomen besteht:

$$3 \quad 5 \quad \text{NP} \rightarrow \text{Det N}$$

□

Abhängig davon, ob auch partielle Zwischenresultate angenommen werden oder nicht, spricht man von einem **aktiven** Chart-Parser oder einem **nicht-aktiven** System. Bei aktiven Chart-Parsern haben Items einen **aktiven** und einen **inaktiven** Abschnitt. Der inaktive Abschnitt umfasst den Bereich der angewandten Regel, der bereits analysiert wurde und der aktive Abschnitt den noch nicht vollständig analysierten Rest. Als Trennzeichen zwischen dem aktiven und inaktiven Abschnitt wird traditionell das Begrenzungssymbol • (gesprochen: *dot*) verwendet. Solche Chartkanten werden auch **geteilte Produktionen** oder **dot-ted Items** genannt. Das Chart-Item

$$0 \quad 2 \quad \text{S} \rightarrow \text{NP} \bullet \text{VP}$$

bezeichnet also den Analysezustand, in dem der Parser die ersten zwei Wörter der Eingabekette bereits als dem Typ NP angehörig erkannt hat und für den Abschnitt, der mit dem dritten Wort beginnt, eine Konstituente vom Typ VP erwartet. Wird eine solche VP im weiteren Verlauf des Analyseprozesses gefunden, ist insgesamt eine Konstituente vom Typ S erkannt worden.

Erweiterte Chart-Parsing-Algorithmen verwenden auch komplexere Items, in denen Informationen mehrerer Basis-Items zusammengefasst werden. Dazu zählen vor allem Algorithmen, die (partiell) überlappende inaktive Abschnitte in einem komplexeren Item zusammenfassen (engl. *local ambiguity packing*) oder Algorithmen, die optionale Elemente in einem Item zulassen und somit – streng genommen – mehrere kontextfreie Regeln zu einem Item zusammenfassen.

Die für Chart-Parsing-Algorithmen charakteristische Zwischenspeicherung partieller Analyseresultate hat den Vorteil, dass die wiederholte Analyse derselben Satzabschnitte, die z. B. bei Backtracking-Algorithmen unvermeidbar sein kann, nicht mehr erforderlich ist. Dies führt zumeist zu einer höheren Effizienz von Chart-Parsingalgorithmen, wobei der Speicherbedarf allerdings höher liegt als bei den entsprechenden Backtracking-Algorithmen.

Die Verwendung der Datenstruktur Chart kann mit verschiedenen Suchstrategien (top-down, bottom-up, left-corner usw.) kombiniert werden und erlaubt zumeist die Verwendung beliebiger kontextfreier Grammatiken, während z. B. top-down verarbeitende Backtracking-Algorithmen bei Grammatiken mit linksrekursiven Regeln wie oben erwähnt nicht notwendigerweise terminieren. Die bekanntesten Varianten von Chart-Parsing-Algorithmen sind der **Earley-Algorithmus** (Earley 1970) und der **Cocke-Kasami-Younger-Algorithmus** (Kasami 1965, Younger 1967). Die zugrunde liegenden Ideen vieler moderner Implementierungen von Chart-Parsing-Systemen gehen auf Martin Kay (Kay 1980) zurück.

PROZEDUR EXPAND

DATEN: Eine kontextfreie Grammatik $G = \langle \Phi, \Sigma, R, S \rangle$

METHODE:

Wenn die Chart bereits eine Kante der Form

$$i \quad j \quad A \rightarrow \alpha \bullet B \beta \quad (\text{mit: } i \leq j)$$

enthält, dann wird für jede Grammatikregel der Form

$$B \rightarrow \gamma \in R$$

ein neues Chart-Item der Form

$$j \quad j \quad B \rightarrow \bullet \gamma$$

angelegt.

Abbildung 3.42: Die Prozedur EXPAND

Ein einfacher Earley-Recognizer: In diesem Abschnitt werden wir den wohl bekanntesten Chart-Parsing-Algorithmus – den Earley-Algorithmus (Earley 1970), benannt nach Jay Earley in einer vereinfachten Form vorstellen.

Im Kern besteht der Earley-Algorithmus aus drei Prozeduren, die zyklisch angewandt werden:

1. EXPAND (oder: PREDICT) ist die Top-down-Komponente des Algorithmus und legt ausschließlich aktive Items an (Abb. 3.42);
2. SCAN (oder: SHIFT) erzeugt (ausschließlich inaktive) Items für die Wörter der Eingabekette (Abb. 3.43);
3. COMPLETE (oder: REDUCE), schließlich, ist diejenige Prozedur, die bereits vorhandene Chart-Einträge zu größeren Einheiten zusammenfasst und die deshalb als Bottom-up-Komponente angesehen werden kann (Abb. 3.44).

Im Folgenden werden die einzelnen Prozeduren definiert. Dabei enthält jede dieser Definitionen einen Daten- und Methoden-Teil. Im Daten-Teil ist aufgelistet, auf welchen Daten (außer der Chart) die Prozedur operiert, im Methoden-Teil, welche Aktionen damit ausgeführt werden.

EXPAND (Abb. 3.42) produziert neue Items, wenn sich bereits ein Item in der Chart befindet, dessen erstes Symbol im aktiven Abschnitt auf der linken Regelseite mindestens einer Grammatikregel vorkommt. Alle Kanten, die von EXPAND produziert werden, sind zyklisch, d.h. sie führen von einer Position j wiederum zu j , und haben einen leeren inaktiven Abschnitt.

EXPAND produziert also Top-down-Hypothesen (daher auch der Name PREDICT; dt. *vorhersagen*) über die weitere Feinstruktur bereits bestehender Annahmen über die Eingabekette.

PROZEDUR SCAN

DATEN: Eine Eingabekette $w = w_1 \ w_2 \ \dots \ w_n$ (mit: $1 \leq n$)

METHODE:

Wenn die Chart ein Item der Form:

$$i \quad j-1 \quad A \rightarrow \alpha \bullet w_j \beta$$

enthält, dann wird ein neues Item der Form

$$i \quad j \quad A \rightarrow \alpha \ w_j \bullet \beta$$

angelegt.

Abbildung 3.43: Die Prozedur SCAN

Beispiel 3.5.9

Wenn wir annehmen, dass die Chart das Item

$$0 \quad 0 \quad S \rightarrow \bullet \ NP \ VP$$

enthält und in der Grammatik die Regel

$$NP \rightarrow DET \ N$$

vorkommt, dann würde EXPAND das Item

$$0 \quad 0 \quad NP \rightarrow \bullet \ DET \ N$$

produzieren.

□

Die Prozedur **SCAN** (Abb. 3.43) kann als Lexikonzugriffsprozedur angesehen werden. Wenn in der Chart bereits ein Item existiert, dass das Vorkommen eines Wortes w_j an der aktuellen Input-Position j voraussagt, und w_j tatsächlich an Position j vorkommt, wird eine neue Chart-Kante generiert, die dieses Vorkommen von w_j als erkannt, d.h. als Bestandteil des inaktiven Abschnitts, registriert.

PROZEDUR COMPLETE

DATEN: —

METHODE:

Wenn die Chart bereits eine Kante der Form

$$i \ j \quad A \rightarrow \alpha \bullet B \beta$$

und eine weitere Kante der Form:

$$j \ k \quad B \rightarrow \gamma \bullet$$

enthält, dann wird ein neues Item

$$i \ k \quad A \rightarrow \alpha B \bullet \beta$$

in die Chart eingetragen.

Abbildung 3.44: Die Prozedur COMPLETE

Beispiel 3.5.10

Angenommen, die Chart enthält bereits das Item

$$0 \ 0 \quad \text{DET} \rightarrow \bullet \text{ der}$$

und nach der Position 0, d.h. als erstes Wort der Eingabekette, befindet sich das Wort *der*, dann wird der Chart durch **SCAN** ein neues Item der Form

$$0 \ 1 \quad \text{DET} \rightarrow \text{der} \bullet$$

hinzugefügt.

□

Im Gegensatz zu **EXPAND** verwendet die Prozedur **SCAN** nicht die Grammatik, sondern lediglich die bereits vorhandenen Chart-Kanten und die Eingabekette. Während **EXPAND** ausschließlich aktive Items produziert, erzeugt **SCAN** typischerweise inaktive Items (in vielen Grammatiken werden nämlich terminale Symbole nur durch unär verzweigende Regeln, z.B. $\text{DET} \rightarrow \text{der}$, $\text{N} \rightarrow \text{Hund}$ usw., eingeführt). Die dritte Prozedur, **COMPLETE**, nimmt weder auf die Eingabekette noch auf die Grammatik Bezug: Sie operiert ausschließlich auf den bereits vorhandenen Chart-Kanten (Abb. 3.44).

Die Aufgabe von **COMPLETE** besteht darin, inaktive Kanten (d.h. vollständig erkannte Teilstrukturen) mit aktiven Kanten zu verbinden.

Beispiel 3.5.11

Angenommen, durch **EXPAND** wurde vorausgesagt, dass sich am Anfang der Eingabekette eine Nominalphrase befindet, und durch **SCAN** wurde ein Determinierer eingelesen, dann verbindet **COMPLETE** diese beiden Informationen zu einem

Item, das eine NP repräsentiert, deren erstes Element, der Determinierer, bereits erkannt ist und deren zweites Element, ein Nomen, noch den Status einer Top-down-Hypothese hat.

Angenommen also, die Chart enthält das inaktive Item

$$0 \quad 1 \quad \text{DET} \rightarrow \text{der} \bullet$$

sowie das aktive Item

$$0 \quad 0 \quad \text{NP} \rightarrow \bullet \text{DET N},$$

dann erzeugt **COMPLETE** das neue Item

$$0 \quad 1 \quad \text{NP} \rightarrow \text{DET } \bullet \text{ N}.$$

□

Mit Hilfe dieser drei Grundprozeduren lässt sich eine einfache Variante des Earley-Algorithmus schematisch spezifizieren. Einige Regeltypen (Tilgungs- und Kettenregeln) werden ausgeschlossen, in einer vollständig spezifizierten Version des Earley-Algorithmus sind sie jedoch durchaus zulässig. Zudem bleiben einige Aspekte, insbesondere die Interaktion zwischen den einzelnen Prozeduren, zunächst unterspezifiziert:

Algorithmus-Schema EARLEY-RECOGNIZER

DATEN: Eine kontextfreie Grammatik $G = \langle \Phi, \Sigma, S, R \rangle$ und eine Chart C. Die Grammatik darf weder Tilgungsregeln (Regeln des Typs: $X \rightarrow \epsilon$) noch Kettenregeln enthalten, die Chart ist im Anfangszustand leer.

INPUT: Eine Eingabekette $w = w_1 w_2 \dots w_n$ ($0 \leq n$)

OUTPUT: Kette akzeptiert/Kette nicht akzeptiert

METHODE:

1. Initialisierung:

- a) Erzeuge für alle das Startsymbol der Grammatik expandierenden Regeln der Form $S \rightarrow \alpha$ eine Chart-Kante der Form

$$0 \quad 0 \quad S \rightarrow \bullet \alpha.$$

- b) Wende die Prozedur **EXPAND** auf diese Kanten solange an, bis keine neuen Items mehr erzeugt werden.

2. Erzeugung weiterer Chart-Kanten: Für alle Positionen $j = 0, \dots, n$ und alle Positionen $i = 0, \dots, j$: Berechne die Items mit der Startposition i und der Endposition j wie folgt:

- a) Wende die Prozedur **SCAN** auf alle Items mit der Startposition i und der Endposition $j - 1$ an.
- b) Wende **EXPAND** und **COMPLETE** solange an, bis diese beiden Prozeduren keine neuen Kanten mehr erzeugen.

3. Auswertung: Wenn die Chart ein Item der Form

$$0 \quad n \quad S \rightarrow \alpha \bullet$$

enthält, dann ist der Rückgabewert *Kette akzeptiert*, sonst *Kette nicht akzeptiert*.

Am Beispiel der Grammatik G aus Beispiel 3.5.1 und der Eingabekette $w = \text{der Hund bellt}$ soll das Vorgehen des Algorithmus verdeutlicht werden. Da es nur eine das Startsymbol expandierende Regel gibt ($S \rightarrow \text{NP VP}$), wird im ersten Schritt der Initialisierung lediglich das Item

$$0 \quad 0 \quad S \rightarrow \bullet \text{NP VP}$$

angelegt. Im zweiten Schritt wird nun die Prozedur EXPAND angewendet, die folgende Items erzeugt:

$$\begin{array}{ll} 0 \quad 0 & \text{NP} \rightarrow \bullet \text{DET N} \\ 0 \quad 0 & \text{DET} \rightarrow \bullet \text{der} \\ 0 \quad 0 & \text{DET} \rightarrow \bullet \text{die} \end{array}$$

Wenn nun SCAN auf das Item

$$0 \quad 0 \quad \text{DET} \rightarrow \bullet \text{der}$$

angewendet wird, erhält man das erste inaktive Item

$$0 \quad 1 \quad \text{DET} \rightarrow \text{der} \bullet$$

Darauf kann COMPLETE mit dem Resultat

$$0 \quad 1 \quad \text{NP} \rightarrow \text{DET} \bullet \text{N}$$

angewandt werden. Für dieses Item produziert EXPAND

$$\begin{array}{ll} 1 \quad 1 & \text{N} \rightarrow \bullet \text{Hund} \\ 1 \quad 1 & \text{N} \rightarrow \bullet \text{Katze} \end{array}$$

Nach abgeschlossener Analyse enthält die Chart die in Tabelle 3.5 angegebenen Items.

Mit Kante 19. ist eine inaktive, mit dem Startsymbol etikettierte Kante erzeugt worden, die den kompletten Satz überspannt – die Eingabekette wird also akzeptiert. Weiter sind alle möglichen Analysen in der Chart gespeichert. Im Fall von ambigen Sätzen zeigt sich ein weiterer Vorteil des Chart-Parsing: Verschiedene Ableitungen werden gleichzeitig verfolgt und sind in der Chart gespeichert, ohne dass eine erneute Analyse erforderlich wäre.

Nr.	Item		Bemerkung
1.	0 0	$S \rightarrow \bullet NP VP$	(Initialisierung (a))
2.	0 0	$NP \rightarrow \bullet DET N$	(Initialisierung (b))
3.	0 0	$DET \rightarrow \bullet der$	(Initialisierung (b))
4.	0 0	$DET \rightarrow \bullet die$	(Initialisierung (b))
5.	0 1	$DET \rightarrow der \bullet$	(SCAN 3.)
6.	0 1	$NP \rightarrow DET \bullet N$	(COMPLETE 2. mit 5.)
7.	1 1	$N \rightarrow \bullet Hund$	(EXPAND 6.)
8.	1 1	$N \rightarrow \bullet Katze$	(EXPAND 6.)
9.	1 2	$N \rightarrow Hund \bullet$	(SCAN 7.)
10.	0 2	$NP \rightarrow DET N \bullet$	(COMPLETE 6. mit 9.)
11.	0 2	$S \rightarrow NP \bullet VP$	(COMPLETE 1. mit 10.)
12.	2 2	$VP \rightarrow \bullet V$	(EXPAND 11.)
13.	2 2	$VP \rightarrow \bullet V NP$	(EXPAND 11.)
14.	2 2	$V \rightarrow \bullet bellt$	(EXPAND 12. bzw. 13.)
15.	2 2	$V \rightarrow \bullet sieht$	(EXPAND 12. bzw. 13.)
16.	2 3	$V \rightarrow bellt \bullet$	(SCAN 14.)
17.	2 3	$VP \rightarrow V \bullet$	(COMPLETE 13. mit 16.)
18.	2 3	$VP \rightarrow V \bullet NP$	(COMPLETE 12. mit 16.)
19.	0 3	$S \rightarrow NP VP \bullet$	(COMPLETE 11. mit 17.)

Tabelle 3.5: Chart für *der Hund bellt*

Earley-Algorithmus mit Vorwärtsverkettung: In der bisherigen Darstellung des Algorithmus blieb die Interaktion der einzelnen Prozeduren weitgehend unterspezifiziert. Die folgende Version EARLEY_{VK} ist in dieser Hinsicht genauer. Zunächst wird eine zusätzliche Prozedur CLOSURE eingeführt:

PROZEDUR CLOSURE

INPUT:

Eine Kante $k = i \ j \ A \rightarrow \alpha \bullet \beta$

DATEN:

Eine kontextfreie Grammatik und eine Chart C.

METHODE:

Wenn $k \notin C$,
dann füge k in C ein und

wenn $\beta = \epsilon$ (k ist eine inaktive Kante),
dann COMPLETE_{VK}(k),
sonst EXPAND_{VK}(k).

SEITENEFFEKT:

Berechnung von C.

CLOSURE trägt Items in die Chart ein und ruft bei inaktiven Kanten COMPLETE_{VK}, bei aktiven Kanten hingegen EXPAND_{VK} auf und erreicht damit eine Hüllensbildung. Diese beiden vorwärtsverkettenden Varianten von EXPAND und COMPLETE sind wie folgt definiert:

PROZEDUR EXPAND_{VK}

INPUT:

Eine Kante $K = i \ j \ A \rightarrow \alpha \bullet \ B \ \beta$

DATEN:

Eine kontextfreie Grammatik $G = \langle \Phi, \Sigma, S, R \rangle$
und eine Chart C.

METHODE:

Führe für alle Regeln der Form $B \rightarrow \gamma \in R$
die Prozedur CLOSURE($j \ j \ B \rightarrow \bullet \ \gamma$) aus.

PROZEDUR COMPLETE_{VK}

INPUT:

Eine Kante $K = j \ k \ A \rightarrow \alpha \bullet$

DATEN:

Eine Chart C

METHODE:

Für jedes item der Form $i \ j \ B \rightarrow \beta \bullet A\gamma \in C$
führe CLOSURE($i \ k \ B \rightarrow \beta A \bullet \gamma$) aus.

Nun muss auch noch die dritte Unterprozedur SCAN angepasst werden:

PROZEDUR SCAN_{VK}

INPUT:

Ein Wort w_i aus der Eingabekette

DATEN:

eine Grammatik G

METHODE:

Für alle Regeln der Form $B \rightarrow w_i \in R$
führe CLOSURE($i - 1 \ i \ B \rightarrow w_i \bullet$) aus.

Schließlich wird nun noch die Hauptprozedur RECOGNIZE neu definiert:

PROZEDUR EARLEY-RECOGNIZER mit Vorwärtsverkettung

INPUT:

Eine Eingabekette $w = w_1 w_2 \dots w_n$

OUTPUT:

Kette akzeptiert/Kette nicht akzeptiert

DATEN:

Eine Grammatik $G = \langle \Phi, \Sigma, S, R \rangle$ und eine Chart C.

Die Grammatik darf weder Tilgungsregeln noch Kettenregeln enthalten, C ist im Anfangszustand leer.

METHODE:

1. Initialisierung: Berechne $\text{CLOSURE}(0 \ 0 \ S \rightarrow \bullet\alpha)$ für alle Regeln der Form $S \rightarrow \alpha \in R$

2. Erzeugung der anderen Items: Berechne $\text{SCAN}_{VK}(w_i)$ für alle i ($1 \leq i \leq n$)

3. Auswertung: Wenn es mindestens eine Kante der Form $0 \ n \ S \rightarrow \alpha \bullet \in C$ gibt,
dann ist der Rückgabewert „Kette akzeptiert“,
sonst „Kette nicht akzeptiert“.

Auf der Basis dieser Spezifikation lässt sich der Earley-Algorithmus nun exemplarisch implementieren. Die Grammatik darf jedoch nach wie vor weder Tilgungs- noch Kettenregeln enthalten und für eine praktisch einsetzbare Implementierung wären außerdem noch die systematische Einbindung eines Lexikons und der Aufbau von Strukturbäumen zu realisieren.

Statistisches Parsing

Statistische Parsing-Modelle sind in den vergangenen Jahren ein besonders aktives Gebiet gewesen. In der Literatur (z. B. Manning und Schütze 2003) werden nicht selten drei unterschiedliche Ziele unterschieden, die die Verwendung von Wahrscheinlichkeiten beim Parsing motivieren:

- Bestimmung des genauen Wortlauts eines Satzes: Diese Aufgabe entsteht, wenn ein Eingabesatz nicht bereits als korrekte, vollständige und diskrete Folge von lexikalischen Einheiten vorliegt, sondern verrauscht, unterspezifiziert oder fehlerhaft sein kann. Dass ein Eingabesatz nicht als eindeutige Folge von Wörtern vorliegt, sondern z. B. als Verband von sich zum Teil wechselseitig ausschließenden Worthypothesen, ist insbesondere bei der Analyse gesprochener Sprache oder bei der Verarbeitung handschriftlicher Eingaben der Fall. Die Aufgabe eines statistischen Parsers besteht dann darin, aus den verschiedenen Pfaden eines Worthypothesengraphen denjenigen zu ermitteln, der die höchste Wahrscheinlichkeit auf der Basis einer

gegebenen probabilistischen Grammatik hat und außerdem eine vollständige Sequenz von Kanten ist, die einen Startknoten mit einem Endknoten verbindet.

- Strukturelle Disambiguierung (*parse selection*): Wie wir bereits in Abschnitt 3.5.2 beschrieben haben, ist bei komplexeren Sätzen und der Verwendung größerer Grammatiken mit einer Vielzahl unterschiedlicher syntaktischer Lesarten für ein und dieselbe Wortkette zu rechnen und die Ambiguität wird natürlich noch größer, wenn zusätzlich anstelle einer Eingabekette ein Worthypothesengraph geparsst werden muss. Die statistischen Informationen können dazu dienen, aus der Vielzahl von syntaktischen Strukturen die n wahrscheinlichsten herauszufiltern.
- Performanzoptimierung des Analyseprozesses: Statistische Informationen können auch bereits während des Parsing-Prozesses verwendet werden, um wahrscheinlichere Pfade im Suchraum zuerst (*best first*) zu verfolgen oder schlecht bewertete, d.h. sehr unwahrscheinliche Pfade gar nicht weiter zu verfolgen (*Pruning*).

Die Aufgabe des statistischen Parsings im Sinne der strukturellen Disambiguierung besteht darin, die Wahrscheinlichkeit einer Strukturbescheinigung δ für einen gegebenen Satz s zu ermitteln, d.h. den Wert $P(\delta|s)$ zu berechnen. Im Falle einer (probabilistischen) kontextfreien Grammatik ist diese Strukturbeschreibung ein Baum, die Grundzüge des statistischen Parsings sind jedoch zunächst unabhängig vom gewählten Grammatikformalismus. Die Wahrscheinlichkeit der Strukturbeschreibung hängt bei einer probabilistischen kontextfreien Grammatik von den Wahrscheinlichkeiten der Regeln der jeweils verwendeten Grammatik g (und gegebenenfalls von zusätzlichen, z. B. lexikalischen Informationen) ab, so dass $P(\delta|s, g)$ zu ermitteln ist. In den meisten Fällen soll diejenige Strukturbeschreibung δ_i ermittelt werden, die diese Wahrscheinlichkeit maximiert:

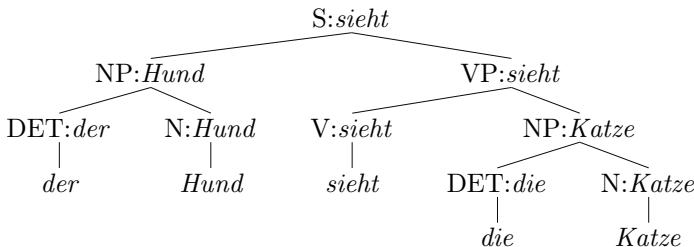
$$\delta_i = \arg \max_{\delta} P(\delta|s, g)$$

In Abschnitt 3.5.1 hatten wir bereits die beiden zentralen Unabhängigkeitsannahmen von probabilistischen kontextfreien Grammatiken genannt: die Unabhängigkeit der Wahrscheinlichkeit einer Regel (oder eines Teilbaums) von ihrem strukturellen Kontext und die Unabhängigkeit von dem lexikalischen Material. Die partielle Rücknahme der ersten Unabhängigkeitsannahme führt zu Modellen wie der history-based grammar (Black et al. 1993), in der der Ableitungskontext als zusätzliches Kriterium verwendet wird. Auch das in Johnson (1998) beschriebene Parsing-Modell, bei dem jede Kategorie zusätzlich mit der Kategorie des Mutterknotens annotiert wird, ist dieser Richtung zuzuordnen. Durch diese einfache Maßnahme gelingt es u.a., die unterschiedlichen Expansionswahrscheinlichkeiten für NPs in Subjekt- bzw. Objektposition zu modellieren. Beispiele für diese unterschiedlichen Expansionswahrscheinlichkeiten sind:

1. Wenn man annimmt, dass Subjekt-NPs (S-dominiert) häufiger nur aus einem Personalpronomen bestehen als Objekt-NPs (VP-dominiert), so gilt: $P(\text{NP} \rightarrow \text{PERPRON}|S) > P(\text{NP} \rightarrow \text{PERPRON}|VP)$.

2. Reflexivpronomina sind im Deutschen kein zulässiges Subjekt, d.h., $P(\text{NP} \rightarrow \text{REFLPRON} \mid S) = 0$.
3. Expletivpronomina als Objekt sind sehr selten.

Die (partielle) Rücknahme der zweiten Unabhängigkeitsannahme führt hingegen zu *lexikalisierten* probabilistischen Grammatiken (Collins 1999). Bei einer lexikalisierten probabilistischen kontextfreien Grammatik werden die nichtterminalen Kategorien eines Baums zusätzlich mit lexikalisiertem Material und gegebenenfalls auch einer Wortartinformation annotiert:



Der Baum wird bottom-up mit dem lexikalischen Material annotiert, mit dem auch die jeweilige Kopf-Konstituente etikettiert ist. Die Objekt-NP erhält das zusätzliche Etikett *Katze*, die VP wird zusätzlich mit *sieht* annotiert und ebenso die ursprünglich lediglich mit *S* annotierte Wurzel des Baums.

Hindle und Rooth (1993) konnten zeigen, dass sich die Disambiguierung von PP-Anbindungsambiguitäten durch die Lexikalisierung einer Grammatik erheblich verbessern lässt.

Diese Ausdifferenzierung hat allerdings ihren Preis: Für jede der „lexikalisierten Kategorien“ muss gegebenenfalls Trainingsmaterial bereit gestellt werden, um die Parameter des Modells schätzen zu können. Bei einer Sprache (wie dem Deutschen), in der es möglich ist, etwa durch Ad-hoc-Komposition, beliebig viele lexikalische Elemente zu bilden, führt dies notwendigerweise zum **Problem der spärlichen Daten** (engl. *sparse data problem*), d.h. zu einer Situation, in der die für eine korrekte Schätzung der Parameter erforderlichen Daten nicht oder zumindest nicht in dem benötigten Umfang zur Verfügung stehen. Ein weiteres Problem besteht darin, dass sich bei manchen Konstruktionen (z. B. im Falle der Koordination von Konstituenten) kein eindeutiger lexikalischer Kopf identifizieren lässt. In Klein und Manning (2003) wird gezeigt, dass sich auch mit nicht-lexikalisierten probabilistischen Grammatiken Resultate erzielen lassen, die den lexikalisierten Systemen wie Collins (1999) sehr nahe kommen.

3.5.3 Literaturhinweise

Einen guten Überblick über GPSG und LFG (im Kontrast zur Prinzipien- und Parameter-Theorie, Chomsky 1981) bietet Sells (1985). Die Darstellung der HPSG in diesem Unterkapitel orientiert sich an dem Standardwerk von Pollard und Sag (1994). Als einführende Bücher in aktuellere Versionen der HPSG bieten sich Sag, Wasow und Bender (2003) für das Englische und Müller (2008) für

das Deutsche an. Eine allgemeine Einführung in die Syntax, die nicht strikt an eine bestimmte linguistische Theorie gebunden ist, liefert Van Valin und LaPolla (1997).

Eine deutschsprachige Einführung in Parsing-Methoden ist Naumann und Langer (1994), in dem auch der hier besprochene Earley-Algorithmus im Detail behandelt wird. Umfassend hinsichtlich der behandelten Algorithmen, aber zugleich auch stellenweise sehr knapp in der Darstellung ist das klassische Werk von Aho und Ullman (1972), das zwar nicht mehr den aktuellen Stand der Forschung repräsentiert, aber dennoch eine hervorragende Basislektüre darstellt. Probabilistische Grammatiken und statistische Modelle für das Parsing werden in Manning und Schütze (2003) dargestellt.

3.6 Semantik

Christian Ebert, Michael Schiehlen, Ralf Klabunde und Stefan Evert

Die Semantik ist die Teildisziplin der Linguistik, die sich mit der Bedeutung natürlichsprachlicher Ausdrücke beschäftigt, seien dies nun Worte (lexikalische Semantik), Sätze (Satzsemantik) oder Texte (Diskurssemantik). Ziel einer semantischen Theorie ist die Analyse derjenigen Prozesse, die den Rezipienten einer Äußerung in die Lage versetzen, die Äußerung mit bestimmten Sachverhalten in der Welt in Beziehung zu setzen. Davon abzugrenzen sind zum einen die Prozesse, die der Produzent einer Äußerung durchläuft, wenn er seine Gedanken in Worte fasst. Ebenfalls nicht Untersuchungsgegenstand der Semantik sind die Prozesse, anhand derer der Rezipient einer Äußerung die Absicht erkennt, die zu der Äußerung bzw. zur Wahl bestimmter Ausdrücke in der Äußerung geführt hat. Bei letzteren Prozessen redet man auch gemeinhin von „Verstehen“. So kann selbst ein einfacher Satz wie

(3.50) *Willi schläft.*

auf sehr verschiedene Arten verstanden werden. Er kann eine schlichte Mitteilung eines Sachverhalts sein. Er kann als Aufforderung dienen, leise zu sein, etwa wenn Willi ein Säugling ist, der gerade eingeschlafen ist. Er kann aber auch vor Gefahr warnen, z. B. wenn Willi der Busfahrer ist, der eine Reisegruppe in den Urlaub fährt und (3.50) auf der Autobahn von einem schockierten Fahrgast in der ersten Reihe an den Rest der Reisegruppe gerichtet wird. Mit der Äußerung eines Satzes können also verschiedene Handlungen ausgeführt werden. Eine Analyse der Handlungsbezogenheit sprachlicher Äusserungen fällt in den Bereich der **Sprechakttheorie**, aber nicht in den Bereich der Semantik. Diese beschäftigt sich tatsächlich nur mit der **literalen Bedeutung**, die für den Fall von (3.50) etwa paraphrasiert werden kann mit: das Individuum, das mit Willi bezeichnet wird, hat die Eigenschaft zu schlafen.

Diese literale Bedeutung ist uns Mitgliedern der Sprachgemeinschaft der Deutschsprecher deshalb zugänglich, weil wir uns per Konvention darauf geeinigt haben, dass bestimmte Worte bestimmte Sachverhalte bezeichnen, so wie wir im Verkehrsunterricht die Konvention gelernt haben, die dem Rotsignal einer Ampel an einer Kreuzung eine Bedeutung gibt. Diese Konventionsidee geht auf (Lewis 1969) des Philosophen David Lewis (1941–2001) zurück und man spricht entsprechend auch statt von literaler von **konventioneller Bedeutung**. Sie ist zu unterscheiden vom **kommunikativen Sinn** einer Äußerung, also dem, was ein Sprecher mit der Äußerung eines Ausdrucks bezwecken will. Dieser Unterschied ist in etwa der zwischen *sagen* und *meinen*. So kann ein Sprecher des Deutschen bei strömendem Regen beispielsweise *Heute haben wir wieder mal Spitzенwetter!* sagen, aber das Gegenteil meinen. Der Philosoph Paul Grice (1913–1988) beschäftigte sich in seiner Arbeit (Grice 1957) mit diesem Unterschied. Die Semantik befasst sich also mit der literalen, konventionellen Bedeutung bzw. dem *Gesagten*. Die Untersuchung des kommunikativen Sinns und des in einer Äußerung damit Gemeinten ist Teil der **Pragmatik**, die in Unterkapitel 3.7 behandelt wird.

Die Semantik ist eine sehr alte Disziplin. Der wichtigste und einflussreichste Beitrag zur Semantik kam – und kommt immer noch – aus der Logik, insbesondere hat die moderne formale Logik zur Entwicklung einer **formalen Semantik** beigetragen. Diese Semantikrichtung stützt sich zur mathematisch exakten Erklärung der Folgerungsbeziehungen zwischen natürlichsprachlichen Sätzen auf die formale Logik. Viele der in diesem Unterkapitel vorgestellten Methoden bauen daher auf dem im Logik-Unterkapitel 2.1 vorgestellten Inventar an Repräsentationen und Schlussfolgerungen auf.

Dabei geht man zunächst vom grundlegenden Bedeutungsbegriff der **Wahrheitsbedingungen** eines einfachen Satzes aus. Die Bedeutung eines einfachen Satzes wie (3.50) zu verstehen, heißt damit, zu wissen, welche Bedingungen in der Welt vorliegen müssen, damit er wahr ist. Diese Idee geht auf den Logiker und Sprachphilosophen Gottlob Frege (1848–1925) zurück, sie bildet auch die Grundlage für die **Montague-Semantik**, den prominentesten Vertreter der logischen Semantik, der auf Arbeiten des Logikers und Sprachtheoretikers Richard Montague (1930–1971) zurückgeht. In Abschnitt 3.6.1 erläutern wir die auf Frege zurückgehenden Grundannahmen bzgl. der Bedeutung natürlichsprachlicher Ausdrücke und wenden uns dann in Abschnitt 3.6.2 den Grundlagen der Montague-Semantik zu.

Während man mit Montagues Arbeiten nun ein formal exaktes System zur Hand hatte, mit dem sich Satzbedeutungen auf adäquate Weise analysieren ließen, merkte man bald, dass es für die Beschreibung ganzer Diskurse anderer Herangehensweisen bedurfte. Ein Phänomen, das hier insbesondere ins Blickfeld rückte, betraf die Analyse von satzübergreifenden Anaphern, für die innerhalb einer satzorientierten Wahrheitsbedingungssemantik kein Platz zu sein schien. Die von Hans Kamp und Irene Heim in den 80er Jahren vorgeschlagene Lösung beruht auf einer Erweiterung des Bedeutungsbegriffs von Wahrheitsbedingungen auf **Kontextänderungspotentiale**. Einen Satz zu verstehen heißt in dieser Sichtweise, zu wissen, wie er einen gegebenen Kontext verändert. In Abschnitt 3.6.3 werden wir als den bedeutendsten Vertreter dieser Richtung der Diskurssemantik, die aufgrund der Betonung der Veränderung auch **dynamische Semantik** genannt wird, die von Kamp entwickelte **Diskursrepräsentationstheorie** (kurz DRT) kennenlernen.

Die **Computersemantik** versucht primär, formalsemantische Untersuchungen zur Bedeutung natürlichsprachlicher Ausdrücke wie Wörtern, Sätzen oder Texten in Algorithmen und Programme umzusetzen, um so eine maschinelle Bedeutungsbestimmung zu erlangen. Eines der Kernprobleme der Computersemantik ist die Verarbeitung von **Mehrdeutigkeiten** (auch **Ambiguitäten** genannt). Sprachliche Ambiguitäten treten auf verschiedenen Ebenen auf, nämlich als lexikalische Ambiguitäten (3.51), syntaktische Ambiguitäten (3.52) oder semantische Ambiguitäten (3.53).

(3.51) *Auf meinem Schreibtisch ist eine Maus!*

(3.52) *Ich habe Maria noch vor der Konferenz in Berlin getroffen.*

(3.53) *Ein Buch hat jeder gelesen.*

Die lexikalische Ambiguität (3.51) ist durch die Lesarten von *Maus* als Tier oder Computerzubehör gegeben. Die syntaktische Ambiguität in (3.52) ergibt sich durch die unterschiedliche Anbindung der Präpositionalphrase an die anderen Konstituenten (vgl. hierzu Unterkapitel 3.5): Entweder fand das Treffen mit Maria in Berlin statt oder die Konferenz. Die semantische Ambiguität in (3.53) ist durch die beiden Möglichkeiten der Quantoren *ein* und *jeder* gegeben, Skopus über den jeweils anderen zu nehmen: Entweder es gibt ein bestimmtes Buch, das jeder gelesen hat, oder für jeden gilt, dass er irgendein Buch gelesen hat.

Den Prozess der Bestimmung der intendierten Bedeutung eines ambigen Ausdrucks bezeichnet man als **Disambiguierung**. Man muss aber eine Ambiguität nicht immer auflösen. Oft kann man mit einer semantischen Repräsentation auch schon inferieren, ohne alle Details der Bedeutung zu kennen. Für solche Fälle kann man Techniken der **Unterspezifikation** benutzen, die im Abschnitt 3.6.4 vorgestellt werden.

3.6.1 Grundlagen der natürlichsprachlichen Semantik

Zunächst soll es darum gehen, den Untersuchungsgegenstand der natürlichsprachlichen Semantik, den Begriff der **Bedeutung** eines Ausdrucks, näher zu betrachten (statt von der *Bedeutung* eines Ausdrucks spricht man auch von seinem **Denotat** und man sagt dann statt *bedeuten* auch **denotieren**). Die im folgenden vorgestellten grundlegenden Überlegungen hierzu gehen großteils auf den Logiker und Sprachphilosophen Gottlob Frege (1848–1925) zurück.

Bedeutungen und Wahrheitsbedingungen

Was ist die Beziehung zwischen Sätzen und realer Welt? Über diese Beziehung macht die Semantik zwei grundlegende Annahmen: Die erste Annahme ist, dass Sätze im Wesentlichen dazu dienen, die Welt zu beschreiben. Das trifft augenscheinlich zunächst nur für einfache Aussagesätze wie *Die Erde ist eine Kugel* zu. Eine Fragesatz wie *Wie spät ist es?* passt nicht zu dieser Annahme. Die semantische Theorie beschäftigt sich also zuvorderst mit Aussagesätzen, wie es ja auch die Aussagenlogik tut. In diesem Kapitel werden wir uns auch auf Aussagesätze beschränken und die Semantik von z. B. Fragesätzen aussen vor lassen.

Die zweite Annahme ist, dass man die Bedeutung eines Satzes kennt, wenn man über die Bedingungen Bescheid weiß, unter denen der Satz wahr ist – d.h. seine **Wahrheitsbedingungen** kennt. Diese Überlegung geht auf Frege zurück, aber auch der Philosoph Ludwig Wittgenstein (1889–1951) vertrat diese Auffassung, wie der folgende, in diesem Zusammenhang oft zitierte Eintrag aus seinem *Tractatus logico-philosophicus* zeigt:

Einen Satz verstehen, heißt, wissen, was der Fall ist, wenn er wahr ist. (Man kann ihn also verstehen, ohne zu wissen, ob er wahr ist.)

(Wittgenstein 1922, Satz Nr. 4.024)

Man versteht in dieser Sichtweise also die Bedeutung des Satzes *Der Präsident der USA hatte am 3. April 2009 Schnupfen*, wenn man die Umstände kennt, unter denen er wahr ist – ohne zu wissen, ob er tatsächlich wahr ist. Etwas anders formuliert: hätte man genügend Information über die tatsächlichen Umstände, könnte man dem Satz einen Wahrheitswert zuordnen. Diese Unsicherheit hinsichtlich der tatsächlichen Umstände wird im Rahmen der Sprachphilosophie durch das Konzept der **möglichen Welten** formalisiert, welches auf den Philosophen Gottfried Leibniz (1646–1716) zurückgeht und u.a. durch die Arbeiten des Philosophen David Lewis (1941–2001) Einzug in die Sprachwissenschaft gehalten hat. Jede mögliche Welt repräsentiert hierbei eine Möglichkeit, wie die Welt sein könnte, und für jede solche Welt lässt sich somit bestimmen, ob ein gegebener Satz bzgl. dieser Welt wahr oder falsch ist. Die Wahrheitsbedingungen eines Satzes lassen sich in diesem Rahmen also als Menge von möglichen Welten konkretisieren, nämlich als all die möglichen Welten, in denen der Satz wahr ist. Eine solche Menge von möglichen Welten nennt man **Proposition** und damit ist die Bedeutung eines Satzes eine Proposition. Hinsichtlich einer bestimmten gegebenen möglichen Welt lässt sich die Wahrheit eines Satzes feststellen, indem man überprüft, ob diese Welt in der durch den Satz ausgedrückten Proposition enthalten ist.

Wir haben also im Prinzip zwei Arten von Bedeutungen: eine „weltenübergreifende“ und eine bzgl. einer bestimmten Welt. Im Hinblick auf erstere denotiert ein Satz eine Proposition, im Hinblick auf letztere einen Wahrheitswert. Diese Unterscheidung von Bedeutungsarten ist auch in anderlei Hinsicht relevant und wir werden darauf in einem späteren Abschnitt zur Intensionalität zurückkommen. Im Moment genügt es, sich die Satzbedeutung bzgl. einer bestimmten Welt als Wahrheitswert vorzustellen.

Das Kompositionalsprinzip

Wenn nun die Bedeutung von Sätzen in einer Welt ihr Wahrheitswert ist, wie kann man dann Rückschlüsse auf die Bedeutung von Satzteilen ziehen? Das **Kompositionalsprinzip**, das Gottlob Frege zugeschrieben wird, leistet hier einen wichtigen Beitrag.

Kompositionalsprinzip:

Die Bedeutung eines komplexen Ausdrucks ist eine Funktion der Bedeutungen seiner Teile und der Art ihrer Kombination.

Zur Illustration dieses Prinzips betrachten wir die Interpretation prädikatenlogischer Formeln, da es hier bei der Definition der Junktoren eingehalten wurde. Für die Interpretation einer Konjunktion zweier Formeln φ und ψ gilt beispielsweise

$$\llbracket(\varphi \wedge \psi)\rrbracket^{\mathcal{M}, g} = f_{\wedge} (\llbracket\varphi\rrbracket^{\mathcal{M}, g}, \llbracket\psi\rrbracket^{\mathcal{M}, g}),$$

wobei f_{\wedge} die Funktion ist, die durch die Wahrheitswertetafel für die Konjunktion beschrieben wird (siehe Unterkapitel 2.1) – also $f_{\wedge}(x, y) = x \cdot y$ ist gerade das Produkt von x und y . Damit ist das Kompositionalsprinzip erfüllt, denn die

Bedeutung $\llbracket(\varphi \wedge \psi)\rrbracket^{\mathcal{M},g}$ des komplexen Ausdrucks $(\varphi \wedge \psi)$ ist die Funktion f_{\wedge} der Bedeutungen $\llbracket\varphi\rrbracket^{\mathcal{M},g}$ und $\llbracket\psi\rrbracket^{\mathcal{M},g}$ seiner Teile φ und ψ .

Allgemein gesprochen wird also das Problem der Bedeutungszuordnung komplexer Ausdrücke in leichter handhabbare Teilprobleme zerlegt. Im Rahmen der natürlichsprachlichen Semantik bedeutet dies, dass es zur Bestimmung der Bedeutung eines Satzes genügt, seine unmittelbaren Bestandteile, deren Bedeutung und die Funktion zu kennen, die diese Bedeutungen miteinander verknüpft.

Das Prinzip der Kompositionalität kann verschieden motiviert werden (vgl. Janssen 1997). Eines der wichtigsten Argumente dafür ist das der **Lernbarkeit**. Zunächst lässt sich feststellen, dass eine Sprache eine unendliche Menge von Sätzen bereitstellt (z.B. lässt sich jeder Satz durch anhängen von neuem Material in einen neuen, längeren Satz verwandeln). Es ist nun aber unplausibel anzunehmen, dass ein Sprecher dieser Sprache all diese Sätze bzw. deren Bedeutung in gewissem Sinne auswendig lernt, denn erstens wird er in der Phase des Spracherwerbs nur mit einer Auswahl aller möglichen Sätze konfrontiert und zweitens würde das schlicht unendlich viel Zeit in Anspruch nehmen. Daher muss man annehmen, dass er über einen Mechanismus verfügt, mit dem er eine endliche Anzahl von Wortbedeutungen zu unendlich vielen Satzbedeutungen gemäß dem Kompositionalitätsprinzip zusammensetzt. Was dieser Sprecher neben diesem Mechanismus also nur noch lernen muss, sind 1. die Wortbedeutung, also das Lexikon einer Sprache, 2. die syntaktischen Regeln zum Aufbau komplexer Ausdrücke und 3. die semantischen Regeln zur Errechnung der Bedeutung derselben. Das Kompositionalitätsprinzip erklärt somit auch, warum ein Sprecher Sätze verstehen kann, die er noch nie gehört hat – aus seinem vorhandenen Wissen über das Lexikon und die Regeln kann er sich deren Bedeutung berechnen. Dieser Berechnungsaspekt ist insbesondere auch für die Computerlinguistik von Relevanz.

Das Kompositionalitätsprinzip hat einige nichttriviale Konsequenzen:

- Jedes Wort, das als Bestandteil eines komplexen Ausdrucks vorkommt, trägt eine Bedeutung. Das heißt, neben Inhaltswörtern tragen auch Funktionswörter wie *nur* oder Interjektionen wie *ojeh* eine Bedeutung.
- Der semantische Interpretationsprozess (in der Computersemantik auch **Semantikkonstruktion** genannt) baut aus den Wortbedeutungen die Satzbedeutung auf, durchwandert den Syntaxbaum also bottom-up.
- Die Bedeutung eines sprachlichen Ausdrucks ergibt sich allein aus dem Ausdruck selber, und nicht aus dem Kontext, der den Ausdruck ergibt. Damit stellt die Bedeutungskomposition den Eckpfeiler für die Abgrenzung zwischen Semantik und Pragmatik dar.
- Die Semantikkonstruktion ist deterministisch, d.h. aufgrund der Syntaxanalyse eindeutig festgelegt.
- Eine kompositionale Semantik kommt nicht ohne Syntax aus. Die Eingabe für die Semantikkonstruktion ist also eine syntaktisch analysierte Wortketten.

Auf einen einfachen Satz wie *Willi schläft* angewendet, der aus der Nominalphrase *Willi*, einem Eigennamen, und der Verbalphrase *schläft* zusammengesetzt ist, besagt das Kompositionalitätsprinzip, dass man sich zunächst mit den Bedeutungen von *Willi* und *schläft* beschäftigen muss, um aus diesen Wortbedeutungen einen Wahrheitswert für den Gesamtsatz zu berechnen. Es bleibt darauf hinzuweisen, dass die Kompositionalität vom technischen Standpunkt aus eigentlich die Ausdrucksfähigkeit nicht beeinträchtigt, wohl aber zu einer Fülle von Ambiguitäten führt (z.B. die zur Erhaltung der Kompositionalität nötige Extralesart von „Nase“ in „jmdn. an der Nase herumführen“).

Referenz

Ein einfacher Vorschlag, einem Eigennamen wie *Willi* eine Bedeutung zuzuweisen, ist, den Eigennamen auf ein Individuum einer Domäne, also einer gegebenen Grundmenge von Entitäten, zu beziehen. Die Bedeutung eines Eigennamens ist also nichts anderes als ein Individuum und man sagt, dass der Name auf das Individuum **referiert**. In dieser Hinsicht verhält sich der natürlichsprachliche Eigenname wie ein Konstantensymbol in der Prädikatenlogik, das mittels der Interpretationsfunktion auch auf ein Individuum der Domäne abgebildet wird.

Allerdings benutzen wir Namen nicht nur zur Bezeichnung von Personen, sondern auch für Städte, Staaten, Flüsse, Seen, Meere, Berge, Himmelskörper und Sternbilder genauso wie für abstrakte oder in der tatsächlichen Welt nicht-existente Objekte (*Pegasus*, *Phönix*, *Atlantis*, *Zeus*). Nach obigem Vorschlag muss die Domäne also entsprechende Individuen für die Referenz all dieser Ausdrücke bereitstellen.

Ein weiterer Typ von natürlichsprachlichen Ausdrücken, mit denen man sich auf einzelne Individuen beziehen kann, sind **definite Kennzeichnungen** wie *der Präsident der USA*, *die Königin von England*, oder *der Schüler*, die im Deutschen primär durch Nominalphrasen mit definitem Artikel gebildet werden. Eine solche definite Kennzeichnung referiert auch auf ein Individuum, wobei es allerdings zwei augenscheinliche Probleme gibt.

Existenz. Auf was referiert die Kennzeichnung, wenn es kein Objekt gibt, auf das die Kennzeichnung zutrifft? Beispielsweise macht die definite Kennzeichnung *Der König von Frankreich* im Jahr 2009 in gewisser Weise keinen Sinn, da es kein Individuum zu diesem Zeitpunkt gibt, dem wir die Eigenschaft, König von Frankreich zu sein, zuschreiben könnten.

Einzigkeit. Auf was referiert die Kennzeichnung, wenn es mehrere in Frage kommende Objekte gibt, auf die die Kennzeichnung zutreffen könnte? Die Kennzeichnung *der Bundesminister* kann beispielsweise für den Innenminister der Bundesregierung im Jahre 2009 stehen, oder für den Außenminister des Jahres 1990 oder für viele andere mehr. Ohne weitere Qualifizierung oder Hinweise aus dem Kontext ist die Einzigartigkeit und damit die Referenz der Kennzeichnung nicht gewährleistet.

Wie genau diese beiden Aspekte der Existenz und Einzigartigkeit zu behandeln sind und ob sie eher als Teil der Bedeutung in der Semantik oder als Präsupposition

tionen in der Pragmatik behandelt werden sollten, war und ist eine oft diskutierte Frage im Rahmen der Sprachphilosophie. Sie wurde initial von Bertrand Russell (1872–1970, als Vertreter der semantischen Sichtweise) und von Peter Strawson (1919–2006, als Vertreter der eher pragmatischen Sichtweise) Mitte des 20. Jahrhunderts geführt und wir werden später auf sie zurückkommen.

Referenz muss nicht eindeutig sein, d.h. auf ein und dasselbe Individuum kann mit unterschiedlichen Ausdrücken referiert werden. Beispielsweise könnte *Willlein* ein Spitzname für *Willi* sein und auf dasselbe Individuum referieren, genauso wie *Obervolta* und *Burkina Faso* denselben Staat in Westafrika bezeichnen. Genauso beziehen sich *Die Königin von England im Jahr 2008* als auch *Queen Elisabeth II.* auf die selbe Person. Solche Ausdrücke mit gleichem Referenten nennt man **koreferent**.

Quantoren

Nachdem wir nun grundlegende Punkte der Bedeutungbestimmung von Eigennamen und definiten Kennzeichnungen erläutert haben, wollen wir die Diskussion auf andere Formen von Nominalphrasen ausdehnen. Was ist die Bedeutung von Nominalphrasen wie *jeder Mann*, *kein Hund*, *ein Student*, etc.? Wenn man auf die Behandlung von Eigennamen sieht, könnte man in Versuchung geraten, die Bedeutung solcher Nominalphrasen als Individuenmengen festzulegen. Für eine quantifizierende NP wie *jeder Mann* würde das vielleicht Sinn machen, denn man könnte festlegen, dass ihre Bedeutung gerade die Menge aller Individuen ist, die die Nomen-Eigenschaft des *Mann*-seins haben. Aber was wäre dann die Bedeutung von *ein Student*? Wenn wir dafür eine einelementige Menge vorschlagen würden, die ein Individuum mit *Student*-Eigenschaft hat, stellte sich immer noch die Frage, welche solche Menge, d.h. welches Individuum dies sein sollte. Außerdem wären z.B. die Bedeutungen der Ausdrücke *kein Hund* und *keine Katze* identisch, da beide die leere Menge denotierten, was zu unintuitiven Resultaten führen würde. Mit dieser Festlegung könnte man z.B. folgenden unsinnigen Schluss ziehen:

(3.54) *Keine Katze bellt. ↔ Kein Hund bellt.*

Wie man sieht, ist es notwendig, für solche Nominalphrasen von Bedeutungen auszugehen, die komplexer als Mengen von Individuen sind. Stattdessen bedient man sich der Quantoren der Logik. Die Übersetzungen der Sätze *Keine Katze bellt* und *Ein Student schläft* in die Prädikatenlogik sind wie folgt.

(3.55) $\neg\exists x(\text{katze}(x) \wedge \text{bellen}(x))$ (*Keine Katze bellt*)

(3.56) $\exists x(\text{student}(x) \wedge \text{schlafen}(x))$ (*Ein Student schläft*)

Die Bedeutung der quantifizierenden NP geht also über einfache Referenz hinaus und wird später in diesem Kapitel ausführlicher behandelt.

Ein wichtiger Punkt betrifft jedoch das Zusammentreffen mehrerer solcher Nominalphrasen. Wenn mehrere Quantoren in einem Satz auftreten, kann dies zu Ambiguitäten führen. Der Satz

(3.57) *Jeder Student kennt ein Buch.*

bedeutet entweder, dass die Studenten jeweils verschiedene Bücher kennen oder dass es ein bestimmtes Buch gibt, das alle Studenten kennen.

In der Semantik werden die verschiedenen Lesarten durch die Stellungen der Quantoren in der Formel und damit durch ihre Skopusverhältnisse ausgedrückt. Daher nennt man diese Art von Ambiguität auch **Skopusambiguität**. Die folgenden Übersetzungen in die Prädikatenlogik zeigen die beiden Lesarten, indem der Existenz- und der Allquantor jeweils unterschiedliche Skopuspositionen einnehmen.

1. $\forall x(\text{student}(x) \rightarrow \exists y(\text{buch}(y) \wedge \text{kennen}(x, y)))$
2. $\exists y(\text{buch}(y) \wedge \forall x(\text{student}(x) \rightarrow \text{kennen}(x, y)))$

Erinnert man sich an das Kompositionalsprinzip, so gibt es an dieser Stelle scheinbar ein Problem. Wie ist es nämlich möglich, dass ein einzelner Satz mehrere Bedeutungen haben kann, wo seine Bedeutung doch durch seine Teile und die Art ihrer Kombination eindeutig bestimmt ist? Die Antwort auf diese Frage liegt in verschiedenen zugrundeliegenden Strukturen, die man einem ambigen Satz wie (3.57) zuweist und die dann gemäß dem Kompositionalsprinzip interpretiert werden können.

Der Grad der Mehrdeutigkeit eines Satzes, d.h. die Anzahl der möglichen Lesarten, steigt rapide mit der Anzahl der darin vorkommenden Quantoren an. Dies ist insbesondere im Rahmen der Computersemantik problematisch, bei der man an möglichst effizienter Verarbeitung interessiert ist. Wir werden diese Aspekte in Abschnitt 3.6.4 über Unterspezifikation näher diskutieren.

Intensionale Bedeutung

Wie oben diskutiert gibt es im Prinzip zwei Betrachtungsweisen bezüglich der Bedeutung von Aussagesätzen. Zum einen kann die Bedeutung eines Satzes als seine Wahrheitsbedingungen, und damit als Proposition (d.h. Menge von möglichen Welten), verstanden werden, zum anderen kann seine Bedeutung bzgl. einer Welt als Wahrheitswert in dieser Welt betrachtet werden. Eine ähnliche Unterscheidung ergibt sich im Rahmen der Referenz, denn auch hier beobachtet man, dass es zwei ähnliche Arten von Bedeutung geben kann.

Die Referenz allein legt die Bedeutung nämlich nicht vollständig fest; Bedeutung ist mehr als Referenz. Früher glaubte man, dass (3.58) falsch sei.

(3.58) *Der Abendstern ist der Morgenstern.*

Dann fand man heraus, dass das mit Abendstern bezeichnete Objekt als auch das mit Morgenstern bezeichnete Objekt der Planet Venus ist, womit also Abendstern und Morgenstern koreferent sind – sie referieren beide auf den Planeten Venus. Trotzdem kann man (3.58) für falsch halten, wenn man sich etwa dieser Tatsache nicht bewusst ist. (3.59) hingegen ist eine Tautologie und muss deswegen zwingend immer für wahr gehalten werden.

(3.59) *Der Abendstern ist der Abendstern.*

Betrachtet man nur die Referenz, so drücken beide Sätze exakt dasselbe aus. Der Philosoph und Mathematiker Gottlob Frege unterscheiden daher den referentiellen Bedeutungsanteil, der sich aus den Dingen ergibt, die ein Ausdruck in einer möglichen Welt bezeichnet, von einem weiteren Bedeutungsanteil, nämlich der Art, wie der Ausdruck das Individuum, auf das er referiert, präsentiert. Die erste Art der Bedeutung eines Ausdrucks nennt man **Extension** (**Bedeutung** bei Frege) die zweite Art **Intension** (**Sinn** bei Frege).

Beispiel 3.6.1

Im Falle von (3.58) und (3.59) könnte man sich diesen Unterschied folgendermaßen deutlich machen:

Intension		Extension
Morgenstern	hellster Stern am Morgenhimmel	Venus
Abendstern	hellster Stern am Abendhimmel	Venus

Man sieht, dass sich die Intensionen der beiden Ausdrücke durchaus unterscheiden, die Extension (bzw. Referenz) aber dieselbe ist. Deshalb war es früher den Menschen möglich (3.58) für falsch und gleichzeitig (3.59) für wahr zu halten. □

Die Intension ist ein Kriterium, mit dem die Extension in einer bestimmten Situation ermittelt werden kann. Im Fall der Aussagesätze von oben ist also die Intension eines Satzes eine Proposition, während die Extension eines Satzes bzgl. einer möglichen Welt sein Wahrheitswert in dieser Welt ist. Die Extension bzgl. einer Welt lässt sich aus der Intension dadurch ermitteln, dass man überprüft, ob diese Welt ein Element der Proposition ist. Man beobachtet, dass, wann immer zwei Ausdrücke eine identische Intension haben, sie auch identische Extensionen haben.

Extensionen allein reichen für eine zufriedenstellende kompositionale Semantik im Hinblick auf manche Phänomene nicht aus. In bestimmten Konstruktionen können Ausdrücke nicht ausgetauscht werden, auch wenn sie dieselbe Extension haben (so genannte **opake Kontexte**), wie folgende Beispiele zeigen:

(3.60) *Fritz glaubt, dass der Abendstern der Abendstern ist.*

(3.61) *Fritz glaubt, dass der Abendstern der Morgenstern ist.*

Betrachten wir zunächst die beiden eingebetteten Sätze, die Fritz glauben soll. Wie oben erwähnt, ist (*dass*) *der Abendstern der Abendstern ist* in (3.60) tautologisch, also in jeder Welt wahr. Damit drückt er die Proposition aus, die sämtliche möglichen Welten umfasst. Der eingebettete Satz in (*dass*) *der Abendstern der Morgenstern ist* in (3.61) ist hingegen nur in manchen Welten wahr – es sind durchaus mögliche Welten vorstellbar, in denen der Abendstern und der Morgenstern nicht koreferent sind – und damit ist die durch ihn ausgedrückte Proposition nur eine Teilmenge aller möglichen Welten. Damit sind also die Intensionen der beiden Sätze unterschiedlich. Die Extensionen bzgl. der aktuellen

Welt, wie wir sie vorfinden, sind allerdings identisch: beide eingebetteten Sätze sind wahr. Wäre nun zur Bestimmung der Wahrheitswerte der Gesamtsätze in (3.60) und (3.61) nur die Extension der eingebetteten Sätze relevant, müssten in der aktuellen Welt beide zusammen wahr oder zusammen falsch sein. Dem ist aber nicht so, denn *glauben* erzeugt einen opaken Kontext und somit ist für die Bestimmung der Wahrheitswerte der Gesamtsätze die Intension der Einbettungen relevant. Da diese sich unterscheiden, kann (3.60) wahr und gleichzeitig (3.61) falsch sein.

Weitere Beispiele für opake Kontexte sind Zitate, indirekte Rede, propositionale Einstellungen (d.h. Objektsätze von Verben wie *glauben*, *entdecken*, *vermuten*, *wissen*) und Modaladverbien (*notwendigerweise*, *angeblich*, *anscheinend*). Insbesondere unterscheidet man zwischen **intensionalen Verben**, deren Objekte intensional verstanden werden müssen, und **extensionalen Verben**. Der Unterschied zeigt sich zwischen *suchen* als intensionalem und *finden* als extensionalem Verb, wie folgendes Satzpaar zeigt:

(3.62) *Fritz sucht das letzte Einhorn.*

(3.63) *Fritz findet das letzte Einhorn.*

Während aus der Wahrheit von (3.63) folgt, dass das letzte Einhorn tatsächlich existiert, folgt dies nicht aus der Wahrheit von (3.62). In anderen Worten: gibt es kein letztes Einhorn, so kann Fritz es trotzdem suchen, womit (3.62) durchaus wahr sein kann. Finden kann er es in diesem Fall hingegen nicht, weswegen (3.63) falsch ist. Im Folgenden werden wir uns auf die einfachere, extensionale Sichtweise beschränken und entsprechend auch die Behandlung von opaken Kontexten und intensionalen Verben außen vor lassen.

3.6.2 Formale Semantik

Nach dieser eher informellen Darstellung grundlegender Begriffe der Semantik, wenden wir uns der Formalisierung zu. Eine der einflussreichsten Theorien der formalen Satzsemantik, an der wir uns zunächst orientieren werden, ist die **Montague-Semantik**. Der Ansatz basiert auf den Arbeiten des amerikanischen Logikers und Sprachtheoretikers Richard Montague (1930–1971), die er in den drei Aufsätzen *English as a Formal Language* (Montague 1970a), *Universal Grammar* (Montague 1970b) und, am einflussreichsten, *The Proper Treatment of Quantification* (Montague 1973) niedergelegt hat.

Eine herausragende Eigenschaft der Montague-Semantik ist ihre systematische Herangehensweise bei der Bedeutungszuordnung sprachlicher Ausdrücke. Die Montague-Semantik zeigt auf, wie Syntax und Semantik mithilfe der Methoden der mathematischen Logik systematisch verbunden werden können. Eine modelltheoretische Interpretation natürlicher Sprache ist insbesondere deshalb möglich, weil – laut Montague – zwischen natürlichen und formalen Sprachen kein wesentlicher Unterschied besteht.

Direkte und indirekte Deutung

Natürlichsprachliche Ausdrücke können entweder direkt modelltheoretisch interpretiert werden (**direkte Deutung**) oder zunächst in eine formale ZwischenSprache *übersetzt* werden, welche selbst wiederum modelltheoretisch interpretiert wird (**indirekte Deutung**). Solange sowohl Übersetzungsschritt und Interpretation kompositional sind, gibt es keinen formalen Unterschied zwischen den beiden Ansätzen, d.h. auf die ZwischenSprache kann ohne Einbußen in der Mächtigkeit verzichtet werden. Zwischen logisch äquivalenten Repräsentationen wird auch auf der Zwischenebene nicht unterschieden. Montague hat beide Verfahren in seinen Aufsätzen untersucht (direkte Deutung: Montague 1970a, indirekte Deutung: Montague 1970b, Montague 1973).

In der Computersemantik spricht eine Reihe von Gründen für die indirekte Deutung: Die Zwischenrepräsentation ist eine weitere Schnittstelle, die bei der Fehlersuche willkommen ist und das System modularer macht. Die Zwischenrepräsentation abstrahiert von vielen sprachspezifischen Merkmalen, bewahrt aber immer noch einen gewissen Einblick in die syntaktische Struktur. Sie ist daher besonders gut geeignet als Eingabe für Prozesse, die zwar eine abstrakte Repräsentation voraussetzen, aber aus Effizienzgründen auf strukturierte Information angewiesen sind (z.B. Generierung (vgl. Unterkapitel 3.8) oder maschinelle Übersetzung (vgl. Unterkapitel 5.7)). Letztendlich beschränkt die Zwischenrepräsentation den Suchraum für semantikverarbeitende Anwendungen, was seine Vor- und Nachteile hat: Auf der einen Seite ist das Auswahlproblem wesentlich einfacher und viele pragmatische Aspekte werden automatisch richtig erfasst, auf der anderen Seite fehlt vielleicht gerade die Repräsentation, die etwa aufgrund sprachlicher Constraints der Zielsprache benötigt wird.

Im Folgenden wird insbesondere die Lambda-Typenlogik als Zwischenrepräsentation zum Einsatz kommen, die in Unterkapitel 2.1 eingeführt wurde. Um präzise vorgehen zu können, benutzen wir für die Übersetzung in die Zwischenrepräsentation eine eigene Schreibweise.

Definition 3.6.1

Ist A ein natürlichsprachlicher Ausdruck, dann bezeichnet $\langle\!\langle A \rangle\!\rangle$ seine Übersetzung in die Zwischenrepräsentation der Lambda-Typenlogik. \square

Prädikation und Modifikation

Wir starten mit einem denkbar einfachen Beispiel, nämlich dem Satz

(3.64) *Willi schläft.*

Bevor wir zu einer semantischen Analyse des Satzes kommen, müssen wir uns Gedanken um seinen syntaktischen Aufbau machen. Wir gehen hierbei von einer sehr einfachen Syntax aus, die nicht die für adäquate linguistische Analysen notwendige Komplexität besitzt, aber anhand derer sich die Semantikkonstruktion anschaulich beschreiben lässt. Eine solch einfache syntaktische Analyse für

(3.64) könnte so aussehen, dass sich die NP *Willi* mit der VP *schläft* zu einem Satz S verbindet: [S [NP *Willi*][VP *schläft*]]

Was die Semantik betrifft, so bedeutet (3.64) intuitiv gesprochen, dass Willi die Eigenschaft hat zu schlafen. Der Satz ist somit ein Beispiel für eine Eigenschaftszuschreibung, auch **Prädikation** genannt. Genauer gesagt handelt es sich um eine **verbale Prädikation**, denn die Eigenschaft wird durch ein intransitives Verb ausgedrückt.

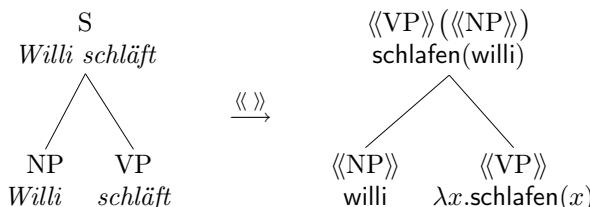
In Abschnitt 3.6.1 hatten wir argumentiert, dass Eigennamen wie *Willi* auf Individuen referieren. Im Hinblick auf die Übersetzung in die Zwischensprache der Lambda-Typenlogik bedeutet das, dass wir die NP *Willi* mittels einer Individuenkonstante vom Typ e repräsentieren, z.B. $\langle\!\langle \text{Willi} \rangle\!\rangle = \text{willi}$. Das intransitive Verb sollte also dementsprechend eine Eigenschaft ausdrücken, die in Kombination mit der Bedeutung des Subjekts eine Satzbedeutung, also einen Wahrheitswert ergibt. Damit muss die Übersetzung von *schläft* also vom Typ $\langle e, t \rangle$ sein, z. B. $\langle\!\langle \text{schläft} \rangle\!\rangle = \lambda x.\text{schlafen}(x)$ (zur Steigerung der Lesbarkeit trennen wir hier und im Folgenden den Präfix der Lambdaoperatoren durch einen Punkt von ihrem Skopuss ab).

Diese beiden Bedeutungen können nun kombiniert werden, in dem die Bedeutung der VP auf die Bedeutung des Subjekts angewendet wird. Diese Anwendung einer Funktion auf ihr Argument wird auch als **Funktionalapplikation** bezeichnet. Im Ergebnis bekommt man damit nach Vereinfachung mittels Lambda-Konversion $\lambda x.\text{schlafen}(x)(\text{willi}) = \text{schlafen}(\text{willi})$. Diese Formel repräsentiert nun die Bedeutung des gesamten Satzes und könnte bzgl. eines Modells interpretiert werden (vgl. Unterkapitel 2.1).

An dieser Stelle sollten wir noch einmal kontrollieren, ob wir tatsächlich dem Kompositionalsprinzip gefolgt sind. Dazu müsste die Bedeutung des Satzes eine Funktion der Bedeutung seiner Teile und der Art ihrer Kombination sein. Die Teile des Satzes sind in unserer einfachen syntaktischen Analyse eine NP und eine VP. Die Bedeutung der NP ist *willi* und die Bedeutung der VP $\lambda x.\text{schlafen}(x)$. Die Bedeutung des gesamten Satzes ist eine Funktion dieser beiden Bedeutungen, nämlich deren Funktionalapplikation

$$(3.65) \langle\!\langle \text{S} \rangle\!\rangle = \langle\!\langle \text{VP} \rangle\!\rangle (\langle\!\langle \text{NP} \rangle\!\rangle)$$

Damit ist das Kompositionalsprinzip erfüllt. Etwas allgemeiner ausgedrückt, berechnet sich also die Bedeutung eines aus NP und VP bestehenden Satzes als Funktionalapplikation der VP-Bedeutung auf die NP-Bedeutung, wie im Folgenden nochmals illustriert. Dass hierbei der syntaktische Strukturbaum links und der semantische Ableitungsbau rechts eine identische Struktur besitzen, ist charakteristisch für eine kompositionale Semantikberechnung.



Eine andere Art der Eigenschaftszuschreibung ist die **nominale Prädikation**, bei der die Eigenschaft über ein Nomen ausgedrückt wird. Ein Beispiel hierfür liefert

(3.66) *Willi ist Philosoph.*

Hier wird Willi die Eigenschaft zugeschrieben, Philosoph zu sein. Wie oben referiert *Willi* wieder auf ein Individuum. Das Nomen *Philosoph* wird nun wieder als eine einfache Eigenschaft analysiert, nämlich als die Eigenschaft, *Philosoph* zu sein. Damit wird ein Nomen auch mit einem Ausdruck des Typs $\langle e, t \rangle$ übersetzt, in diesem Falle z.B. $\langle\langle \text{Philosoph} \rangle\rangle = \lambda x.\text{philosoph}(x)$. Was die Typen betrifft, ließe sich nun die Übersetzung des Nomens wie im vorigen Fall auf das Subjekt-individuum anwenden, aber welchen Beitrag leistet dann die sogenannte **Kopula ist?** Wie in Abschnitt 3.6.1 zu den Konsequenzen der Kompositionnalität schon angemerkt, muss jeder Ausdruck, der als Bestandteil eines komplexeren Ausdrucks vorkommt, eine Bedeutung tragen. Wenn wir also annehmen, dass die syntaktische Struktur etwa derart ist, dass sich die Kopula *ist* mit dem prädikativen Nomen zu einer VP verbindet, die dann ihrerseits mit der Subjekts-NP einen Satz formt, so müssen wir auch dieser Kopula eine Bedeutung zuweisen. Das ist jedoch einfacher, als es scheinen mag, denn wir geben ihr schlicht eine Bedeutung ohne semantischen Effekt:

(3.67) $\langle\langle \text{ist} \rangle\rangle = \lambda P \lambda x. P(x)$

Wenn wir wieder Funktionalapplikation des Verbs (hier: der Kopula) auf ihr Komplement (hier: das prädiktative Nomen) als semantischen Kombinationsmodus annehmen, dann errechnet sich die Bedeutung der VP *ist Philosoph* als

$$\langle\langle \text{ist} \rangle\rangle(\langle\langle \text{Philosoph} \rangle\rangle) = \lambda P \lambda x. P(x)(\lambda x.\text{philosoph}(x)) = \lambda x.\text{philosoph}(x)$$

In gewissem Sinne ist die Bedeutung der Kopula also so angelegt, dass sie die Bedeutung ihres Arguments (hier: des prädiktativen Nomens) einfach weiterreicht ohne sie zu verändern. Die Errechnung der Bedeutung des Gesamtsatzes (3.66) geschieht wieder wie oben durch Funktionalapplikation der VP-Bedeutung auf die Subjekts-NP-Bedeutung, was *philosoph(willi)* ergibt.

Ein ähnlicher Fall betrifft die Prädikation mittels eines **prädiktativen Adjektivs** wie in

(3.68) *Willi ist fröhlich.*

Auch hier scheint es Sinn zu machen, das Adjektiv zunächst als Eigenschaft vom Typ $\langle e, t \rangle$ zu betrachten, also $\langle\langle \text{fröhlich} \rangle\rangle = \lambda x.\text{fröhlich}(x)$. Nimmt man wieder die semantische leere Interpretation der Kopula wie zuvor an, ergibt sich eine analoge Ableitung für (3.68).

$$\begin{aligned}\langle\langle \text{Willi ist fröhlich} \rangle\rangle &= (\langle\langle \text{ist} \rangle\rangle(\langle\langle \text{fröhlich} \rangle\rangle))(\langle\langle \text{Willi} \rangle\rangle) \\ &= (\lambda P \lambda x. P(x)(\lambda x.\text{fröhlich}(x)))(\text{willi}) \\ &= \text{fröhlich}(\text{willi})\end{aligned}$$

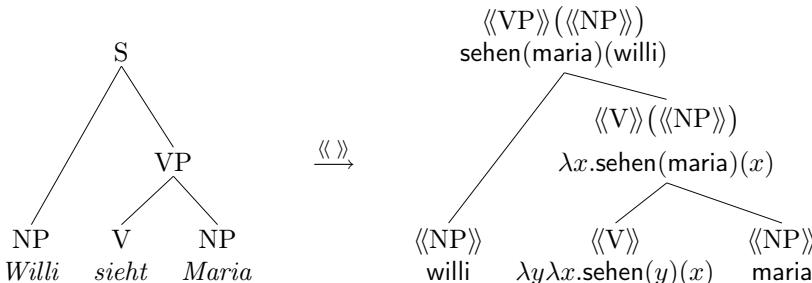
Bislang hatten wir nur Sätze betrachtet, in denen genau einem Objekt eine Eigenschaft zugeschrieben wurde. Solche Zuschreibungen können aber auch komplexer werden, wenn man z. B. transitive Verben untersucht.

(3.69) *Willi sieht Maria*

Intuitiv schreibt dieser Satz zwei Individuen die Eigenschaft zu, in einer sehen-Beziehung zueinander zu stehen, wobei Willi derjenige ist, der Maria sieht. Wir gehen von der folgenden einfachen syntaktischen Struktur aus, bei der das transitive Verb *sieht* mit seinem Komplement *Maria* eine VP bildet, die dann wieder mit der Subjekts-NP einen Satz formt.

(3.70) [S [NP *Willi*] [VP [V *sieht*] [NP *Maria*]]]

Die Bedeutung des transitiven Verbs *sieht* ist vom Typ $\langle e, \langle e, t \rangle \rangle$. Sie verlangt nach einem Individuum (dem Objekt), um ein einfaches Prädikat zu formen, das auf ein weiteres Individuum (das Subjekt) angewendet einen Wahrheitswert liefert. Im Falle von (3.69) hätten wir also $\langle\langle \text{sieht} \rangle\rangle = \lambda y \lambda x. \text{sehen}(y)(x)$. Die Berechnung der Gesamtbedeutung erfolgt nun mittels zweifacher Funktionalapplikation der Verbbedeutung auf Objekt- bzw. Subjektbedeutung. Wir geben die Ableitung wieder in einer Baumstruktur an, um die Kompositionalität der Berechnung zu verdeutlichen.



Neben der Prädikation als Eigenschaftszuschreibung gibt es noch ein weiteres semantisches Verfahren, nämlich die **Modifikation**. Anders als bei der Prädikation bleibt bei der Modifikation der grundlegende Charakter der modifizierten Bedeutung erhalten, wobei der Modifikator diese in gewisser Hinsicht anreichert oder abändert. Diese Zweiteilung spiegelt sich auch im Hinblick auf die notwendige Präsenz von semantischen Bedeutungskomponenten wider. Während **Komplemente** notwendige Bestandteile zur Berechnung einer Gesamtbedeutung darstellen, sind **Adjunkte** optional und liefern einen zusätzlichen Beitrag. Entsprechend dienen Komplemente häufig als Argumente in einer Prädikation, während Adjunkte ihren Bedeutungsbeitrag via Modifikation der Ursprungsbedeutung leisten. Die folgenden Sätze illustrieren diese Begriffe.

(3.71) *Willi sieht Maria mit dem Fernglas.*

(3.72) * *Willi sieht (mit dem Fernglas).*

(3.73) *Willi sieht Maria.*

Da Komplemente notwendige Bestandteile zur Bedeutung liefern, sind sie nicht weglassbar, während Adjunkte als optionale Bestandteile weggelassen werden können. Wie man anhand eines Vergleichs der Grammatikalität von (3.71)–(3.73) erkennt, ist *Maria* als direktes Objekt ein Komplement des transitiven Verbs, während die Präpositionalphrase *mit dem Fernglas* ein Adjunkt ist.

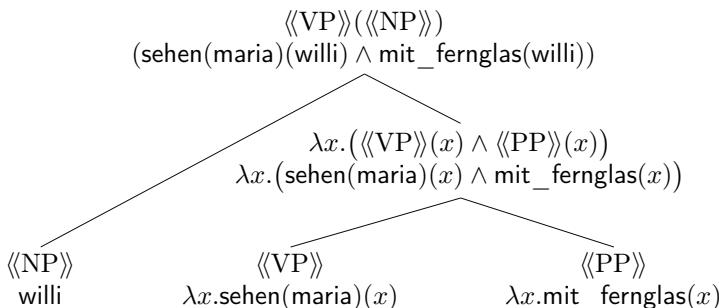
Was die formale Analyse betrifft, so gehen wir zunächst vereinfachend davon aus, dass die Bedeutung der PP durch eine einfache Eigenschaft vom Typ $\langle e, t \rangle$ (in etwa die Eigenschaft, mit einem Fernglas ausgestattet zu sein) gegeben ist: $\langle\langle \text{mit dem Fernglas} \rangle\rangle = \lambda x. \text{mit_fernglas}(x)$. Weiterhin nehmen wir an, dass die PP an die VP *sieht Maria* angebunden ist. Damit wird bei der kompositionellen Berechnung der Satzbedeutung der semantische Beitrag der PP an dieser Stelle einfließen, d.h. die PP wird die VP-Bedeutung modifizieren (eine andere Möglichkeit wäre, dass die PP an die Objekt-NP angebunden wäre und die PP somit Maria modifizieren würde; s. auch die Erläuterungen zur PP-Anbindung im Syntax-Unterkapitel 3.5). Wir gehen also von folgender syntaktischen Struktur aus:

$$(3.74) [\text{s} [\text{NP } \text{Willi }][\text{VP } [\text{VP } [\text{V } \text{sieht }][\text{NP } \text{Maria }]][\text{PP } \text{mit dem Fernglas }]]]$$

Die Bedeutung der VP *sieht Maria* hatten wir oben als $\lambda x. \text{sehen}(\text{maria})(x)$ aus der Funktionalapplikation der Bedeutung des Verbs auf die Bedeutung des Objekts errechnet. Nun stellt sich die Frage, wie sich die PP-Bedeutung mit dieser verknüpft. Die Antwort liefert eine Modifikationsregel für die Berechnung der modifizierten VP, die die VP-Bedeutung durch Konjunktion mit der PP-Bedeutung kombiniert:

$$(3.75) \lambda x. (\langle\langle \text{VP} \rangle\rangle(x) \wedge \langle\langle \text{PP} \rangle\rangle(x))$$

Wir geben die Ableitung hier wieder in (leicht verkürzter) Baumdarstellung:



Die Modifikationsregel stellt somit eine andere semantische Kombinationsmöglichkeit als die Funktionalapplikation dar. Ihre Formalisierung in (3.75) zeigt nochmals die oben erwähnten Eigenschaften: die Bedeutung des modifizierten Ausdrucks (hier die zur VP *sieht Maria* gehörige Eigenschaft vom Typ $\langle e, t \rangle$) bleibt dem Charakter nach erhalten (d.h. das Resultat der Modifikation ist wieder eine Eigenschaft vom selben Typ) und wird nur angereichert (um die Konjunktion mit dem Bedeutungsbestandteil der PP).

Die endgültige Repräsentation ist in mancherlei Hinsicht allerdings noch unbefriedigend. Die Modifikation durch die Eigenschaft $\lambda x.\text{mit_fernglas}(x)$ betrifft dort nämlich Willi, aber tatsächlich wird in (3.71) das eigentliche Ereignis des „Maria-Sehens“ durch die zusätzliche Information *mit dem Fernglas* qualifiziert. Um eine solche genauere Formalisierung zu erreichen, ist es notwendig, Ereignisse (engl. *events*) explizit über Variablen zu repräsentieren. Im Rahmen dieses Kapitels können wir auf derartige Erweiterungen leider nicht eingehen.

Ein anderer Fall von Modifikation ergibt sich mit **attributiven Adjektiven**. Das sind solche Adjektive, die innerhalb einer NP ein Nomen modifizieren, wie im folgenden Beispiel:

(3.76) *Ein fröhlicher Hund bellt.*

Was die syntaktische Struktur betrifft, so nehmen wir an, dass das Adjektiv als Attribut zum Nomen fungiert, also dass innerhalb der NP folgende Struktur vorliegt: $[N[Adj\ fröhlicher][N\ Hund]]$. Da wir die Bedeutung von Determinierern wie *ein* erst im nächsten Abschnitt besprechen werden, konzentrieren wir uns hier nur auf die Modifikation der Bedeutung von *Hund* mittels des Adjektivs. Auch hier kommt die Modifikationsregel wieder zum Einsatz, die die beiden Bedeutungsbeiträge durch Konjunktion kombiniert, so dass sich folgende Ableitung ergibt.

$$\begin{aligned}\langle\!\langle\text{fröhlicher Hund}\rangle\!\rangle &= \lambda x.(\langle\!\langle\text{fröhlicher}\rangle\!\rangle(x) \wedge \langle\!\langle\text{Hund}\rangle\!\rangle(x)) \\ &= \lambda x.(\text{fröhlich}(x) \wedge \text{hund}(x))\end{aligned}$$

Aus der Eigenschaft, ein Hund zu sein, wird also durch Modifikation die neue Eigenschaft, ein Hund und fröhlich zu sein.

Diese Form der adjektivischen Modifikation durch Konjunktion liefert leider nur für eine Teilklasse der Adjektive die korrekte Bedeutung. Würde man etwa für ein Adjektiv wie *angebliche* eine Grundbedeutung $\lambda x.\text{angeblich}(x)$ annehmen und diese auf dieselbe Weise mit dem modifizierten Nomen verbinden, so ergäbe sich beispielsweise für *angebliche Dieb* die Bedeutungsrepräsentation

$$\langle\!\langle\text{angebliche Dieb}\rangle\!\rangle = \lambda x.(\text{angeblich}(x) \wedge \text{dieb}(x))$$

Hätte z.B. Willi diese Eigenschaft, so wäre Willi ein Dieb und angeblich. Das ist natürlich unsinnig, denn 1. macht es keinen Sinn von einem Individuum zu sagen, dass es angeblich ist, und 2. hat ein *angeblicher Dieb* eben gerade nicht die Eigenschaft, Dieb zu sein. Das Adjektiv *angeblich* muss also anders behandelt werden, nämlich als Prädikat über Nomen-Bedeutungen: $\langle\!\langle\text{angeblich}\rangle\!\rangle = \lambda P \lambda x. \text{angeblich}(P)(x)$. Die Kombination mit dem Nomen erfolgt dann über Funktionalapplikation der Adjektivbedeutung auf die Nomenbedeutung und wir erhalten das bessere:

$$\langle\!\langle\text{angebliche Dieb}\rangle\!\rangle = \lambda x. \text{angeblich}(\text{dieb})(x)$$

Wir müssen also zwei Klassen von Adjektiven unterscheiden, nämlich solche, die ihren Bedeutungsbeitrag durch Konjunktion mit der Nomenbedeutung leisten, und solche, die dies eher prädikativ bzgl. der Nomenbedeutung tun. Erstere

nennt man **intersektive Adjektive**, denn die Menge der Individuen mit der resultierenden Eigenschaft ist die Schnittmenge (Intersektion) der Menge der Individuen mit der Adjektiveigenschaft und der Menge der Individuen mit der Nomeneigenschaft. Letztere heißen **skopale Adjektive**, da sie zu Skopusambiguitäten führen können (z.B. *angebliche Dieb aus Hannover*).

Zusammenfassend lässt sich sagen, dass wir zwei semantische Verfahren, die Prädikation und die Modifikation, und zwei semantische Verknüpfungsregeln kennengelernt haben, die Funktionalapplikation und die Modifikationsregel. Letztere sind hier noch einmal allgemein für die Kombination zweier beliebiger Bedeutungen α und β (von passendem Typ) notiert:

$$(3.77) \text{ **Funktionalapplikationsregel:** } FA(\alpha, \beta) = \alpha(\beta)$$

$$(3.78) \text{ **Modifikationsregel:** } Mod(\alpha, \beta) = \lambda x. (\alpha(x) \wedge \beta(x))$$

Damit können wir noch einmal das Kompositionalsprinzip am konkreten Fall der Bedeutungsberechnung von Satz (3.64) verdeutlichen, das besagt, dass die Bedeutung des Satzes (also $\langle\langle Willi schläft\rangle\rangle$) eine Funktion (hier FA) der Bedeutung seiner Teile (also $\langle\langle schläft\rangle\rangle$ und $\langle\langle Willi\rangle\rangle$) ist:

$$(3.79) \langle\langle Willi schläft\rangle\rangle = FA(\langle\langle schläft\rangle\rangle, \langle\langle Willi\rangle\rangle)$$

Quantoren

Bislang hatten wir als Nominalphrasen nur Eigennamen betrachtet, die auf ein Individuum referieren. In der Diskussion der Grundlagen in Abschnitt 3.6.1 haben wir bereits angedeutet, dass komplexere Nominalphrasen wie *jeder Mann*, *keine Katze* oder *ein Student* nach einer komplexeren Bedeutungsdefinition verlangen. Solche Ausdrücke nennt man **quantifizierende Nominalphrasen**, denn sie machen in gewissem Sinne eine quantitative Aussage. Betrachten wir folgenden Satz:

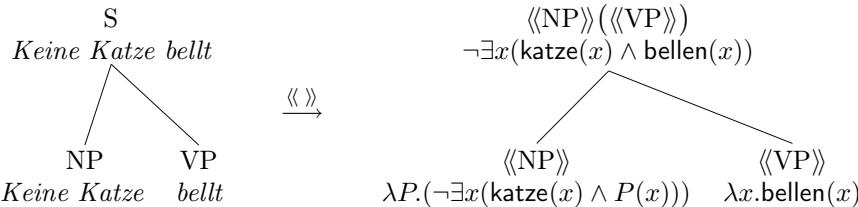
$$(3.80) Keine Katze bellt.$$

Intuitiv gesprochen besagt der Satz, dass es keine Katze gibt, die die Bellen-Eigenschaft hat. Was die syntaktische Struktur betrifft, ist er parallel zu Satz (3.64), wobei hier die NP *keine Katze* die Rolle des Subjekts übernimmt. Wollten wir wie oben vorgehen, müssten wir die VP-Bedeutung $\lambda x. bellen(x)$ auf deren Bedeutung anwenden. Da die VP-Bedeutung vom Typ $\langle e, t \rangle$ ist, müsste also *keine Katze* auf ein Individuum referieren. Aber welches Individuum sollte das sein? Die Aussage des Satzes ist ja gerade, dass es kein Individuum gibt, auf das die VP-Eigenschaft zutrifft.

Die Idee zur Lösung dieses scheinbaren Problems ist, die Berechnungsrichtung der Funktionalapplikation umzukehren und nicht die VP-Bedeutung auf die Subjektsbedeutung, sondern die Subjektsbedeutung auf die VP-Bedeutung anzuwenden: nicht *bellen* ist eine Eigenschaft des Subjekts, sondern *keine Katze* ist eine Eigenschaft von *bellen* – nämlich die, von keiner Katze getan zu werden. Damit muss $\langle\langle \text{keine Katze} \rangle\rangle$ vom Typ $\langle\langle e, t \rangle, t \rangle$ sein, so dass eine VP-Bedeutung vom Typ $\langle e, t \rangle$ als Argument genommen und eine Satzbedeutung vom Typ t als Ergebnis geliefert wird. Folgende Bedeutung ist von entsprechendem Typ:

$$(3.81) \langle\langle \text{keine Katze} \rangle\rangle = \lambda P. (\neg \exists x (\text{katze}(x) \wedge P(x)))$$

Die kompositionale Berechnung der Gesamtbedeutung ist hier wieder in Baumdarstellung gegeben. Man beachte die Umkehrung der Funktionalapplikation am Wurzelknoten des Baumes im Vergleich zu den vorangegangenen Analysen.



Die logische Formelrepräsentation der Gesamtbedeutung kann in etwa paraphrasiert werden mit *es ist nicht der Fall, dass es ein Individuum gibt, das die Katzen- und Bellen-Eigenschaft hat*, was eine adäquate Übersetzung für Satz (3.80) darstellt.

Was ist aber nun mit der Ableitung der Bedeutung von Satz (3.64)? Da $\langle\langle \text{Willi} \rangle\rangle$ vom Typ e ist, würde die Funktionalapplikation neuer Art hier nicht funktionieren und man müsste deshalb die Fälle, in denen die Subjekts-NP ein Eigename ist, von denen mit quantifizierender Subjekts-NP unterscheiden. Weiterhin ist es unschön, dass der syntaktischen Kategorie NP zwei semantische Typen entsprechen. Das alles lässt sich vermeiden, wenn man Eigennamen nicht mehr als Individuen vom Typ e betrachtet, sondern eine **Typanhebung** vornimmt, indem man statt des Individuums die Menge der Eigenschaften, die das Individuum hat, betrachtet. Mit anderen Worten wird der Eigename *Willi* nicht mehr als *willi*, sondern wie folgt übersetzt:

$$(3.82) \langle\langle \text{Willi} \rangle\rangle = \lambda P. P(\text{willi})$$

In der Herleitung der Bedeutung von Satz (3.64) kommt somit auch die neue Version der Funktionalapplikation zum Tragen:

$$\begin{aligned} \langle\langle \text{Willi schläft} \rangle\rangle &= \langle\langle \text{Willi} \rangle\rangle (\langle\langle \text{schläft} \rangle\rangle) \\ &= \lambda P. P(\text{willi})(\lambda x. \text{schlafen}(x)) \\ &= \text{schlafen}(\text{willi}) \end{aligned}$$

Damit ist auch die Bedeutung von Eigennamen zusammen mit der Bedeutung quantifizierender NPs vom Typ $\langle\langle e, t \rangle, t \rangle$. Diese eindeutige Korrespondenz von syntaktischer Kategorie und semantischem Typ ist eines der Postulate von Montague im Rahmen seiner Semantiktheorie. In den letzten Jahren hat sich allerdings eine etwas andere Sichtweise eingebürgert, nach der man für jeden Lexikoneintrag von einem möglichst niedrigen Typ ausgeht und diesen bei Bedarf anhebt. Wir werden weiter unten auf eine andere Möglichkeit zur Kombination von quantifikationalen NP-Bedeutungen kommen und dann die Umkehrung der Funktionalapplikation und die Typanhebung von eben rückgängig machen.

Was jetzt noch fehlt, ist, auch die NP-Bedeutung von *keine Katze* aus ihren Bestandteilen abzuleiten. Als syntaktische Struktur nehmen wir an, dass sich ein Determinierer mit einem Nomen zu einer NP verbindet, also $[\text{NP} [\text{Det } \text{keine}] [\text{N } \text{Katze}]]$. Da wir die gewünschte Gesamtbedeutung der NP kennen und zudem wissen, dass wir Nomen als Eigenschaften vom Typ $\langle e, t \rangle$ (s. den Fall der nominalen Prädikation in (3.66)) analysieren wollen, können wir uns die Bedeutung des Determinierers ausrechnen. Zunächst stellen wir fest, dass sich die NP-Bedeutung mittels Funktionalapplikation der Det-Bedeutung auf die N-Bedeutung ergeben muss. Damit muss $\langle\langle \text{keine} \rangle\rangle$ vom Typ $\langle\langle e, t \rangle, \langle\langle e, t \rangle, t \rangle \rangle$ sein und wir können folgende Gleichung auflösen:

$$\langle\!\langle \text{keine Katze} \rangle\!\rangle = \lambda P. (\neg \exists x (\text{katze}(x) \wedge P(x))) = \langle\!\langle \text{keine} \rangle\!\rangle (\lambda x. \text{katze}(x))$$

Als Ergebnis bekommt man für die Bedeutung des Determinierers Folgendes.

$$(3.83) \quad \langle\!\langle \text{keine} \rangle\!\rangle = \lambda Q \lambda P. (\neg \exists x(Q(x) \wedge P(x)))$$

Andere Determinierer wie *ein* und *jede* leisten entsprechende semantische Beiträge.

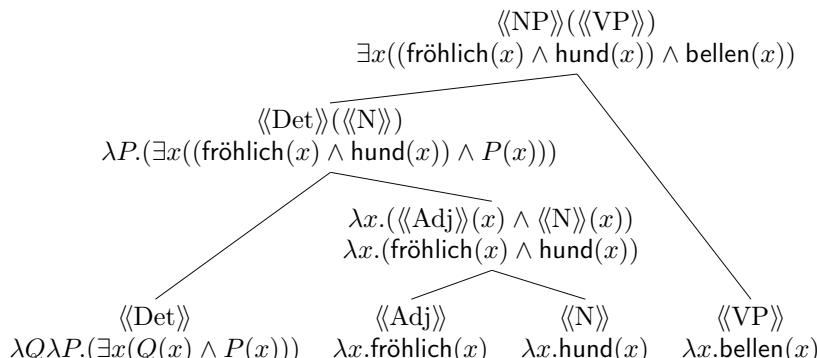
$$(3.84) \quad \langle\!\langle ein \rangle\!\rangle = \lambda Q \lambda P. (\exists x(Q(x) \wedge P(x)))$$

$$(3.85) \quad \langle\!\langle \text{jede} \rangle\!\rangle = \lambda Q \lambda P. (\forall x(Q(x) \rightarrow P(x)))$$

Ein Determinierer braucht also zwei Argumente. Das erste wird in den obigen Fällen vom Nomen geliefert und schränkt die Quantifikation in gewisser Weise ein: der quantifikationale Gehalt von *keine Katze* betrifft nur noch Katzen und keine anderen Individuen. Aus diesem Grund nennt man es **Restriktor**. Das zweite Argument (oben durch die VP-Bedeutung gegeben) gibt den Wirkungsbereich der Quantifikation an und wird (**Nuklear-)****Skopos** oder auch **Nukleus** genannt. Die Bedeutung für die NP *jede Frau* mit dem Restriktor $\lambda x.\text{frau}(x)$ ist beispielsweise wie folgt.

$$\langle\!\langle \text{jede Frau} \rangle\!\rangle = \lambda Q \lambda P. (\forall x(Q(x) \rightarrow P(x))) (\lambda x.\text{frau}(x)) = \lambda P. \forall x(\text{frau}(x) \rightarrow P(x))$$

Sie kann angesehen werden als die Menge der Eigenschaften für die gilt, dass, wenn ein Individuum eine Frau ist, es auch diese Eigenschaft hat. Zur weiteren Illustration der Komposition von Quantorenbedeutungen geben wir die Bedeutungsberechnung von (3.76) vollständig an.



Definite Kennzeichnungen

Wie steht es nun um die Bedeutung von definiten Kennzeichnungen wie der Subjekts-NP in folgendem berühmten Satz, den schon Bertrand Russell für seine Argumentationen benutzt hat?

(3.86) *Der gegenwärtige König von Frankreich ist kahlköpfig.*

Man könnte versucht sein, diese NP ähnlich wie einen Eigennamen durch ein Individuum zu deuten, doch welches Individuum sollte das sein, wenn wir von der gegenwärtigen (und zur Zeit Russells gültigen) Tatsache ausgehen, dass es keinen König von Frankreich gibt? In gewisser Weise scheint (3.86) also falsch zu sein. Merkwürdigerweise scheint aber die negierte Äußerung von (3.86) in derselben Weise falsch zu sein:

(3.87) *Der gegenwärtige König von Frankreich ist nicht kahlköpfig.*

Das ist insofern bemerkenswert, als hier das Prinzip des tertium non datur (s. Unterkapitel 2.1) verletzt scheint, welches gilt, wenn wir die definite Kennzeichnung durch einen Eigennamen ersetzen:

(3.88) *Willi ist kahlköpfig.*

(3.89) *Willi ist nicht kahlköpfig.*

Falls wir (3.88) als falsch bewerten, müssen wir zwangsläufig die Negation (3.89) als wahr betrachten, und umgekehrt. Das scheint bei (3.86) vs. (3.87), die beide gleichzeitig falsch erscheinen können, nicht so zu sein.

Russells Lösungsidee (die auch Montague übernommen hat) ist es, definiten Kennzeichnungen eine quantifikationale Analyse zuzuschreiben. Er schlägt vor, den definiten Artikel wie folgt zu analysieren.

(3.90) $\langle\langle \text{der} \rangle\rangle = \lambda Q \lambda P. \exists x(Q(x) \wedge \forall y(Q(y) \rightarrow x = y) \wedge P(x))$

Übersetzen wir das modifizierte Nomen *gegenwärtige König von Frankreich* als Eigenschaft $\lambda x. \text{KvF}(x)$ und das prädiktative Adjektiv als $\lambda x. \text{kahl}(x)$, so ergibt sich als Gesamtbedeutung für (3.86)

(3.91) $\exists x(\text{KvF}(x) \wedge \forall y(\text{KvF}(y) \rightarrow x = y) \wedge \text{kahl}(x))$

Diese Formel besteht aus drei Teilen, die die schon in Abschnitt 3.6.1 erwähnten Aspekte definiter Kennzeichnungen wiederspiegeln:

Existenz $\exists x(\text{KvF}(x))$ Es gibt einen König von Frankreich

Eindeutigkeit ... $\wedge \forall y(\text{KvF}(y) \rightarrow x = y)$... und zwar genau einen

Prädikation ... $\wedge \text{kahl}(x))$... und dieser ist kahlköpfig

Gibt es keinen König von Frankreich (formal genauer: erfüllt bei der modell-theoretischen Interpretation kein Individuum die KvF Eigenschaft), so ist (3.91) falsch. Russells Vorschlag erlaubt nun auch eine Ableitung in der die negierte Äußerung (3.87) unter diesen Umständen gleichsam falsch ist. Dazu muss die Negation **intern** interpretiert, d.h. nur auf die Prädikation bezogen, werden.

$$(3.92) \exists x(\text{KvF}(x) \wedge \forall y(\text{KvF}(y) \rightarrow x = y) \wedge \neg\text{wahl}(x))$$

Damit ist auch diese Formel, wie gewünscht, falsch, falls kein König von Frankreich existiert.

Russells Analyse wurde von Peter Strawson dahingehend kritisiert, dass bei Nichtexistenz eines entsprechenden Individuums Sätze mit definiten Kennzeichnungen nicht falsch, sondern unangemessen sind. Er schlägt vor, die beobachteten Existenz- und Einzigkeitsbedingungen als Präsuppositionen zu verstehen, also als Voraussetzungen, die erfüllt sein müssen, damit ein Satz überhaupt einen Wahrheitswert hat (s. auch Unterkapitel 3.7.3). Damit wären die Sätze (3.86) und (3.87) weder wahr noch falsch, sondern ohne Wahrheitswert. Eine genauere Betrachtung der Strawsonschen Idee muss an dieser Stelle leider ausbleiben.

Skopusambiguität

Bislang hatten wir quantifizierende NPn nur in Subjektposition angetroffen, aber natürlich wollen wir auch imstande sein, solche NPn als Objekte zu analysieren. Betrachten wir hier zunächst den Fall von Kopulakonstruktionen mit quantifizierenden NPn.

$$(3.93) Fido \text{ ist ein fröhlicher Hund.}$$

Im Fall von nominaler Prädikation (3.66) hatten wir für die Analyse der Kopula die „semantisch leere“ Bedeutung $\langle\langle ist \rangle\rangle = \lambda P \lambda x. P(x)$ vom Typ $\langle\langle e, t \rangle, \langle e, t \rangle\rangle$ angenommen. Für den obigen komplexeren Fall wird dies nicht genügen, denn das Argument der Kopula ist nun eine quantifizierende NP vom Typ $\langle\langle e, t \rangle, t \rangle$. Wir benötigen also zusätzlich eine weitere, entsprechend angepasste Repräsentation. Die folgende vom Typ $\langle\langle\langle e, t \rangle, t \rangle, \langle e, t \rangle\rangle$ leistet das Gewünschte.

$$(3.94) \langle\langle ist_{NP} \rangle\rangle = \lambda G \lambda x. G(\lambda y. x = y)$$

Die Berechnung der Gesamtbedeutung sieht damit wie folgt aus, wobei wir die Analyse der NP nicht noch einmal angeben, sondern von oben übernehmen. Man beachte auch, dass hier die mehrfachen Vorkommen der Variable x teilweise eine Umbenennung notwendig machen (s. Unterkapitel 2.1).

$$\begin{aligned} \langle\langle Fido \text{ ist ein fröhlicher Hund} \rangle\rangle &= \langle\langle Fido \rangle\rangle \left(\langle\langle ist \rangle\rangle (\langle\langle \text{ein fröhlicher Hund} \rangle\rangle) \right) \\ &= \langle\langle Fido \rangle\rangle \left(\lambda G \lambda x. G(\lambda y. x = y) (\lambda P. \exists x ((\text{fröhlich}(x) \wedge \text{hund}(x)) \wedge P(x))) \right) \\ &= \lambda P. P(\text{fido}) \left(\lambda z. \exists x ((\text{fröhlich}(x) \wedge \text{hund}(x)) \wedge z = x) \right) \\ &= \exists x ((\text{fröhlich}(x) \wedge \text{hund}(x)) \wedge \text{fido} = x) \end{aligned}$$

Ein weiterer Vorteil der Russellschen Analyse von definiten Kennzeichnungen ist auch, dass man diese wie andere quantifizierende NPn behandeln kann. Das trifft auch für Kopulakonstruktionen wie die Folgende zu.

$$(3.95) Willi \text{ ist der König von Frankreich.}$$

$$\exists x(\text{KvF}(x) \wedge \forall y(\text{KvF}(y) \rightarrow x = y) \wedge \text{willi} = x)$$

Wir erweitern nun unsere Betrachtungen von Kopulakonstruktionen auf Konstruktionen mit transitiven Verben und quantifizierenden NPn in Objektposition.

(3.96) *Willi sieht einen Hund.*

Ginge man von einer grundlegenden syntaktischen Struktur wie in (3.70) aus, so wäre die Kombination von Verbbedeutung und Objektbedeutung nicht direkt möglich, da ersteres vom Typ $\langle e, \langle e, t \rangle \rangle$ und letzteres vom Typ $\langle \langle e, t \rangle, t \rangle$ ist. Außerdem ergibt sich ein Problem, das wir schon in Abschnitt 3.6.1 angesprochen haben, wenn zusätzlich noch das Subjekt quantifizierend ist.

(3.97) *Jede Frau sieht einen Hund.*

Dieser Satz kann intuitiv auf zwei Arten verstanden werden, d.h. er ist **mehrdeutig** (oder auch: **ambig**) und hat zwei **Lesarten**.

1. Es gibt einen bestimmten Hund, z.B. den fröhlichen Hund Fido, für den gilt, dass jede Frau ihn sieht.
2. Für jede Frau gibt es einen (möglicherweise verschiedenen) Hund, den sie sieht.

Das Problem ist nun, dass man aus *einer* zugrundeliegenden syntaktischen Struktur nur *eine* logische Repräsentation kompositionally ableiten kann. Wie könnte man dann aber der Tatsache Rechnung tragen, dass Satz (3.97) zwei Lesarten hat?

Die Lösung muss darin bestehen, für jede Lesart eine zugrundeliegende Struktur zur Verfügung zu haben. Wie genau diese Strukturen zustande kommen, ist abhängig von den Annahmen zur Grammatik und wir werden deshalb nur die Strukturen selbst und ihre Interpretation, nicht jedoch ihre detaillierte Herleitung besprechen. Dabei orientieren wir uns wieder an Montagues System. Die Grundidee hier ist, den Satz zunächst mit Platzhaltern anstelle von quantifizierenden NPn abzuleiten. Für Satz (3.96) würde beispielsweise zunächst Folgendes abgeleitet:

(3.98) *Willi sieht v_0 .*

Hier ist v_0 ein Platzhalter, der anstelle einer beliebigen NP stehen kann. Die Bedeutung eines solchen Platzhalters ist einfach eine entsprechende Individuenvariable vom Typ e : $\langle\langle v_i \rangle\rangle = x_i$. Damit ist die Bedeutung von passendem Typ für die Kombination mit dem transitiven Verb und die Gesamtbedeutung des Satzes (3.98) lässt sich ganz einfach berechnen.

$$(3.99) \langle\langle \text{Willi sieht } v_0 \rangle\rangle = \lambda P.P(\text{willi}) \left(\lambda y \lambda x. \text{sieht}(y)(x)(x_0) \right) = \text{sehen}(x_0)(\text{willi})$$

In dieser Repräsentation ist die Variable x_0 frei und eine Paraphrase könnte lauten, dass Willi das Individuum, für das x_0 steht, sieht. Der Trick ist nun, dafür zu sorgen, dass x_0 mit der Bedeutung der quantifizierenden NP in Beziehung

gesetzt wird. Dies geschieht über eine neue Kombinationsregel (die neben der Funktionalapplikation und der Modifikationsregel steht), welche aus der Repräsentation des Satzes mit einer freien Variablen eine Eigenschaft macht, die dann als Skopussargument für die quantifikationale Bedeutung der NP dienen kann:

$$\langle\langle \text{NP} \rangle\rangle (\lambda x_i. \langle\langle \text{S} \rangle\rangle)$$

Dabei ist es natürlich wichtig, über genau die Variable x_i zu abstrahieren, für die der entsprechende Platzhalter v_i (mit gleichem Index) in S zu finden ist. In Montagues System wird dies über eine Indizierung der Regeln erreicht. Auf syntaktischer Seite wird dabei der Platzhalter v_i durch den NP-Ausdruck ersetzt, so dass man am Ende einen grammatischen Satz ohne Platzhalter erhält. Auf diese Details gehen wir im Folgenden nicht weiter ein.

Insgesamt ergibt sich für den gerade besprochenen Fall folgende Repräsentation, die die Bedeutung von (3.96) korrekt charakterisiert.

$$\begin{aligned} (3.100) \quad & \langle\langle \text{einen Hund} \rangle\rangle (\lambda x_0. \langle\langle \text{Willi sieht } v_0 \rangle\rangle) \\ & = \lambda P. \exists x (\text{hund}(x) \wedge P(x)) (\lambda x_0. \text{sehen}(x_0)(\text{willi})) \\ & = \exists x (\text{hund}(x) \wedge \text{sehen}(x)(\text{willi})) \end{aligned}$$

Dieses Verfahren, die Satzbedeutung zunächst mit Platzhalter abzuleiten und dann einen Quantor auf eine entsprechend gebildete Eigenschaft anzuwenden, nennt Montague **Quantifying In** (dt. hineinquantifizieren), denn der Quantor bindet eine Variable im Satz und quantifiziert in gewisser Weise dort „hinein“.

In dem ambigen Fall von (3.97) mit zwei quantifizierenden NPs verfahren wir ähnlich und leiten entsprechend zunächst eine Repräsentation mit Platzhalter an der Objektstelle ab. Anschließend wenden wir den Objektquantor auf die abgeleitete Eigenschaft an, um zum endgültigen Resultat zu kommen.

$$\begin{aligned} (3.101) \quad & \langle\langle \text{einen Hund} \rangle\rangle (\lambda x_0. \langle\langle \text{jede Frau sieht } v_0 \rangle\rangle) \\ & = \lambda P. \exists x (\text{hund}(x) \wedge P(x)) (\lambda x_0. \forall x (\text{frau}(x) \rightarrow \text{sehen}(x_0)(x))) \\ & = \exists y (\text{hund}(y) \wedge \forall x (\text{frau}(x) \rightarrow \text{sehen}(y)(x))) \end{aligned}$$

Damit haben wir die erste der oben angegebenen Lesarten abgeleitet. Die Formel ist wahr, wenn es (im Modell) ein Individuum gibt, das ein Hund ist, sodass für alle Frauen gilt, dass sie diesen Hund sehen. Wie man an der Ableitung erkennen kann, erscheint die Bedeutung von *jeder Frau* im Skopuss von *einen Hund*. Man sagt entsprechend, dass der zum Objekt gehörige Quantor (bzw. etwas ungenauer, dass die entsprechende NP) **Skopuss über** den zum Subjekt gehörigen Quantor (bzw. NP) **hat/nimmt**.

Wie leitet man aber nun die zweite Lesart ab? Im Prinzip durch den gleichen Mechanismus, nur diesmal auf beide NPs angewandt. Man leitet also zunächst folgendes ab:

$$(3.102) \quad v_1 \text{ sieht } v_0.$$

Bei der Semantikkomposition gibt es an dieser Stelle in unserem aktuellen Ansatz allerdings wieder ein Problem, denn das Subjekt wird als Individuenvariable x_1 übersetzt und ist vom Typ e . Wir hatten aber oben nach (3.80) zur Ableitung von einfachen Sätzen mit quantifikationalen Subjekten vorgeschlagen, bei der Berechnung einer Satzbedeutung die Subjektbedeutung auf die VP-Bedeutung anzuwenden und damit die ursprüngliche Berechnungsrichtung umzukehren. Das würde aufgrund des Typ e -Subjekts hier nicht funktionieren. Deshalb machen wir diese Maßnahme einfach wieder rückgängig! Die VP-Bedeutung wird wie ursprünglich auf die Objekt- und Subjektbedeutung angewandt. Damit können wir als Typ für Eigennamen auch wieder den niedrigen Typ e annehmen und somit die Typanhebung von oben ignorieren (d.h. «Willi» = willi ist wieder vom Typ e). Wir bekommen also zunächst

$$(3.103) \quad \langle\!\langle v_1 \text{ sieht } v_0 \rangle\!\rangle = \langle\!\langle \text{sieht} \rangle\!\rangle(x_0)(x_1) = \text{sehen}(x_0)(x_1)$$

An dieser Stelle können nun beide Quantoren hineinquantifiziert werden. Entscheidend ist hierbei die Reihenfolge. Beginnen wir zunächst mit dem Objektquantor «einen Hund».

$$\begin{aligned} (3.104) \quad & \langle\!\langle \text{einen Hund} \rangle\!\rangle(\lambda x_0. \langle\!\langle v_1 \text{ sieht } v_0 \rangle\!\rangle) \\ &= \lambda P. \exists x (\text{hund}(x) \wedge P(x))(\lambda x_0. \text{sehen}(x_0)(x_1)) \\ &= \exists x (\text{hund}(x) \wedge \text{sehen}(x)(x_1)) \end{aligned}$$

An dieser Stelle kann jetzt der Subjektquantor «jede Frau» per Quantifying-In seinen Beitrag leisten:

$$\begin{aligned} (3.105) \quad & \langle\!\langle \text{jede Frau} \rangle\!\rangle(\lambda x_1. \exists x (\text{hund}(x) \wedge \text{sehen}(x)(x_1))) \\ &= \lambda P. \forall y (\text{frau}(y) \rightarrow P(y))(\lambda x_1. \exists x (\text{hund}(x) \wedge \text{sehen}(x)(x_1))) \\ &= \forall y (\text{frau}(y) \rightarrow \exists x (\text{hund}(x) \wedge \text{sehen}(x)(y))) \end{aligned}$$

Dies ist eine Repräsentation der zweiten Lesart von (3.97). Für jedes Individuum gilt, dass, wenn es eine Frau ist, ein Hund existiert, den es sieht. Umgekehrt zur ersten Lesart erscheint hier die Bedeutung von *einen Hund* im Skopus der Bedeutung von *jede Frau*. Da die Skopusverhältnisse in diesem Fall entscheidend für die Mehrdeutigkeit sind, spricht man auch von einer **Skopusbewegung**.

Wenn wir die Reihenfolge beim Quantifying In umdrehen, also zuerst den Subjektquantor und dann den Objektquantor entsprechend mit (3.103) verknüpfen, erhalten wir wieder eine Repräsentation der Bedeutung der ersten Lesart, genau genommen exakt dieselbe wie in (3.101). Wir erinnern allerdings daran, dass die Ableitung in (3.101) nun nicht mehr zur Verfügung steht, da wir ja die Funktionalapplikation, die Subjekt- und VP-Bedeutung kombiniert, wieder wie ursprünglich (also VP-Bedeutung angewendet auf Subjekt-Bedeutung) annehmen.

Wir fassen die Ergebnisse dieses Abschnitts zusammen. Transitiv Verben sind vom Typ $\langle e, \langle e, t \rangle \rangle$. Auf semantischer Seite kombinieren sich die Bedeutungen von Verb und Objekt zu einer VP-Bedeutung und von VP und Subjekt zu einer

Satzbedeutung. Sind Subjekt oder Objekt Eigennamen vom Typ e , können sie also vom Verb bzw. der VP als Argument genommen werden. Da quantifizierende NPn von höherem Typ sind, müssen zunächst Platzhalter v_i deren Rolle bei der Berechnung einer vorläufigen Satzbedeutung mit freien Variablen übernehmen. Die endgültige Satzbedeutung erhält man dann durch Hineinquantifizieren der NP-Bedeutungen in die vorläufige Satzbedeutung, wobei die Reihenfolge des Hineinquantifizierens die abgeleitete Lesart bestimmt.

Bei Ansätzen der generativen Grammatik in der Tradition von Noam Chomsky werden im Prinzip ähnliche Strukturen abgeleitet, allerdings auf andere Art. Hier werden die quantifizierenden NPn syntaktisch zunächst tatsächlich an der Stelle generiert, an der sie an der Oberfläche stehen. Um zu einer Interpretation zu gelangen, müssen sie allerdings von dort wegbewegt werden, wobei sie an ihrer ursprünglichen Position eine Spur hinterlassen. Diese Spuren kann man in etwa mit den Platzhaltern bei Montague vergleichen, denn sie werden ähnlich wie diese interpretiert. Den Prozess der Quantorenbewegung nennt man **Quantifier Raising**.

Theorie der generalisierten Quantoren

Für die bisher besprochenen Determinierer *ein*, *jeder*, *kein* und den definiten Artikel konnten wir in (3.83)–(3.85) und (3.90) Bedeutungsbeiträge angeben, die am Ende zu Repräsentationen führten, die alle prädikatenlogisch interpretierbar waren. Obwohl wir also bei der Kombination der Bestandteile Gebrauch von dem mächtigeren System der Lambda-Typenlogik gemacht haben, scheinen sich die Endergebnisse alle im einfacheren System der Prädikatenlogik ausdrücken zu lassen. Ist die Prädikatenlogik erster Stufe vielleicht generell mächtig genug, um die Bedeutung natürlichsprachlicher Sätze zu erfassen?

Die Antwort ist leider negativ, denn es gibt quantifikationale NPn, deren Bedeutungsbeitrag sich nicht prädikatenlogisch fassen lässt. Ein Beispiel hierfür sind solche NPn, die mit dem Determinierer *die meisten* gebildet werden.

(3.106) *Die meisten Kinder lachen.*

Intuitiv gesprochen erachten wir diesen Satz als wahr, wenn es mehr lachende Kinder als nicht-lachende Kinder gibt, in anderen Worten, wenn mehr als die Hälfte der Kinder lacht. Der Bedeutungsbeitrag des Determinierers kann also in etwa mit *mehr als die Hälfte* umschrieben werden, was mit den in der Prädikatenlogik zur Verfügung stehenden Quantoren nicht ausgedrückt werden kann.

Ähnlich komplex sind die Bedeutungen der beiden Quantoren *viele* und *wenige*, wie folgender Satz verdeutlicht.

(3.107) *Viele Gäste waren heute mit dem Essen nicht zufrieden.*

Viele Gäste ist hier relativ zum Normalfall zu interpretieren. Nehmen wir an, es handelte sich um insgesamt 50 Gäste. In einem Nobelrestaurant würden wahrscheinlich schon 4 unzufriedene Gäste (also in diesem Fall 8%) als „viele“ angesehen werden. In einer Mensa wären vielleicht 20 unzufriedene Gäste (also 40%) als

„viele“ anzusehen. Bei der Interpretation dieser beiden Quantoren muss deshalb weiteres Wissen einfließen, was einer zusätzlichen Spezifizierung bedarf.

Man verallgemeinert deshalb die im vorigen Abschnitt vorgestellte Sichtweise auf Quantifikation und betrachtet Determinierer als Prädikate, die etwas über das Verhältnis von Restriktor und Skopus aussagen. Die Restriktor- und Skopuseigenschaften werden dabei der Einfachheit halber als Mengen aufgefasst (s. Unterkapitel 2.1 zur Austauschbarkeit dieser beiden Sichtweisen). In dieser Sichtweise macht der Satz *Jeder Hund bellt* eine Aussage über die Menge der Hunde und die Menge der bellenden Individuen, indem er behauptet, dass erstere eine Teilmenge der letzteren ist. *Jeder* setzt also seinen Restriktor $\lambda x.\text{hund}(x)$ und seinen Skopus $\lambda x.\text{bellen}(x)$ über die Teilmengenrelation miteinander in Beziehung. Als semantischen Eintrag könnten wir also folgenden angeben.

$$\langle\langle \text{jeder} \rangle\rangle = \lambda P \lambda Q. P \subseteq Q$$

Dieser Bedeutungsbeitrag ist in der Sprache der Mengenlehre gegeben und äquivalent zu (3.85). Der Satz *Jeder Hund bellt* bekommt damit folgende Bedeutungsrepräsentation

$$\langle\langle \text{Jeder Hund bellt} \rangle\rangle = \text{hund} \subseteq \text{bellen}$$

Sie ist genau dann wahr, wenn die Menge der Hunde im Modell eine Teilmenge der Individuen ist, die die Bellen-Eigenschaft haben, in anderen Worten: wenn im Modell alle Hunde bellen.

Entsprechend lässt sich der Determinierer *die meisten* als

$$\langle\langle \text{die meisten} \rangle\rangle = \lambda P \lambda Q. |P \cap Q| > |P \setminus Q|$$

definieren. Damit erhält (3.106) die Bedeutungsrepräsentation

$$\langle\langle \text{Die meisten Kinder lachen} \rangle\rangle = |\text{kind} \cap \text{lachen}| > |\text{kind} \setminus \text{lachen}|$$

die wie gewünscht wahr ist, wenn die Menge der lachenden Kinder ($\text{kind} \cap \text{lachen}$) größer ist als die Menge der nicht-lachenden Kinder ($\text{kind} \setminus \text{lachen}$). Tabelle 3.6 gibt die Semantik einiger Quantoren an, wobei P den Restriktor und Q den Skopus bezeichnet. *NUM* steht in der Tabelle für einen beliebigen Numeralausdruck wie *zwei*, *drei*, *vier*, etc und n für die ihm entsprechende natürliche Zahl. Bei den Quantoren zu *viele* und *wenige* gibt ein Kontextparameter c die Prozanzahl an, ab der die „viele“-Grenze erreicht ist.

Zwei wichtige Eigenschaften dieser Relationen auf Individuenmengen sollen im Folgenden kurz angesprochen werden.

Konservativität: Eine Relation R zwischen Mengen (und damit eine Determiniererbedeutung) nennt man **konservativ** wenn für alle Mengen P und Q gilt:

$$R(P, Q) \leftrightarrow R(P, P \cap Q).$$

Um Aussagen $R(P, Q)$ mit konservativen Relationen R zu verifizieren oder zu falsifizieren, muss man also nur die Individuen der Restriktormenge P in Betracht

Determinierer	Wahrheitsbedingung
jeder	$P \subseteq Q$
ein	$P \cap Q \neq \emptyset$
kein	$P \cap Q = \emptyset$
NUM	$ P \cap Q \geq n$
genau NUM	$ P \cap Q = n$
mindestens NUM	$ P \cap Q \geq n$
höchstens NUM	$ P \cap Q \leq n$
mehr als die Hälfte	$ P \cap Q > 0.5 * P $
die meisten	$ P \cap Q > P \setminus Q $
viele	$ P \cap Q > c * P $, c kontextuell
wenige	$ P \cap Q < c * P $, c kontextuell
der	$ P = 1$ und $P \cap Q \neq \emptyset$

Tabelle 3.6: Semantik einiger Quantoren

ziehen. Die Individuen in $Q \setminus P$ und außerhalb von P und Q sind irrelevant. Beispielsweise gilt für *jeder*, dass aus $P \subseteq Q$ folgt, dass $P \subseteq (P \cap Q)$ – und umgekehrt. Um die Konservativität auf sprachlicher Seite nachzuvollziehen, betrachtet man Satzpaare wie Folgendes.

(3.108) *Jede Frau schläft.* gdw. *Jede Frau ist eine Frau, die schläft.*

Gilt die *genau dann, wenn*-Beziehung zwischen beiden Sätzen (wie hier), so spricht das für die Konservativität der Determiniererbedeutung.

Monotonie: Die Eigenschaft der Monotonie betrifft das Inferenzpotenzial eines Quantorenausdrucks hinsichtlich einer Vergrößerung bzw. Verkleinerung seiner Restriktor- bzw. Skopusmenge. Sind Q, P Mengen und $P' \supseteq P, Q' \supseteq Q$ Obermengen von P bzw. Q , so nennt man eine Relation R zwischen Mengen (und damit eine Determiniererbedeutung)

- | | |
|-------------------------|---------------------------------------|
| links monoton steigend | wenn $R(P, Q) \rightarrow R(P', Q)$, |
| links monoton fallend | wenn $R(P', Q) \rightarrow R(P, Q)$, |
| rechts monoton steigend | wenn $R(P, Q) \rightarrow R(P, Q')$, |
| rechts monoton fallend | wenn $R(P, Q') \rightarrow R(P, Q)$. |

Die Bedeutung von *jeder* ist beispielsweise links monoton fallend und rechts monoton steigend, denn wenn $\langle\langle \text{jeder} \rangle\rangle(P')(Q) = P' \subseteq Q$ gilt, dann gilt auch $\langle\langle \text{jeder} \rangle\rangle(P)(Q)$ und $\langle\langle \text{jeder} \rangle\rangle(P)(Q')$. Gilt also $P' \subseteq Q$, kann man den Restriktor beliebig verkleinern bzw. den Skopus beliebig erweitern. Das spiegelt sich auch in folgenden korrekten natürlichsprachlichen Schlussfolgerungen wieder.

Aus der Kombination von Determiniererbedeutung und Restriktorbedeutung ergibt sich im Allgemeinen ein logischer Ausdruck vom Typ $\langle\langle e, t \rangle, t \rangle$. Ein solches Objekt entspricht einer Menge von Mengen und man nennt es **generalisierten Quantor**. Beispielsweise ist die Bedeutung der NP *vier Hunde* gerade die Menge all der Mengen, die mit der Menge der Hunde vier oder mehr Elemente gemeinsam haben.

$$\begin{aligned}\langle\langle \text{vier Hunde} \rangle\rangle &= \langle\langle \text{vier} \rangle\rangle (\langle\langle \text{Hunde} \rangle\rangle) = \lambda P \lambda Q. (|P \cap Q| \geq 4) (\text{hund}) \\ &= \lambda Q. |\text{hund} \cap Q| \geq 4 = \{Q \mid |\text{hund} \cap Q| \geq 4\}\end{aligned}$$

Ein Fragment des Deutschen

Montague analysiert in seinen Schriften unterschiedliche Fragmente des Englischen mit semantisch interessanten Phänomenen. Diese Herangehensweise war insofern ein Fortschritt, als er zum ersten Mal eine formal explizite Behandlung einer unendlichen Teilmenge der natürlichen Sprache geben konnte. Wir geben im Folgenden die im vorigen Abschnitt diskutierten Regeln und lexikalischen Bedeutungen nochmals zusammengefasst für eine Analyse eines kleinen Fragments des Deutschen an. Die Syntaxregeln in Abbildung 3.45 sind dabei als einfache, kontextfreie Grammatik angegeben (vgl. Unterkapitel 2.2 und 3.5), bei der wir, wie vorher, Kongruenz außer Acht lassen. Regel R_5 ist für die Analyse von Kopulakonstruktionen mit prädikativen Adjektiven und Nomen zuständig, wobei hier zwei kontextfreie Phrasenstrukturregeln zusammengefasst sind. Regel R_6^i ist die syntaktische Regel, die beim Quantifying In einen grammatischen Satz durch Ersetzung des Platzhalters erzeugt. Diese Ersetzung des Platzhalters ist eine Operation, die über die einfache Verkettung einer kontextfreien Grammatik hinausgeht. Außerdem ist die Regel mit dem Index des Platzhalters indiziert. Streng genommen haben wir es also mit mehreren Regeln zu tun.

$$\begin{array}{ll} R_1: S \rightarrow NP VP & R_4: VP \rightarrow V NP \\ R_2: NP \rightarrow Det N & R_5: VP \rightarrow ist (Adj \mid N) \\ R_3: N \rightarrow Adj N & R_6^i: S^i \rightarrow NP S \\ & \text{wobei } v_i \text{ in } S \text{ durch } NP \text{ ersetzt wird} \end{array}$$

Abbildung 3.45: Syntaxregeln

Die Lexikoneinträge in Tabelle 3.7 ordnen den Worten entsprechende syntaktische Kategorien zu. Im semantischen Lexikon werden einige Wörter auf logische Ausdrücke abgebildet (siehe Tabelle 3.8). Die Determinierer werden dabei wie in Tabelle 3.6 angegeben übersetzt. Es mag dabei auffallen, dass wir uns nicht an die von Montague postulierte Kategorie-Typ-Korrespondenz halten. Z. B. werden *Willi* und *jede Frau* als NP kategorisiert, aber die entsprechenden Bedeutungen sind vom Typ e bzw. $\langle\langle e, t \rangle, t \rangle$. Weiterhin gibt es keinen Lexikoneintrag für die Kopula in Verbindung mit prädikativen Adjektiven oder Nomen durch Regel R_5 , wie wir sie in (3.67) angegeben hatten. Wir werden unten sehen, wie die korrekte Bedeutung in diesen Fällen trotzdem abgeleitet werden kann.

Kategorie	Wörter	Bemerkung
NP	{Willi, Maria, Robert}	Eigenamen
N	{Mann, Frau, Student, Buch, Fisch}	Nomen
Det	{jeder, die, kein, eine, drei, ...}	Determinierer
Adj	{fröhlich, verheiratet, ...}	Adjektive
VP	{schläft, rennt, spricht}	intransitive Verben
V	{ist, sieht, kennt, findet, sucht}	transitive Verben

Tabelle 3.7: Syntaktisches Lexikon

Übersetzung		Bemerkung
«Willi»	= willi	genauso Maria, Robert
«schläft»	= $\lambda x.\text{schlafen}(x)$	genauso rennt, spricht
«fröhlich»	= $\lambda x.\text{fröhlich}(x)$	genauso verheiratet
«sieht»	= $\lambda y\lambda x.\text{sehen}(y)(x)$	genauso kennt, etc.
«ist»	= $\lambda G\lambda x.G(\lambda y.x = y)$	
« v_i »	= x_i	

Tabelle 3.8: Semantisches Lexikon

Zu jeder syntaktischen Regel muss jetzt noch eine semantische Regel angegeben werden, die bestimmt, wie die Bedeutung des komplexeren Ausdrucks aus den Bedeutungen seiner Teile errechnet wird. Wie oben gesehen, gehört z. B. zur syntaktischen Regel $S \rightarrow NP VP$ die semantische Regel $\langle\langle S \rangle\rangle = \langle\langle VP \rangle\rangle(\langle\langle NP \rangle\rangle)$, die besagt, dass man das Resultat $\langle\langle S \rangle\rangle$ für S durch Funktionalapplikation des Resultats $\langle\langle VP \rangle\rangle$ für NP auf das Resultat $\langle\langle NP \rangle\rangle$ für VP erhält. Die semantischen Regeln sind in Tabelle 3.9 zu finden. Diese Grammatik erlaubt es uns,

syntaktische Regel	semantische Regel
$R_1: S \rightarrow NP V$	$\langle\langle S \rangle\rangle = \langle\langle VP \rangle\rangle(\langle\langle NP \rangle\rangle)$
$R_2: NP \rightarrow Det N$	$\langle\langle NP \rangle\rangle = \langle\langle Det \rangle\rangle(\langle\langle N \rangle\rangle)$
$R_3: N \rightarrow Adj N$	$\langle\langle N \rangle\rangle = \lambda x.(\langle\langle Adj \rangle\rangle(x) \wedge \langle\langle N \rangle\rangle(x))$
$R_4: VP \rightarrow V NP$	$\langle\langle VP \rangle\rangle = \langle\langle V \rangle\rangle(\langle\langle NP \rangle\rangle)$
$R_5: VP \rightarrow ist (Adj N)$	$\langle\langle VP \rangle\rangle = \langle\langle Adj \rangle\rangle$ bzw. $\langle\langle N \rangle\rangle$
$R_6^i: S^i \rightarrow NP S$	$\langle\langle S^i \rangle\rangle = \langle\langle NP \rangle\rangle(\lambda x_i. \langle\langle S \rangle\rangle)$

Tabelle 3.9: Semantische Regeln

die im vorangegangenen Abschnitt vorgestellten Phänomene zu behandeln. Besondere Beachtung verdient Regel R_5 . Durch die syntaktische Regel wird das Wort *ist* in die abgeleitete Zeichenkette eingebracht. Damit ist dieses Wort im formalen Sinne kein „Teil“ komplexerer Ausdrücke mehr – beispielsweise hat die

VP *ist fröhlich* als Bestandteil nur das Adjektiv *fröhlich*, auf das die Regel R_5 angewendet wurde). Deshalb muss die Kopula *ist* auch keine Bedeutung wie in (3.67) mehr erhalten und die Bedeutung der VP kann einfach mit der Bedeutung des prädikativen Komplements identifiziert werden. Eine solche Behandlung direkt über die syntaktische Regel bietet sich insbesondere für semantisch „leere“ Ausdrücke an. Solche Ausdrücke, die nur im Zusammenhang mit anderen eine Bedeutung erlangen, nennt man **synkategorematisch**. Natürlich ist *ist* nur in unserem einfachen Fragment semantisch leer, in einer vollständigeren Behandlung würde *ist* zumindest temporale Information beitragen.

Des Weiteren fällt auf, dass wir laut syntaktischer Regeln zwar die Struktur $[S[NP[Det Jeder][N Hund]][VP schläft]]$ herleiten, aber nicht interpretieren können, da die VP vom Typ $\langle e, t \rangle$ laut semantischer Regel von R_1 auf die Subjektsbedeutung angewendet werden muss, was aufgrund des Subjekttyps $\langle \langle e, t \rangle, t \rangle$ unmöglich ist. Es bleibt also nur, den Satz *Jeder Hund schläft* mittels syntaktischer Variablen und der Regel R_6 abzuleiten und zu interpretieren. Ein derartiger Ausschluss von Ableitungen aufgrund semantischer Typunverträglichkeiten war bei Montague nicht vorgesehen, ist aber in der heutigen Semantikauffassung weit verbreitet. Man spricht allgemein bei semantischen Systemen, bei denen die Bedeutungskombinatorik durch die Typen bestimmt wird, von **typgetriebener Interpretation**.

Am Ende wollen wir nochmals die Ableitung der beiden Lesarten von (3.97) in Baumdarstellung in Abbildung 3.46 geben, um den Prozess des Quantifying In zu verdeutlichen. Da der Syntaxbaum und der semantische Ableitungsbaum aufgrund der Kompositionnalität genau dieselbe Struktur haben, haben wir zur Platzersparnis beide Aspekte in einem Baum vereint und durch einen Punkt • voneinander getrennt. Da in diesem Beispiel nur die Determinierer *jeder* und *ein* vorkommen, deren Bedeutungen sich problemlos in der Prädikatenlogik ausdrücken lassen, haben wir diese Darstellung der (äquivalenten) mengentheoretischen Darstellung aus Tabelle 3.6, wie in der Literatur üblich, vorgezogen.

Bewertung der Montague-Semantik als Computersemantik

Wesentliche Beiträge der Montague-Semantik zur Computersemantik sind zum einen eine ganze Reihe von Grundsatzentscheidungen; z.B. ist Kompositionalität einer der tragenden Bestandteile der Montague-Semantik und auch in der Computersemantik unverzichtbar. Der andere große Beitrag der Montague-Semantik ist der Lambda-Kalkül, der hier zum ersten Mal zum Zwecke der Semantikkonstruktion eingesetzt wurde. Auch wenn die Montague-Semantik eine reine Satzsemantik ist, ist der Semantik-Aufbau nach den Prinzipien der Montague-Semantik immer noch Stand der Technik. Als Einführung in die Problematik sei das klassische Werk (Pereira und Shieber 1987) genannt.

3.6.3 Diskursrepräsentationstheorie

Die Diskursrepräsentationstheorie (DRT) ist eine Theorie der Semantik natürlicher Sprache, die Anfang der achtziger Jahre von Hans Kamp entwickelt wurde

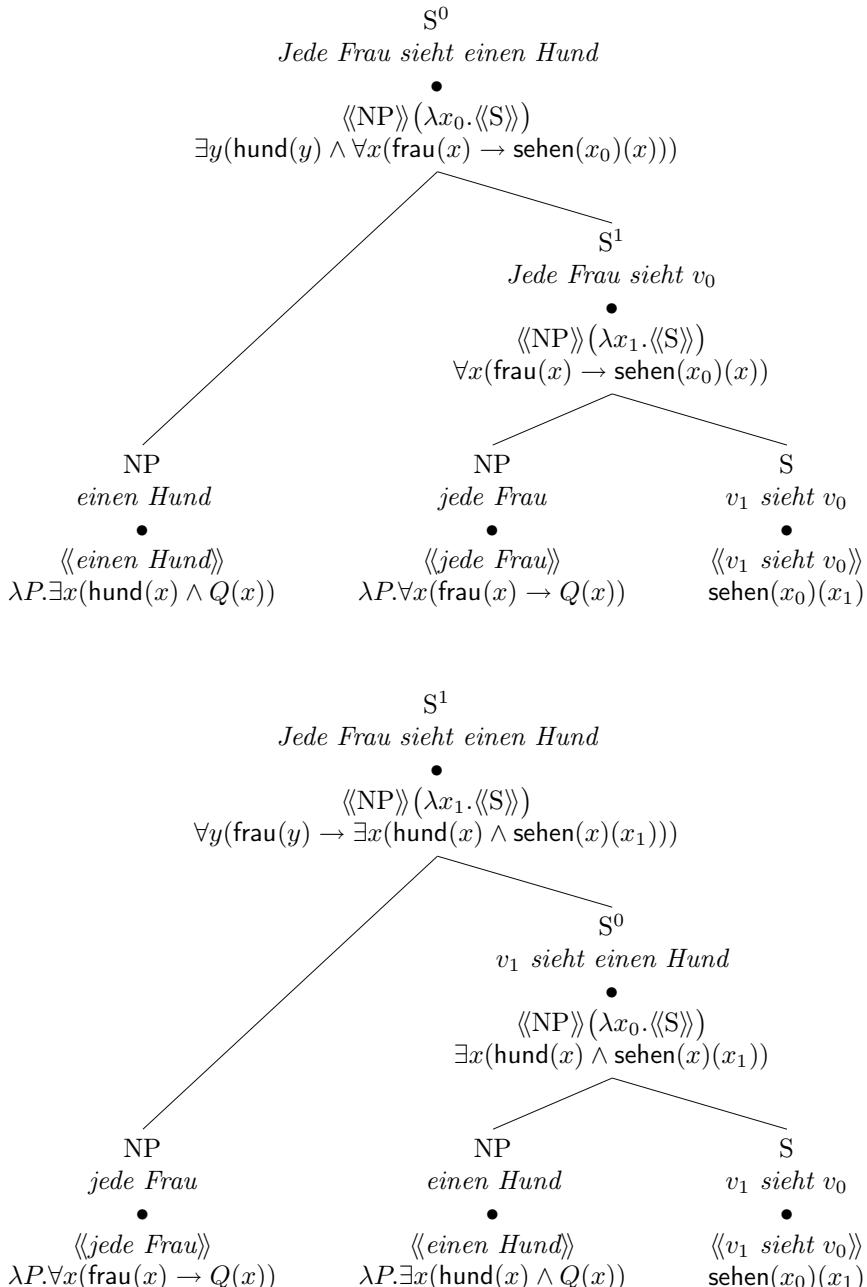


Abbildung 3.46: Kompositionale Ableitungen der beiden Lesarten von (3.97)

(Kamp 1981, Kamp und Reyle 1993). Hauptuntersuchungsgegenstand ist bei der DRT nicht wie in der Montague-Semantik der Satz, sondern der **Diskurs** oder Text, d.h. eine kohärente Satzfolge. Im Gegensatz zur Montague-Semantik betrachtet die DRT die semantische Interpretation nicht als eine direkte Beziehung zwischen syntaktischer Analyse und Modell, sondern zieht eine Zwischenebene der **Diskursrepräsentation** ein. Die Beispiele (3.110) und (3.111) zeigen zwei Mini-Diskurse, bei denen die Wahrheitsbedingungen des jeweils ersten Satzes identisch sind. Trotzdem ist der eine kohärent, der andere aber nicht verständlich. Diese Inkohärenz wird in der DRT auf die Repräsentation zurückgeführt. Repräsentationen scheinen also ein unverzichtbarer Teil einer diskursorientierten semantischen Theorie zu sein.

(3.110) *Ein Bauer schläft. Er schnarcht.*

(3.111) **Nicht jeder Bauer schläft nicht. Er schnarcht.*

Die Bedeutung eines Ausdrucks ist in der DRT vor allem sein Beitrag zur Diskursrepräsentation. Bedeutung wird in dieser linguistisch inspirierten Sichtweise als Instruktion aufgefasst, eine mentale Repräsentation (einen „Gedanken“) zu konstruieren.

Die Repräsentationsebene besteht aus einer besonderen Art von Formeln, den **Diskursrepräsentationsstrukturen** (DRSen), welche die im Diskurs enthaltene Information wiedergeben. Insofern als der Diskurs die Wirklichkeit beschreibt, ist auch die DRS ein partielles Modell der Realität. Die Wahrheit einer DRS kann in einem vollständigen Modell überprüft werden, indem man versucht, die DRS in das Modell einzubetten. Durch diesen Bezug auf Modelle ist in der DRT also auch die modelltheoretische Sichtweise der Logiker präsent. Die DRT ist eine **dynamische Semantik**, d.h. eine Logik, in der die Formeln nicht direkt die Modelle beschreiben (wie in der statischen Logik), sondern den Kontext verändern, wobei Kontext formal als Menge von Modellen definiert ist, die zu einem bestimmten Zeitpunkt im Diskurs noch zulässig sind.

Diskursrepräsentationsstrukturen

In ihrer linguistischen Abdeckung geht die DRT insbesondere in zwei Bereichen über die Montague-Semantik hinaus:

1. in der Behandlung von satzinterner und satzüberschreitender Anaphorik
2. in der Analyse der Zeitformen hinsichtlich ihres Beitrags zur Diskuskohärenz

Der Fokus wird in diesem Unterkapitel auf dem ersten Punkt liegen, der durch die Beispielsätze in (3.112)–(3.114) verdeutlicht wird. Das Problem in den Beispielen ist zuerst einmal nicht ihre Bedeutung (zu allen Beispielen ist eine adäquate prädikatenlogische Übersetzung angegeben), sondern der Prozess, wie man von den Sätzen zu ihrer Bedeutung gelangt.

(3.112) Ein Bauer schläft. Er schnarcht.

$$\exists x(\text{bauer}(x) \wedge \text{schlafen}(x) \wedge \text{schnarchen}(x))$$

(3.113) Jeder Bauer, der einen Esel hat, schlägt ihn.

$$\forall x \forall y (\text{bauer}(x) \wedge \text{esel}(y) \wedge \text{haben}(x, y) \rightarrow \text{schlagen}(x, y))$$

(3.114) Wenn ein Bauer einen Esel hat, schlägt er ihn.

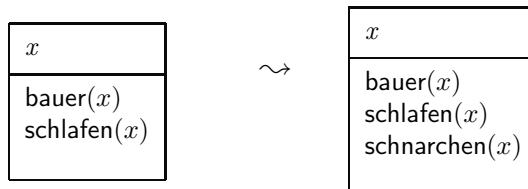
$$\forall x \forall y (\text{bauer}(x) \wedge \text{esel}(y) \wedge \text{haben}(x, y) \rightarrow \text{schlagen}(x, y))$$

In (3.112) ist es mit den Methoden der Montague-Semantik zwar möglich, die gewünschte Formel zu erstellen – man muss nur der Aneinanderreihung von Sätzen dieselbe Semantik geben wie der Konjunktion von Sätzen –, jedoch ist die in der Montague-Semantik benutzte Methode fragwürdig: Mit ihr kann nur für den gesamten Text auf einmal eine Formel erstellt werden, da im gesamten Text Pronomina auftreten können, die von Nominalphrasen vorhergehender Sätze gebunden werden. Intuitiv kann man einem einzelnen Satz aber auch schon einen Sinn geben – die Interpretation sollte **inkrementell** vorgehen.

Im ersten Satz (3.112) wird ein Individuum eingeführt, über das der folgende Satz weitere Aussagen macht. Individuen werden in der DRT als **Diskursreferenten** repräsentiert. Aussagen über die Individuen werden als **atomare Konditionen** kodiert, die wie atomare Prädikatenlogische Formeln interpretiert werden. Eine DRS wird graphisch als ein Kasten dargestellt, in dem oben (im **Universum**) die Diskursreferenten und unten die Konditionen stehen. Das Universum ist in der DRT also die Menge der Diskursreferenten, und als solche von der Domäne eines logischen Modells (für die auch der Begriff Universum gebräuchlich ist; vgl. Unterkapitel 2.1) zu unterscheiden.

Der zweite Satz führt nicht zu einer eigenen DRS, er ergänzt vielmehr die durch den ersten Satz gegebene Information und somit die DRS:

Ein Bauer schläft. *... Er schnarcht.*

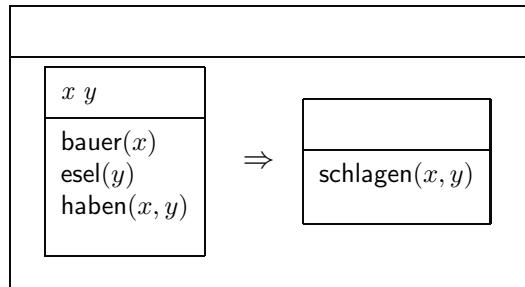


Die Sätze (3.113) und (3.114) (auch **Eselssätze**; engl. *donkey sentences* genannt) illustrieren das Phänomen der so genannten **Eselsanapher**, (engl. *donkey anaphor*), das sich in der Montague-Semantik nicht erklären lässt: Eine indefinite Nominalphrase fungiert als Allquantor, wenn sie in der Restriktion eines Allquantors oder in einem Konditionalsatz auftritt und eine Variable im zugehörigen Skopus bzw. Matrixsatz bindet. Wie in der Behandlung von (3.112) gesehen, werden Existenzquantoren in der DRT nicht explizit eingeführt. Die Art der Quantifikation über die Diskursreferenten im Universum einer DRS ist

kontextabhängig. Allquantoren und Konditionalsätze werden in der DRT als Implikationen interpretiert. Eine Implikation ist eine **komplexe Kondition**, die zwei Unter-DRSen über einen Implikationspfeil verbindet.

Beispiel 3.6.2

Die Übersetzung von (3.113) und (3.114) in eine DRS liefert das folgende Ergebnis.



Der durch *ein Bauer* eingeführte Existenzquantor in dem Konditionalsatz (3.114) wirkt hierbei als Allquantor (wie bei *Jeder Bauer* in (3.114)), weshalb beide Übersetzungen dasselbe Ergebnis liefern. \square

Um nun genau zu verstehen, wie solche DRSen zu interpretieren sind, ist es zunächst notwendig, sie formal zu definieren.

Formale Definition von DRSen

Da komplexe Konditionen selbst wiederum DRSen enthalten, müssen DRSen und DRS-Konditionen gleichzeitig definiert werden.

Definition 3.6.2

1. Eine DRS ist ein Paar $\langle U, Con \rangle$, wo U eine endliche Menge von Diskursreferenten ist (das *Universum*) und Con eine endliche Multimenge von DRS-Konditionen.
2. Wenn P ein Prädikatensymbol der Stelligkeit n ist und x_1, \dots, x_n Diskursreferenten sind, dann ist $P(x_1, \dots, x_n)$ eine (atomare) DRS-Kondition.
3. Negation: Wenn K eine DRS ist, dann ist $\neg K$ eine (komplexe) DRS-Kondition
4. Disjunktion: Wenn K_1, \dots, K_n endlich viele, aber mindestens zwei DRSen sind, dann ist $K_1 \vee \dots \vee K_n$ eine (komplexe) DRS-Kondition
5. Implikation: Wenn K_1 und K_2 zwei DRSen sind, dann ist $K_1 \Rightarrow K_2$ eine (komplexe) DRS-Kondition.

\square

Die DRS in Beispiel 3.6.2 hat nach dieser Definition die folgende Form.

$$\langle \emptyset, \{\langle \{x, y\}, \{\text{bauer}(x), \text{esel}(y), \text{haben}(x, y)\} \rangle\} \Rightarrow \langle \emptyset, \{\text{schlagen}(x, y)\} \rangle \rangle$$

Nach der formalen Definition von DRSen kann die modelltheoretische Interpretation von DRSen definiert werden. Damit wird dann klar, wieso beispielsweise die DRS aus Beispiel 3.6.2 die Sätze (3.113) und (3.114) adäquat repräsentiert.

Modelltheoretische Interpretation von DRSen

Man interpretiert nun DRSen bezüglich eines Modells $\mathcal{M} = (D, F)$ der Prädikatenlogik mit einer Domäne D und einer Funktion F , die jedem Prädikatsensymbol eine der Stelligkeit entsprechende Relation über D zuweist (siehe Unterkapitel 2.1). Weiterhin sei g eine (möglicherweise partielle) Funktion von der Menge der Diskursreferenten in die Domäne D . An die Stelle der Variablen in der Prädikatenlogik treten nun also die Diskursreferenten des Universums einer DRS. Damit macht es Sinn, von g als einer Variablenbelegung zu sprechen. Diese Funktion kann partiell sein, d.h. sie muss nicht für jeden Diskursreferenten definiert sein.

Definition 3.6.3

Eine Variablenbelegung g' **erweitert** eine Variablenbelegung g um eine Menge DR von Diskursreferenten (formal geschrieben als $g' \supseteq_{DR} g$), wenn g' allen Diskursreferenten, für die g definiert ist, denselben Wert zuweist wie g und g' außerdem allen Diskursreferenten in DR einen Wert zuweist. \square

Die Schreibweise $\llbracket \cdot \rrbracket^{\mathcal{M}, g}$ bezeichnet analog zur Darstellung der Semantik der Prädikatenlogik in Unterkapitel 2.1 den Wahrheitswert einer DRS bzw. einer DRS-Kondition im Modell \mathcal{M} unter der Variablenbelegung g . Die Definition der Wahrheit einer DRS ist wie folgt.

Definition 3.6.4

Für DRSen K, K_1, \dots, K_n mit Universen $U_K, U_{K_1}, \dots, U_{K_n}$ und Konditionen $Con_K, Con_{K_1}, \dots, Con_{K_n}$ definiert man $\llbracket \cdot \rrbracket^{\mathcal{M}, g}$ wie folgt.

1. $\llbracket K \rrbracket^{\mathcal{M}, g} = 1$ gdw. es eine Variablenbelegung g' mit $g' \supseteq_{U_K} g$ gibt, sodass für alle Konditionen $C \in Con_K$ gilt: $\llbracket C \rrbracket^{\mathcal{M}, g'} = 1$.
2. $\llbracket P(x_1, \dots, x_n) \rrbracket^{\mathcal{M}, g} = 1$ gdw. $\langle g(x_1), \dots, g(x_n) \rangle \in F(P)$.
3. $\llbracket \neg K \rrbracket^{\mathcal{M}, g} = 1$ gdw. $\llbracket K \rrbracket^{\mathcal{M}, g} = 0$.
4. $\llbracket K_1 \vee \dots \vee K_n \rrbracket^{\mathcal{M}, g} = 1$ gdw. $\llbracket K_1 \rrbracket^{\mathcal{M}, g} = 1$ oder \dots oder $\llbracket K_n \rrbracket^{\mathcal{M}, g} = 1$.
5. $\llbracket K_1 \Rightarrow K_2 \rrbracket^{\mathcal{M}, g} = 1$ gdw. für alle Variablenbelegungen g mit $g' \supseteq_{U_{K_1}} g$, die alle Konditionen $C \in Con_{K_1}$ wahr machen (d.h. $\llbracket C \rrbracket^{\mathcal{M}, g'} = 1$), gilt, dass $\llbracket K_2 \rrbracket^{\mathcal{M}, g'} = 1$.

\square

Eine DRS ist wahr gdw. es eine Variablenbelegung g' gibt (hier kommt die implizite Existenzquantifikation ins Spiel), so dass g' für alle Diskursreferenten im Universum der DRS definiert ist und g' alle Konditionen der DRS wahr

macht (Punkt 1 der Definition). Eine DRS $\langle\{x_1, \dots, x_n\}, \{C_1, \dots, C_2\}\rangle$ ist also äquivalent zu der prädikatenlogischen Formel

$$\exists x_1 \dots \exists x_n (C'_1 \wedge \dots \wedge C'_n),$$

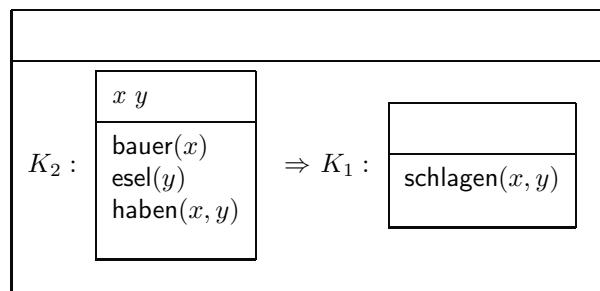
wobei C'_i jeweils die prädikatenlogische Übersetzung der Kondition C_i ist. Die Interpretationsdefinitionen in den Punkten 2–4 entsprechen denen der Prädikatenlogik. Interessant ist wieder die Definition der Implikation in 5: Hier wird über die Variablenbelegungen, die die linke DRS K_1 überprüfen, (und damit über die Diskursreferenten im Universum dieser DRS) allquantifiziert. Die rechte DRS K_2 wird unter der Variablenbelegung g' ausgewertet, in der die Diskursreferenten der DRS K_1 definiert sind. Die Kondition $\langle\{x_1, \dots, x_n\}, \{C_1, \dots, C_2\}\rangle \Rightarrow K_2$ ist also äquivalent zur prädikatenlogischen Formel

$$\forall x_1 \dots \forall x_n (C'_1 \wedge \dots \wedge C'_n \rightarrow K'_2).$$

Subordination und Zugänglichkeit

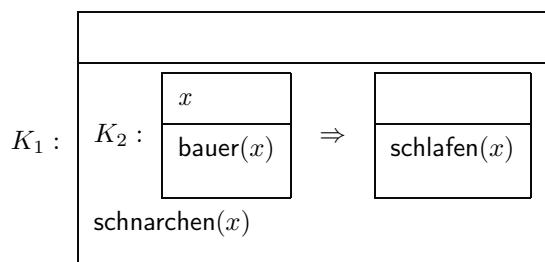
In der DRS in Beispiel 3.6.2 – wiederholt als (3.115) – zeigt sich, dass ein Diskursreferent in einer DRS-Kondition einer DRS K_1 auch durch einen Diskursreferenten gebunden werden kann, der nicht im Universum derselben DRS K_1 , sondern im Universum einer anderen DRS K_2 auftritt.

(3.115) *Jeder Bauer, der einen Esel hat, schlägt ihn.*



Das nicht wohlgeformte Beispiel (3.116) zeigt aber, dass der bindende Diskursreferent nicht in jeder DRS K_2 stehen darf.

(3.116) **Jeder Bauer schläft. Er schnarcht.*



In der Kondition $schnarchen(x)$ in K_1 kann x nicht durch den Diskursreferenten x im Universum von K_2 gebunden werden. Vielmehr müssen die DRS K_2 mit dem Binder und die DRS K_1 mit dem zu bindenden Element in einer bestimmten strukturellen Beziehung stehen: K_2 muss von K_1 aus **zugänglich** sein. In Beispiel (3.116) ist die Zugänglichkeitsbeziehung verletzt, daher ist es nicht wohlgeformt. Zugänglichkeit wird über **Subordination** definiert.

Definition 3.6.5

1. Eine DRS K_1 ist einer DRS K_2 **unmittelbar subordiniert** ($K_1 \prec K_2$) gdw. K_1 Bestandteil einer komplexen DRS-Kondition von K_2 ist.
2. Eine DRS K_1 ist einer DRS K_2 **subordiniert** ($K_1 < K_2$) gdw. $K_1 \prec K_2$ oder $\exists K_3 : K_1 \prec K_3 < K_2$.
3. Eine DRS K_1 ist einer DRS K_2 **schwach subordiniert** ($K_1 \leq K_2$) gdw. $K_1 < K_2$ oder $K_1 = K_2$.
4. Eine DRS K_2 ist von einer DRS K_1 aus **zugänglich** gdw. gilt: $K_1 \leq K_2$ oder K_2 tritt als erstes Argument einer Implikation $K_2 \Rightarrow K_3$ auf und $K_1 \leq K_3$.
5. Das Vorkommen eines Diskursreferenten x in einer Kondition der DRS K_1 kann nur dann von einem Vorkommen des Diskursreferenten x im Universum einer DRS K_2 gebunden werden, wenn K_2 von K_1 aus zugänglich ist.

□

Mit dieser Definition ist klar, dass die DRS in (3.116) nicht wohlgeformt ist, wohl aber die DRS in (3.115): K_2 ist von K_1 aus zugänglich, da K_2 nach Punkt 4 der Definition erstes Argument einer Implikation $K_2 \Rightarrow K_3$ ist und $K_1 K_3$ schwach subordiniert ist, da sogar $K_1 = K_3$ gilt.

Ein Fragment des Deutschen

Wie die Montague-Semantik behandelt auch die DRT ein Fragment. Das in diesem Abschnitt betrachtete Fragment des Deutschen orientiert sich an dem Grammatikfragment der ersten beiden Kapitel von (Kamp und Reyle 1993). Die Tabelle 3.10 und die Abbildung 3.47 zeigen Lexikon und Grammatik dieses extensionalen Fragments.

DRS-Konstruktion

Zur DRS-Konstruktion wird der Syntaxbaum für einen neu zu bearbeitenden Satz zunächst in die globale DRS für den Diskurs eingetragen. Dann werden so viele **Konstruktionsregeln** als möglich angewandt, die die syntaktische Analyse in DRSen, Diskursreferenten und Konditionen zerlegen.

Kategorie	Wörter	Bemerkung
PN	{Fritz, Maria, Meier, Anna Karenina}	Eigennamen
N	{Mann, Frau, Bauer, Esel, Pferd, Buch}	Nomen
Det	{jeder, ein}	Artikel
Vi	{hustet, schläft, stinkt}	intransitive Verben
Vt	{hat, liebt, mag, hasst, schlägt}	transitive Verben
Pro	{er, sie, es, ihn, ihm, ihr}	Pronomen
RPro	{der, die, das}	Relativpronomen
Cnj	{und, oder}	Satzkonjunktionen

Tabelle 3.10: Lexikoneinträge

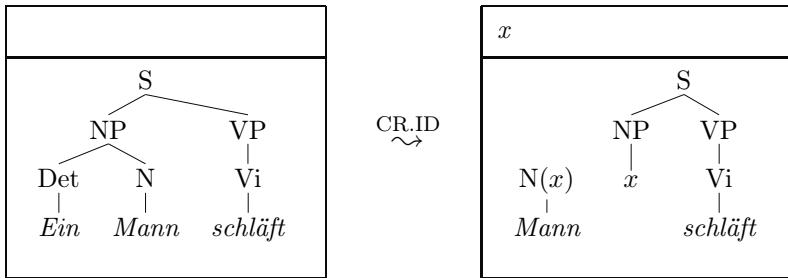
S	→	wenn S dann S	NP(Gap)	→	∅
S	→	S Cnj S	N	→	N Cnj N
S	→	NP VP	N	→	N RelS
S(Gap)	→	NP(Gap) VP	RelS	→	RPro S(Gap)
NP	→	NP oder NP	VP	→	VP Cnj VP
NP	→	Det N	VP	→	Vi
NP	→	PN	VP	→	Vt NP
NP	→	Pro			

Abbildung 3.47: Syntaxregeln

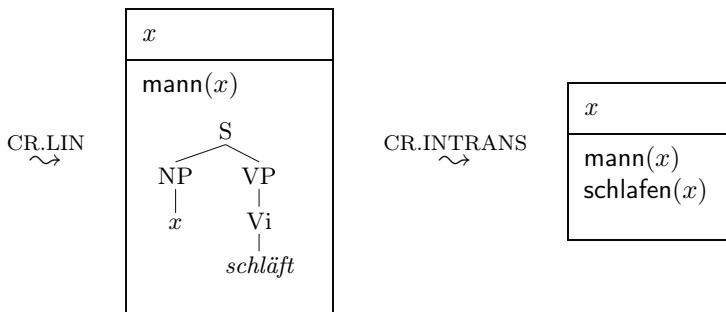
Die einzelnen Konstruktionsregeln sollen anhand einiger Beispiele erläutert werden. Eine vollständige Beschreibung aller Konstruktionsregeln kann in (Kamp und Reyle 1993) nachgelesen werden. Zunächst soll die Übersetzung des Satzes

(3.117) *Ein Mann schläft.*

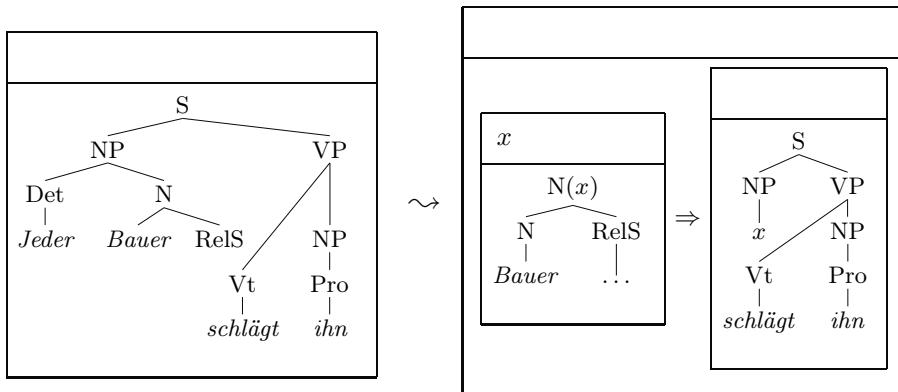
betrachtet werden. Im folgenden Konstruktionsschritt wird zunächst die Konstruktionsregel CR.ID (CR steht für engl. *Construction Rule*) angewandt, die die indefinite NP in einen neuen Diskursreferenten im Universum der DRS übersetzt. Im Baum wird die NP durch den Diskursreferenten ersetzt; der Nomenteil der NP wird ein neuer eigener Baum, der ebenfalls mit dem Diskursreferenten markiert wird.



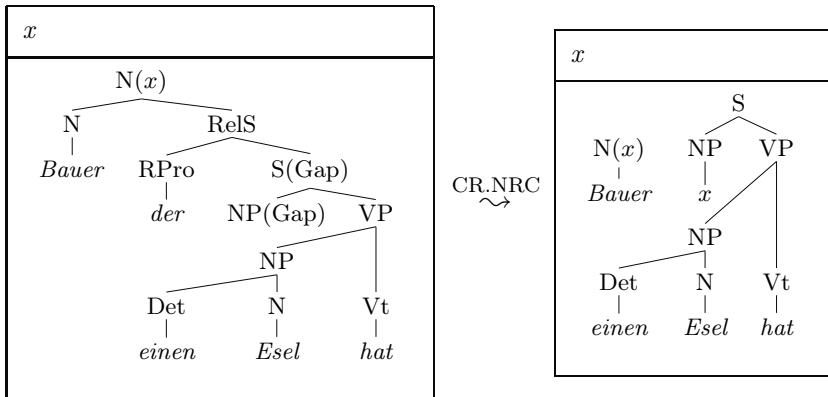
Die Regel CR.LIN ersetzt ein Nomen, das mit einem Diskursreferenten markiert ist, durch eine Kondition. Die Regel CR.INTRANS erstellt aus einem transitiven Verb und seinem Subjekt eine Kondition, was zum Endergebnis der Übersetzung führt.



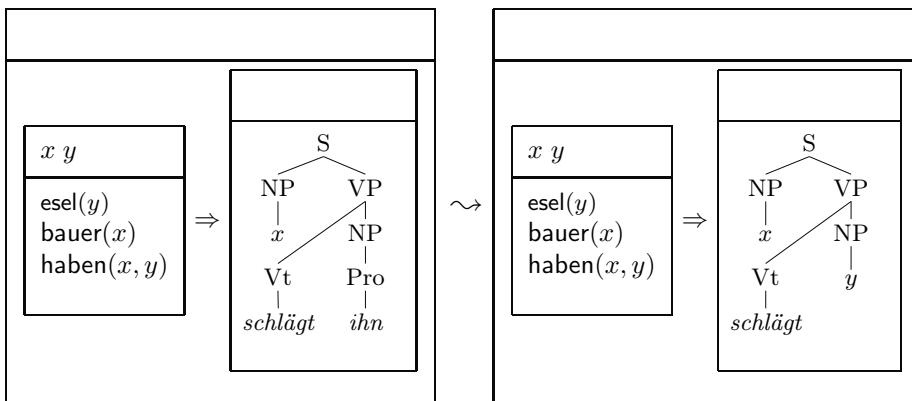
Nun betrachten wir die DRS-Konstruktion für den Eselssatz (3.113) im Detail. Die Regel CR.EVERY übersetzt das Wort *jeder* in eine komplexe Implikationskondition. In die linke DRS, die Restriktion, wird ein neuer Diskursreferent und das mit dem Diskursreferenten markierte Nomen eingetragen. In die rechte DRS, den Skopus, kommt der Satz, der die quantifizierte NP enthält; die NP wird dabei durch den Diskursreferenten ersetzt.



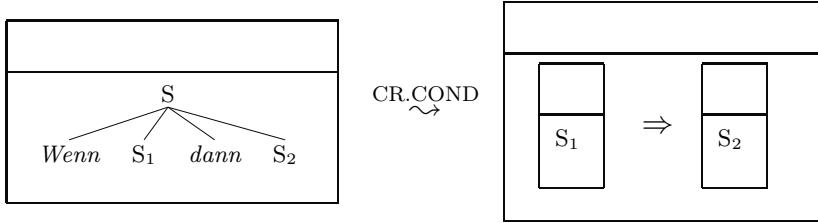
Die Regel CR.NRC bearbeitet den Relativsatz. An die Stelle der leeren NP wird der Diskursreferent des Nomens gesetzt. Die Übersetzung der Restriktions-DRS ist damit wie folgt:



Die Übersetzung der NP *einen Esel* erfolgt wieder mit der Regel CR.ID und die Übersetzung des transitiven Verbs mittels einer Regel CR.TRANS, die ähnlich wie CR.INTRANS eine entsprechende Kondition erzeugt. In der Skopus-DRS übersetzt die Regel CR.PRO das Pronomen in einen Diskursreferenten, der im Universum einer von der lokalen DRS aus zugänglichen DRS steht. Im Beispiel sind zwei Diskursreferenten zugänglich, von denen aufgrund syntaktischer Restriktionen der Diskursreferent für den Esel gewählt wird.



Durch eine letztmalige Anwendung der Regel CR.TRANS zur Übersetzung von *schlägt* erhält man das endgültige Ergebnis von (3.113) aus Beispiel 3.6.2. Wie oben erwähnt, liefert die Übersetzung von (3.114) dasselbe Ergebnis. Verantwortlich dafür ist die Regel CR.COND, die Hauptsatz und Nebensatz eines konditionalen Gefüges zu einer Implikationskondition verknüpft. Schematisch dargestellt wirkt diese Regel also folgendermaßen:



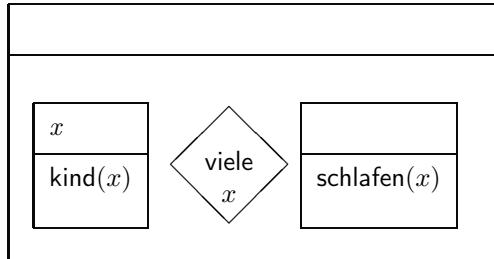
Weitere Entwicklungen

Wir wollen noch eine weitere Entwicklung in der DRT charakterisieren, nämlich die Einbindung generalisierter Quantoren.

Es ist sehr einfach, in das Fragment noch generalisierte Quantoren einzufügen. Generalisierte Quantoren werden als **Duplexbedingungen** visualisiert, d.h. zwischen Restriktion und Skopos wird eine Raute gesetzt, in der der Quantor und die gebundene Variable vermerkt werden. Die Konstruktionsregel entspricht der Regel für den Allquantor CR.EVERY. Die Definition der Zugänglichkeit 3.6.5 wird dahingehend erweitert, dass die Restriktion eines Quantors von seinem Skopos und allen darin eingebetteten DRSSen aus zugänglich ist. Der Satz

(3.118) *Viele Kinder schlafen.*

hat beispielsweise die Übersetzung



In der flachen Schreibweise würde man diese Duplexbedingung als

$viele(x, \langle \{x\}, \{\text{kind}(x)\} \rangle, \langle \emptyset, \{\text{schlafen}(x)\} \rangle)$

notieren, was so zu lesen ist wie die Anwendung des generalisierten Quantors zu *viele* aus Tabelle 3.6 auf die Menge *kind* als Restriktor und *schlafen* als Skopos. Weitere Entwicklungen wie die Integration von Pluralanaphern und Tempus sind in Kamp und Reyle (1993), Kapitel 4 und 5 beschrieben.

Bewertung der DRT als Computersemantik

Die DRT ist bis dato die semantische Theorie, die die meisten Phänomene integriert, und bietet sich daher als Grundlage für eine computersemantische Beschreibung an, wo es ja auch darum geht, eine möglichst große Abdeckung zu erzielen.

Bisher wurden Beschreibungsformalismen und semantische Theorien vorgestellt, die allesamt die natürliche Sprache tiefer analysieren und daher neue Ambiguitäten einführen. Der folgende Abschnitt geht auf die explizite Behandlung von Ambiguität ein und stellt entsprechende Techniken zur Verarbeitung hochambiger sprachlicher Ausdrücke vor.

3.6.4 Ansätze zur Unterspezifikation

Bisher hat sich dieses Unterkapitel damit beschäftigt, mit welchen Repräsentationen und Interpretationen man die Bedeutung eines sprachlichen Ausdrucks modellieren kann. Oft hat ein Ausdruck sehr viele mögliche Interpretationen, d.h. er ist hochgradig ambig. Satz (3.119) z. B. hat aufgrund der Kombinationsmöglichkeiten der Quantoren und der PP-Anbindung elf unterschiedliche Interpretationen.

(3.119) *Jeder Mann sah drei Kinder mit einem Fernglas.*

Z. B. kann es drei Kinder mit je einem Fernglas gegeben haben, die jeder Mann sah, oder es gab drei Kinder und jeder Mann sah diese mit einem (möglicherweise unterschiedlichen) Fernglas, oder für jeden Mann gab es eine Gruppe von drei Kindern, die zusammen ein Fernglas hatten, oder jeder Mann hatte ein Fernglas, mit dem er eine Gruppe von drei Kindern beobachtete, oder ...

Wie oben erwähnt, kann man in Montagues System jede Lesart durch Hineinquantifizieren von Quantorenbedeutungen in die Satzbedeutung in einer bestimmten Reihenfolge erhalten. Allgemein betrachtet gibt es damit bei einem Satz mit n Quantoren $n!$ verschiedene Reihenfolgen des Hineinquantifizierens, d.h. die Zahl der theoretisch ableitbaren Lesarten ist exponentiell in der Zahl der Quantoren der Worte. Satz (3.120) hat in einem in Stuttgart entwickelten LFG-Parser bereits 552 syntaktische Lesarten und etwa 35000 semantische Lesarten.

(3.120) *Als er wieder zurückkam, sah der 23jährige den BMW mit einem Unbekannten am Steuer entschwinden.*

Den Prozess, aus einer theoretisch möglichen Menge von Lesarten in einer konkreten Situation die tatsächlich gemeinte Lesart zu bestimmen, nennt man **Disambiguierung**. Disambiguierung ist aber ein schweres Problem: Sogar kompetente Sprecher haben manchmal Schwierigkeiten, die korrekte Lesart eines vorliegenden Ausdrucks zu bestimmen. Umso mehr gilt dies für den Computer, der ja bei der Semantikkonstruktion auf kein Weltwissen zurückgreifen kann.

Auf der anderen Seite muss man sich zur Lösung vieler Aufgaben aber auch gar nicht auf eine konkrete Lesart festlegen. Zum einen kann das daran liegen, dass die zu schließende Information in derselben Weise ambig ist wie die Information in den Prämissen (z.B. die Ambiguität der Skopusbeziehung zwischen Existenz- und Allquantor in Satz (3.121)). Häufig liegt dieser Fall bei der Übersetzung von einer Sprache in eine andere vor: Der Zielsprachliche Ausdruck ist in derselben Weise ambig wie der quellsprachliche Ausdruck (**ambiguitätserhaltende Übersetzung**, siehe die Sätze (3.122) und (3.123)).

- (3.121) *Wenn Fritz morgen kommt, dann sind alle Jungen zu einer Party gekommen.* Fritz kommt morgen. → Alle Jungen sind zu einer Party gekommen.
- (3.122) *Wo ist die Maus? ~ Where is the mouse?*
(Nagetier oder Eingabegerät)
- (3.123) *I saw a fish in every pond. ~ In jedem Teich habe ich einen Fisch gesehen.* (Skopusambiguität)

Zum anderen braucht man sich dann nicht auf eine konkrete Lesart der gesamten Prämisse festzulegen, wenn die zu ziehende Inferenz nur auf einem Teil der Prämisse basiert. Im Satz (3.124) muss z.B. nicht festgelegt werden, ob es sich bei dem Bus um ein Fahrzeug oder eine Computerkomponente handelt, um die gewünschte Inferenz zu ziehen.

- (3.124) *Maria rannte zur Bushaltestelle. Der Bus, der dort stand, war defekt. Sie musste 5 Minuten warten.* → Maria musste 5 Minuten warten.

Es ist also erstrebenswert, eine Repräsentationsebene aufzubauen, auf der bestimmte Ambiguitäten nicht aufgelöst sind, d.h. die neutral gegenüber diesen Ambiguitäten ist. Eine solche Repräsentation ist bzgl. der Ambiguitäten **unterspezifiziert**.

Man betrachte nun aber folgenden kurzen Diskurs, der ins Deutsche übersetzt werden soll.

- (3.125) *I saw a fish in every pond. I caught the fish.*

Die Übersetzung des ersten Satzes (beispielsweise in Deutsch) ist einfach, wenn die Skopusambiguität unterspezifiziert gehalten werden kann (siehe (3.123)). Bei der Übersetzung des nächsten Satzes muss man aber entscheiden, ob von einem oder mehreren Fischen die Rede ist, d.h. die Ambiguität des ersten Satzes ist nun doch aufzulösen. Es ergibt sich als allgemeine Anforderung an die unterspezifizierte Repräsentation, dass die spezifischen Lesarten sehr einfach aus ihr extrahierbar sein sollen. Ein Unterspezifikationsformalismus muss also nicht nur Repräsentationen für unterspezifizierte und spezifische Repräsentationen bereitstellen, sondern auch eine **Disambiguierungsroutine**, die sehr einfach zu jeder unterspezifizierten Repräsentation alle spezifischen Repräsentationen aufzählt, die von ihr überdeckt werden. Eine spezielle Anforderung an die Disambiguierungsroutine ist die **Monotonie** (Alshawi und Crouch 1992):

Monotone Disambiguierung:

Um zu den spezifischen Lesarten zu gelangen, sollte es nicht nötig sein, die Repräsentation tiefgreifend zu verändern; es sollte genügen, mehr Information hinzuzufügen.

Satz (3.124) hält eine weitere Forderung bereit: Der Formalismus sollte es ermöglichen, bestimmte Ambiguitäten aufzulösen (den Bezug des Pronomens *sie*) und andere unterspezifiziert zu halten (die erwähnte lexikalische Ambiguität von

Bus). Der Unterspezifikationsformalismus sollte also nicht nur voll unterspezifizierte, sondern auch teilweise unterspezifizierte Repräsentationen zur Verfügung stellen; die Disambiguierungsroutine sollte auch eine teilweise Disambiguierung von Repräsentationen ermöglichen.

Hole Semantics

Einer der ersten Formalismen zur Behandlung von Skopusbäiguitäten auf Basis der DRT wurde in Reyle (1993) vorgestellt. In Bos (1995) sowie Bos (2002) wurde dieselbe Unterspezifikationsmethode verallgemeinert und auf die Prädikatenlogik angewandt.

Die erste Einsicht, die diesen Formalismen zugrunde liegt, ist, dass sich die einzelnen Lesarten bei Skopusbäiguitäten nicht in den vorkommenden Prädikaten und Variablen unterscheiden, sondern einzig und allein in der Stellung bestimmter Teilformeln in der Gesamtformel. Wenn man nun etwas über diese Stellung sagen will, so muss man die Stellung der Formeln durch so genannte **Labels** l_i explizit benennen. Jeder Teilformel (vom Typ t) wird ein Label zugewiesen, das in der Repräsentation vor die Teilformel gesetzt wird. Wenn eine Teilformel als Argument einer anderen Teilformel auftritt, so wird die eingebettete Teilformel durch ihr Label ersetzt. Das Label für die gesamte Formel heißt **Toplabel** und wird gesondert vermerkt. Eine Repräsentation ist jetzt ein Paar aus Toplabel und einer Menge gelabelter Constraints. Die Mengen der gelabelten Constraints für die beiden Lesarten (3.127) des Satzes (3.126) sind in (3.128) angegeben. Die Toplabels sind l_1 bzw. l_3 .

(3.126) *Jedes Kind sieht ein Bild.*

(3.127) (a) $\forall x(\text{kind}(x) \rightarrow \exists y(\text{bild}(y) \wedge \text{sehen}(x, y)))$

(b) $\exists y(\text{bild}(y) \wedge \forall x(\text{kind}(x) \rightarrow \text{sehen}(x, y)))$

(3.128) (a) $\langle l_1, \{ l_1 : \forall x(l_2 \rightarrow l_3),$

$l_2 : \text{kind}(x),$

$l_3 : \exists y(l_4 \wedge l_5),$

$l_4 : \text{bild}(y),$

$l_5 : \text{sehen}(x, y) \} \rangle$

(b) $\langle l_3, \{ l_1 : \forall x(l_2 \rightarrow l_5),$

$l_2 : \text{kind}(x),$

$l_3 : \exists y(l_4 \wedge l_1),$

$l_4 : \text{bild}(y),$

$l_5 : \text{sehen}(x, y) \} \rangle$

Da in allen Lesarten exakt dieselben Teilformeln vorkommen, müssen sie in der unterspezifizierten Repräsentation nur einmal aufgeführt werden. Um die Landepunkte in den Skopusdomänen variabel zu halten, werden auch für die Skopusbäiguitäten explizite Labels eingeführt. In Anlehnung an Bos (1995) werden Labels für Skopusbäiguitäten in diesem Unterkapitel als **Holes** h_i bezeichnet. Für (3.128) ergibt sich damit folgende Darstellung:

(3.129) $\langle h_1, \{ l_1 : \forall x(h_2 \rightarrow h_3),$

$l_2 : \text{kind}(x),$

$l_3 : \exists y(h_4 \wedge h_5),$

$l_4 : \text{bild}(y),$

$l_5 : \text{sehen}(x, y) \} \rangle$

Es bleibt die Frage, wie festgelegt wird, welches Label sich auf welches Hole beziehen kann. In (3.129) gibt es offensichtlich keine Lesart, in der l_2 in die Skopusdomäne h_3 kommt. Anstatt direkt die Label-Hole-Zuordnung zu kodieren, werden die **Subordinationsbeziehungen** zwischen Labels und Holes notiert. Jedes Hole (mit Ausnahme des Toplabels natürlich) ist genau einem Label direkt subordiniert, wie sich direkt aus den Teilformeln in (3.129) ablesen lässt. Es bleibt also nur zu bestimmen, welches Label welchem Hole subordiniert ist. Abbildung 3.48 zeigt die Situation für die beiden Lesarten in (3.128).

	h_1	h_2	h_3	h_4	h_5
l_1	(a) (b)				(b)
l_2	(a) (b)	(a) (b)			(b)
l_3	(a) (b)		(a)		
l_4	(a) (b)		(a)	(a) (b)	
l_5	(a) (b)		(a) (b)		(a) (b)

Abbildung 3.48: Subordinationsbeziehungen

In die unterspezifizierte Repräsentation werden nur diejenigen Subordinationsbeziehungen eingetragen, die in allen Lesarten gelten und die sich nicht über Transitivität ergeben (d.h. im Beispiel $l_1 \leq h_1, l_2 \leq h_2, l_3 \leq h_1, l_4 \leq h_4, l_5 \leq h_3, l_5 \leq h_5$). Graphisch kann die Subordinationsordnung in einem Hasse-Diagramm (vgl. Unterkapitel 2.3) veranschaulicht werden, das für das Beispiel in Abbildung 3.49 dargestellt ist.

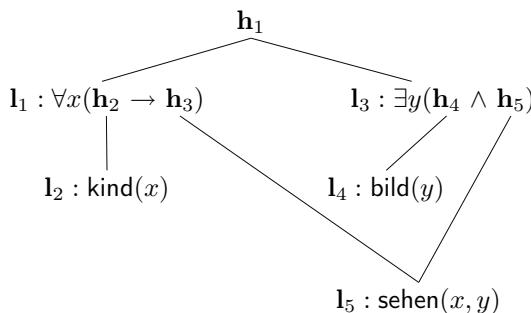


Abbildung 3.49: Hassediagramm der Subordinationsconstraints

Dabei müssen nur einige globale Wohlgeformtheitskriterien beachtet werden:

1. Subordination ist eine partielle Ordnungsrelation, d.h. reflexiv, transitiv und antisymmetrisch (vgl. die Subsumptionsrelation in Unterkapitel 2.3). Aus der Antisymmetrie folgt, dass der Subordinationsgraph keine Zyklen enthalten darf.

2. Alle Labels sind dem Toplabel subordiniert, d.h. alle Teilformeln sind in der Gesamtformel enthalten.
3. Wenn ein Prädikat l_2 zwei Holes h_1 und h_2 verbindet (d.h. wenn die beiden Holes in den Argumentstellen eines durch l_2 bezeichneten Prädikats stehen), dann darf es kein Label l_1 geben, das beiden Holes subordiniert ist. (Eine Teilformel kann also nicht an zwei unterschiedlichen Stellen einer Formel stehen.) Die Subordinationsordnung beschreibt einen Baum.

Eine (teilweise) Disambiguierung, also Ambiguitätsauflösung, wird durch Hinzufügen beliebiger weiterer Subordinationsconstraints erreicht. Damit ist das Postulat der monotonen Disambiguierung erfüllt, denn die Repräsentation wird nur erweitert und nicht grundlegend verändert.

Durch die Wohlgeformtheitsrestriktionen werden indirekt auch die möglichen Disambiguierungsschritte eingeschränkt. Der Subordinationslink zwischen l_2 und h_3 , der oben diskutiert wurde, verletzt z.B. die Wohlgeformtheitsrestriktion 3, weil damit Label l_2 sowohl unter h_2 als auch unter h_3 zu stehen kommt. Ein erlaubter Subordinationslink ist beispielsweise $l_1 \leq h_5$, der bewirkt, dass der Allquantor im Skopuss des Existenzquantors erscheint. Die Repräsentation mit diesem zusätzlichen Link steht also für die (eindeutige) Lesart (3.127a) von (3.126), in der *ein Bild* Skopus über *jedes Kind* nimmt.

Unterspezifizierte Diskursrepräsentationsstrukturen

Um die beschriebene Unterspezifikationsmethode auf DRSen anwenden zu können, muss jeder Skopusdomäne, d.h. jeder DRS, ein eigenes Label gegeben werden. Eine DRS kann viele Konditionen und Diskursreferenten im Universum enthalten, daher können DRS-Labels im Gegensatz zu Prädikatenlabels auch auf mehrere Constraints verweisen, wodurch sogenannte **unterspezifizierte Diskursrepräsentationsstrukturen** (UDRSen) entstehen. In (3.131) sind zwei Lesarten des Satzes (3.130) in Label-Darstellung angegeben.

(3.130) *Jedes Kind gab den meisten Besuchern ein Bild.*

(3.131)

(a) $\langle l_1, \{ l_1 : \text{jeder}(x, l_2, l_3)$	$l_2 : x$	$l_3 : \text{die_meisten}(y, l_4, l_5)$	(b) $\langle l_3, \{ l_1 : \text{jeder}(x, l_2, l_5)$
$l_2 : \text{kind}(x)$			$l_2 : x$
$l_3 : \text{die_meisten}(y, l_4, l_5)$		$l_3 : \text{die_meisten}(y, l_4, l_1)$	
$l_4 : y$		$l_4 : y$	
$l_4 : \text{besucher}(y)$		$l_4 : \text{besucher}(y)$	
$l_5 : z$		$l_5 : z$	
$l_5 : \text{bild}(z)$		$l_5 : \text{bild}(z)$	
$l_5 : \text{geben}(x, y, z) \})$		$l_5 : \text{geben}(x, y, z) \})$	

Werden alle variablen Labelpositionen durch neue Labels ersetzt und die allen sechs Lesarten gemeinsamen Subordinationsbeziehungen in die unterspezifizierte

Repräsentation übernommen, ergibt sich die folgende UDRS, die in Abbildung 3.50 noch einmal grafisch dargestellt ist.

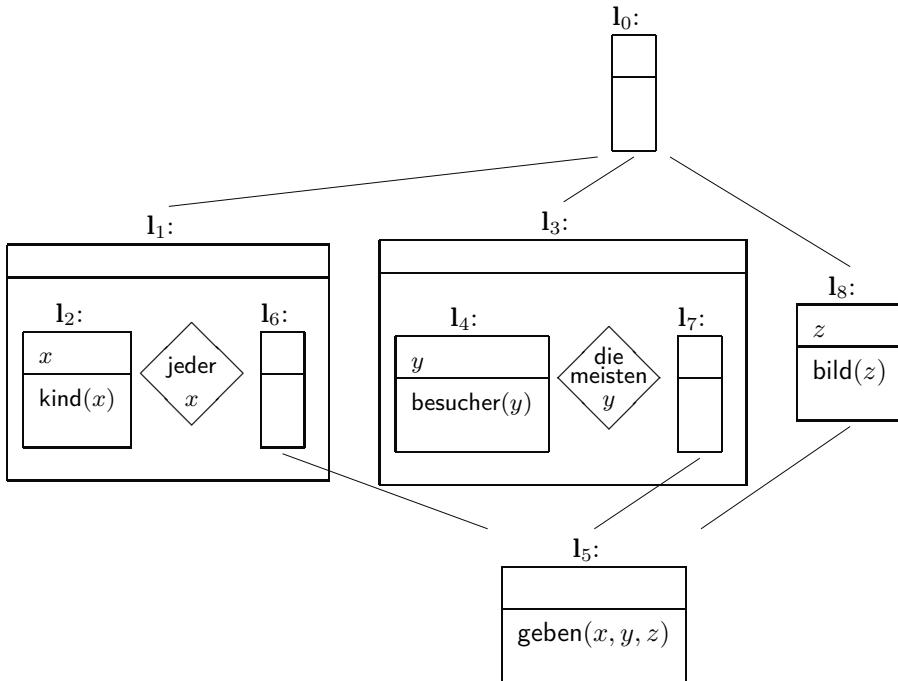
$$\langle l_0, \{ l_1 : \text{jeder}(x, l_2, l_6), l_2 : x, l_2 : \text{kind}(x), l_3 : \text{die_meisten}(y, l_4, l_7), \\ l_4 : y, l_4 : \text{besucher}(y), l_8 : z, l_8 : \text{bild}(z), l_5 : \text{geben}(x, y, z) \}, \\ \{ l_1 \leq l_0, l_3 \leq l_0, l_8 \leq l_0, l_5 \leq l_6, l_5 \leq l_7, l_5 \leq l_8 \} \rangle$$


Abbildung 3.50: UDRS in graphischer Darstellung

Die Semantikkonstruktion wird bei Verwendung des UDRS-Mechanismus stark vereinfacht: Alle Constraints werden im Lexikon eingeführt. Die wesentliche semantische Kompositionsoperation ist nicht funktionale Applikation, sondern Mengenvereinigung. Die syntaktische Struktur wird nur dazu benutzt, um Diskursreferenten und Labelpositionen geeignet zu koindizieren. Ein Vorteil der vereinfachten Semantikkonstruktion ist es, dass semantische Ambiguitäten nun gleichzeitig mit syntaktischen und lexikalischen behandelt werden können, wodurch es möglich wird, jedem wohlgeformten Diskurs genau eine underspezifizierte Repräsentation zuzuweisen. Man beachte auch, dass die UDRS-Repräsentation quadratischen Platz einnimmt bzgl. der Anzahl der Worte im Diskurs. Die Komplexität der Erstellung einer syntaktischen Repräsentation wird also durch die Erweiterung auf die Semantik nicht gesteigert.

Neben der Hole Semantik und der UDRT gibt es noch eine Reihe anderer Unterspezifikationsansätze, die in ähnlicher Weise funktionieren. Zu nennen sind

hier vor allem die **Minimal Recursion Semantics** (MRS; Copestake et al. 2005), die als Semantikkomponente in der HPSG definiert wurde und dort oft Verwendung findet, und Ansätze, die mit **Dominance Constraints** arbeiten (Koller 2004).

3.6.5 Lexikalische Semantik

Bislang war in diesem Unterkapitel von der Satz- bzw. Diskurssemantik die Rede. Die grundlegende Fragestellung hinter den Ansätzen zur Satzsemantik war, wie kompositionell aus den Bedeutungen der Konstituenten die Bedeutung des jeweiligen Satzes berechnet werden kann. Hierbei wurde bislang immer so getan, als ob die Wörter bzw. genauer die lexikalischen Einheiten in jedem Kontext, in dem sie verwendet werden, immer dieselbe Bedeutung haben. Diese Annahme, die Bedeutung eines Wortes sei somit immer als genau ein logisches Prädikat zu behandeln, ist jedoch falsch. So hätten die DRSSen für die Sätze (3.132) und (3.133) die in Abbildung 3.51 angegebene Übersetzung.

(3.132) *Karl kennt das Buch vollständig.*

(3.133) *Karl kennt das Buch im Schaufenster.*

Diese DRSSen lassen jedoch wichtige Aspekte der lexikalischen Bedeutung außer Acht. Die DRSS-Bedingung *buch(y)* besagt lediglich, dass der Referent *y* die Eigenschaft besitzt, ein Buch zu sein. Die Bedeutung von *Buch* variiert aber im jeweiligen Kontext. *Buch* referiert in Satz (3.132) nicht auf ein physikalisches Objekt, sondern auf die Information, dessen Träger dieses Objekt ist. In (3.133) hingegen kann *Buch* sowohl auf das Objekt, als auch auf die Information referieren. Allerdings wird mittels der lokalen Präpositionalphrase *im Schaufenster* das Objekt lokalisiert und nicht die Information. Das Nomen *Schaufenster* besitzt ebenfalls mehrere Bedeutungen. In Satz (3.133) referiert es auf einen lokalen Bereich hinter der Schaufensterscheibe, der von der Straße aus direkt einsehbar ist. In einem Satz wie *Das Schaufenster zerbrach* referiert das Nomen jedoch auf

<i>e x y</i>	<i>e x y z</i>
$\text{karl}(x)$ $\text{buch}(y)$ $\text{kennen}(e,x,y)$ $\text{vollständig}(e)$	$\text{karl}(x)$ $\text{buch}(y)$ $\text{schaufenster}(z)$ $\text{kennen}(e,x,y)$ $\text{in}(y,z)$

Abbildung 3.51: DRSSen für die Sätze 3.132 (links) und 3.133 (rechts)

diese Scheibe. Nicht zu vergessen sei die Bedeutung der Präposition *in*, die in Satz (3.133) lokal interpretiert wird, aber auch eine temporale Bedeutung oder

eine Bedeutung der Art und Weise haben kann. Schließlich besitzt das Verb *kennen* ebenfalls mehrere Bedeutungen, die allesamt mit Informationsbesitz zu tun haben, aber höchst unterschiedlich ausdifferenziert werden, wie diese Beispiele zeigen:

- (3.134) *Karl kennt die Prädikatenlogik.*
- (3.135) *Helmut's Vater kenne ich als äußerst bescheiden.*
- (3.136) *Kennst du New York schon?*

Satz (3.134) bedeutet soviel wie *Karl weiß Bescheid über die Prädikatenlogik*, während Satz (3.135) angibt, dass der Sprecher Helmut's Vater als äußerst bescheiden erlebt hat. Satz (3.136) schließlich stellt die Frage nach Erfahrungen mit oder Erlebnissen in New York.

Für die Computersemantik ist also auch eines der grundlegenden Probleme, wie die variierenden Wortbedeutungen in einer kompositionellen Semantikkonstruktion berücksichtigt bzw. in eine diskurssemantische Repräsentation integriert werden können.

Die obigen Beispiele zeigen, dass in den bisher dargestellten Formalismen Wortbedeutungen zwar als logische Prädikate wiedergegeben werden, also grundsätzlich extensional als Mengen definiert sind, diese extensionale Beschreibung in realistischen Szenarien aber nicht zu adäquaten Bedeutungsrepräsentationen führt.

Praktisch ist eine extensionale Beschreibung wohl nur in einfachen „Spielzeugszenarien“ durchführbar, so dass eine

andere Repräsentation von lexikalischem Wissen benötigt wird. Naheliegend wäre ein intensionaler Ansatz mit wörterbuchartigen Definitionen, also die Angabe hinreichender und notwendiger Bedingungen, wann ein Prädikat erfüllt ist. Aber auch ein solcher Ansatz beinhaltet zahlreiche praktische Probleme wie z. B. Abgrenzungsschwierigkeiten und insbesondere die unvermeidliche Zirkularität, Wortbedeutungen durch andere Wörter zu erklären (siehe hierzu insbesondere Wierzbicka 1996, S.274ff.).

Ein anderer Ansatz wäre, die Zirkularität als unvermeidlich zu akzeptieren und Wörter durch ihre Relationen zu anderen Wörtern zu definieren. Besonders interessant sind hierbei diejenigen Relationen, die Inferenzen erlauben, wie z. B. die Subsumptionsrelation oder die Teil-Ganzes-Relation, denn so kann ein Sprachverarbeitungssystem aus der Satzbedeutung Information extrahieren, ohne alle Details kennen oder umfangreiches Weltwissen besitzen zu müssen. Z. B. könnte ein entsprechendes System die Frage *Hat Paul einen Vogel gesehen?* beantworten, wenn es weiß, dass Paul einen Raben gesehen hat und dass *Rabe* einen Unterbegriff (ein sog. Hyponym) von *Vogel* bezeichnet. Auf diese Weise entstehen komplexe lexikalisch-semantische Netze, und die Bedeutung eines Wortes kann mit seiner Position in diesem Netz gleichgesetzt werden. Die Konzeption lexikalisch-semantischer Netze wird im Unterkapitel 4.3 ausführlich behandelt, so dass wir es in diesem Beitrag bei dieser kurzen Skizze belassen können.

Lexikalisch-semantische Netze enthalten in erster Linie Wissen über Konzepte, die untereinander durch unterschiedliche Relationen verknüpft sind. Wörter sind mit diesen Konzepten verlinkt, allerdings nicht immer eindeutig. So können z.B. die Wörter *Semmel* und *Brötchen* auf dasselbe Konzept verweisen (dies ist ein Beispiel für die sog. **Synonymie**), oder aber das ambige Wort *Maus* auf mehrere Konzepte (die sog. **Homonymie**). Dies bedeutet aber, dass wir in der formalen Semantik nicht einfach ein Wort wie *Maus* durch das entsprechende Prädikat `maus()` ersetzen sollten; je nach Kontext müssen wir verschiedene Prädikate `maus1()` und `maus2()` einsetzen (und evtl. weitere Prädikate `maus3()`, `maus4()`, ...), die den Konzepten entsprechen.

Ein wichtiger Schritt für eine vollständige semantische Analyse, die die logisch-kompositionelle Information mit der lexikalischen Semantik aus einem Wortnetz verknüpft, ist also eine **Lesartendisambiguierung** (engl. *word sense disambiguation*). Für die Lesartendisambiguierung existieren mehrere Verfahren, von denen wir vor allem das bekannteste Verfahren, den Lesk-Algorithmus (Lesk 1986), vorstellen werden.

Neben der klassischen Lesartendisambiguierung mittels des Lesk-Algorithmus stellen wir einen Ansatz vor, der mit Unterspezifikation arbeitet. Lexikalische Einheiten erhalten eine unterspezifizierte semantische Repräsentation, die allen Lesarten gemein ist. Zusätzlich zu dieser unterspezifizierten Repräsentation werden Regeln angegeben, die auf pragmatisches und/oder Weltwissen Bezug nehmen und mit deren Hilfe die im Kontext relevanten Lesarten festgelegt werden. Um die im Kontext relevante Lesart mit der Forderung nach Kompositionallität zu verbinden, werden Operationen angegeben, mit deren Hilfe die spezifizierten Bedeutungsrepräsentationen für eine kompositionelle Analyse passend umgeformt werden. Die Anpassung an das kombinatorische System der Semantikkonstruktion wird durch Verletzungen von Typen- und/oder Sortenanforderungen erzwungen. Das hier vorgestellte **Generative Lexikon** gehört in diese Klasse und arbeitet mit dieser Methode.

Zuvor sollen die verschiedenen Phänomene in der lexikalischen Semantik etwas ausführlicher dargestellt werden, als dies bisher geschehen ist.

Variabilität von Wortbedeutungen

Ein Blick auf die Struktur von Wörterbucheinträgen genügt, um sich klarzumachen, dass die Variabilität der Bedeutung lexikalischer Einheiten unterschiedlicher Art ist. In Wörterbüchern wird dies durch die Verwendung unterschiedlicher Schriften und Auflistungen angezeigt. Wie schwierig es ist, die verschiedenen Lesarten einer lexikalischen Einheit zu bestimmen, würde jedoch ein lexikonübergreifender Vergleich der Darstellung dieser Lesarten zeigen. Diese Angaben zu den Wortbedeutungen sind oftmals nicht konsistent.

Mit **Polysemie** wird die Eigenschaft von Wörtern bezeichnet, verschiedene Konzepte zu denotieren, die sich aber semantisch irgendwie aufeinander beziehen. Ein Wort ist also polysem, wenn es mehrere Bedeutungen besitzt, die in einem begrifflichen Zusammenhang stehen.

Beispiel 3.6.3

Klassische Beispiele für Polysemie liefern Nomen wie *Schule* oder *Zeitung* (vgl. Bierwisch 1983):

- (3.137) *Die Schule liegt an der Hauptstraße.* (Gebäude)
- (3.138) *Die Schule prägt die Kinder von frühen Jahren an.*
(Prinzip der Erziehung und Wissensvermittlung)
- (3.139) *Die Schule erteilte einen Verweis.* (Institution)
- (3.140) *Die Zeitung wurde im Jahre 1949 gegründet.* (Institution)
- (3.141) *Die Zeitung liegt auf dem Tisch.* (Objekt)
- (3.142) *Die Zeitung wird gerne gelesen.* (Informationsstruktur)

▪

Systematische Relationen zwischen den Lesarten polysemter Wortklassen werden u.a. in Apresjan (1973), Nunberg (1979), Bierwisch (1983) und Pustejovsky (1995) dargestellt. Z. B. sind Nomen wie *Tür* oder *Fenster* polysem, da sie einerseits auf das Objekt, andererseits aber auch auf den Durchgang oder die Öffnung referieren. Man vergleiche *Hans streicht die Tür rot* vs. *Hans betrat den Raum durch die Tür*. Solche systematischen Alternationen tauchen auch in anderen Bereichen auf wie z.B. bei der Tier/Nahrung-Alternation (*Der Fisch schwimmt schnell* vs. *Der Fisch schmeckt hervorragend*.).

Von der Polysemie ist die **Homonymie** abzugrenzen: Homonyme sind lexikalische Grundformen, die zwar dieselbe phonologische (**Homophone**) oder graphematische Form (**Homographe**) besitzen, deren verschiedene Bedeutungen aber in keinem semantischen Zusammenhang stehen.

Beispiel 3.6.4

Bank ist ein Homonym, dessen zwei Bedeutungen im Plural unterschieden werden können (*Bänke* vs. *Banken*). Die Bedeutung von *Bank* als Unternehmen für den Geldverkehr ist wiederum polysem, denn das Nomen kann sowohl ein Bankgebäude denotieren, als auch die Institution:

- (3.143) *Die Bank ist vollständig verlast.*
- (3.144) *Die Bank beschließt, die Gebühren zu erhöhen.*

Modern ist graphematisch ein Homonym (Adjektiv oder Verb), dessen Bedeutungen bei der Aussprache jedoch durch Vokallänge und Betonung unterschieden werden. □

Einen weiteren Bereich stellt die **Metonymie** dar. Metonymie liegt vor, wenn eine Verschiebung in der begrifflichen Interpretation vorgenommen wird. Im Gegensatz zur Polysemie, bei der im Lexikon einem Wort explizit oder implizit

mehrere Bedeutungsvarianten zugewiesen werden, handelt es sich bei der Metonymie um eine Verwendung in einer nicht-wörtlichen Bedeutung. Allerdings ist diese nicht-wörtliche Bedeutung auf bestimmte Weise mit der wörtlichen Bedeutung verbunden. Verschiebungen von der wörtlichen zu einer metonymischen Bedeutung finden häufig in systematischer Weise statt und treffen auf ganze Wortklassen zu, wobei jedoch auch subtile Ausnahmen innerhalb einer solchen Klasse bestehen. Grundlegende Relationen wie Teil-Ganzes, Verursacher-Effekt usw. stellen den Bezug zwischen den verschiedenen Bedeutungen her.

Beispiel 3.6.5

Beispiele für Metonymien geben die folgenden Sätze an:

- (3.145) *Hans hört gerne Wagner.*
- (3.146) *Die Firma rief an.*
- (3.147) *Berlin entschied, die Reform nachzubessern.*
- (3.148) *Das grüne Trikot hat die Sprintankunft gewonnen.*

In (3.145) referiert der Eigenname *Wagner* auf das musikalische Produkt (Produzent-Produkt-Relation). In (3.146) bezieht sich die NP *die Firma* auf eine Person (Institution-Angehörige-Relation), in (3.147) referiert der Eigenname *Berlin* auf eine Institution mit Entscheidungsgewalt (Ort-Institution-Relation), und in (3.148) referiert die NP *das grüne Trikot* auf den Träger dieses Kleidungsstücks. □

Einen weiteren Bereich, der aber nicht auf die lexikalische Semantik beschränkt ist, stellt die **Metapher** dar. Mit Metaphorik wird eine nicht-wörtliche Rede bezeichnet, bei der eine bestimmte Ähnlichkeit zwischen der Ausdrucks- und der intendierten Bedeutung bestehen muss.

Beispiel 3.6.6

Die kompositionelle Interpretation des Satzes

- (3.149) *Mein BMW frisst die Straße wie Spaghetti.*

liefert sicherlich nicht die intendierte Bedeutung. Die eigentliche Interpretation wird vielmehr durch den Vergleich mit den Spaghetti hergestellt: Das typische Aufsaugen langer Spaghetti beim Essen ähnelt der Fahrweise des BMW auf der Straße. □

Auf weitere lexikalische Phänomene wie die Relativität von Adjektivbedeutungen oder die kontextabhängige Semantik präzisierender oder depräzisierender Ausdrücke wie *eigentlich* oder *ungefähr* kann in diesem Abschnitt nicht eingegangen werden. Hierzu vergleiche man z.B. Lang et al. (1991) und Pinkal (1985).

Lesartendisambiguierung mit dem Lesk-Algorithmus

Das hier vorgestellte Verfahren zur Lesartendisambiguierung verwendet Wörter, die im Kontext des näher zu bestimmenden Wortes auftreten, um die korrekte Lesart zu ermitteln.

Im vorliegenden Unterabschnitt werden zuerst allgemeine Aspekte der Disambiguierung von Wortbedeutungen erörtert. Als Beispiel dient uns das mehrdeutige deutsche Substantiv *Maus*: es kann damit entweder ein Nagetier (*Maus₁*) oder die Computermaus (*Maus₂*) gemeint sein. Der üblichen Konvention folgend unterscheiden wir Lesarten durch Anhängen einer tiefgestellten Ziffer.

Zunächst muss festgelegt werden, wie viele Lesarten ein mehrdeutiges Wort besitzt und wie diese im Einzelnen definiert sind. Für unser Beispiel beschränken wir uns auf die klar abgegrenzten Bedeutungen *Maus₁* und *Maus₂* und legen Definitionen aus dem *Deutschen Universalwörterbuch* (Duden 2006) zugrunde:

(3.150)

Maus₁: kleines [graues] Nagetier mit spitzer Schnauze, das [als Schädling] in menschlichen Behausungen, auf Feldern u. in Wäldern lebt.

(3.151)

Maus₂: meist auf Rollen gleitendes, über ein Kabel mit einem PC verbundenes Gerät, das auf dem Tisch hin u. her bewegt wird, um den Cursor od. ein anderes Markierungssymbol auf dem Monitor des Computers zu steuern.

Weitere Lesarten wie z.B. die Verwendung als Kosenname (*Du süße Maus₃!*) sollen hier unberücksichtigt bleiben. Eine feinere Einteilung führt besonders bei Verben rasch zu Abgrenzungsproblemen, die sich in den völlig unterschiedlichen Lesarteninventaren verschiedener Wörterbücher sowie in Diskrepanzen bei der manuellen Annotierung von Lesarten widerspiegeln (siehe z.B. Véronis 1998). Ein häufig genanntes Beispiel ist das Substantiv *Bank*, das sowohl eine Sitzgelegenheit (*Bank₁*) als auch eine Kreditanstalt (*Bank₂*) bezeichnen kann. Während *Die deutsche Rechtschreibung* (Duden 2004) sich auf diese grobe Einteilung beschränkt, unterscheidet Duden (2006) bei der zweiten Lesart zwischen den Unterbedeutungen als Unternehmen (*Bank_{2a}*) und als Gebäude (*Bank_{2b}*). Bei dem Satz *Stefan geht heute zur Bank, um Geld abzuheben* ist aber nicht offensichtlich, ob hier *Bank_{2a}* oder *Bank_{2b}* gemeint ist.

Soll in der weiteren Verarbeitung auf Informationen aus einer semantischen Ressource wie z.B. den Wortnetzen WordNet (Fellbaum 1998) und GermaNet (Kunze und Wagner 1999) zurückgegriffen werden, so muss notwendigerweise das dort vorgegebene Lesarteninventar verwendet werden.

Die Hauptaufgabe der Lesartendisambiguierung besteht nun darin, jeder Verwendung eines mehrdeutigen **Zielwortes** den passenden Lesartenindex zuzuweisen. Es handelt sich also wie bei vielen computerlinguistischen Anwendungen um ein **Klassifikationsproblem**. Zur Illustration sind bei den folgenden Beispielsätzen für das Zielwort *Maus* die korrekten Lesarten durch einen tiefgestellten Index angegeben.

(3.152)

- a. Ein *Klick* mit der **Maus₂**, und der *Computer* zaubert ein Video auf den *Monitor*.
- b. **Mäuse₁** begeistern weltweit als Comic- und *Zeichentrickfiguren* ein riesiges Publikum, gleichzeitig werden sie als *Schädlinge* *gejagt*.
- c. Auch hier ersetzt das *Touchpad* die **Maus₂**.
- d. Da war eine **Maus₁**, die ein Kabel *angeknabbert* hat.
- e. Die **Maus₁** hat bei der Schulkameradin inzwischen ein neues Zuhause gefunden.
- f. Von da an spielte der VfB mit der Abwehr des FC Katz und **Maus**.
- g. Trost von der **Maus** – ihr vertrauen sich nicht nur Kinder gerne an.

In den meisten Fällen kann ein menschlicher Leser die korrekte Bedeutung bereits anhand des Satzkontextes erkennen. In (3.152a) – (3.152d) liefern insbesondere die kursiv gedruckten Wörter eindeutige Hinweise. Zum Teil handelt es sich dabei um semantisch oder thematisch verwandte Wörter (*Computer*, *Monitor*, *Schädlinge*, *Touchpad*), zum Teil um typische syntagmatische Wortverbindungen (*Klick*, *angeknabbert*). Bei (3.152e) könnte allerdings auch eine Computermaus gemeint sein; die korrekte Lesart *Maus₁* ergibt sich hier erst aus einem größeren Zusammenhang.

Die letzten zwei Beispiele stellen Sonderfälle dar, für die es keinen Sinn macht, dem Substantiv eine eigenständige Lesart zuzuweisen. Bei (3.152f) ist *Maus* Teil einer Redewendung (*mit jmdm. Katz und Maus spielen*). Auch wenn es sich hierbei um eine metaphorische Verwendung von *Maus₁* handelt, geht es in diesem Satz nicht um ein Nagetier; es wäre also irreführend, semantisches Wissen zu *Maus₁* aus einem Wortnetz abzurufen. In (3.152g) wird *Maus* als Eigename verwendet: es handelt sich um die Hauptfigur der *Sendung mit der Maus*. Derartige Sonderfälle sind in realen Texten keineswegs selten und müssen von automatischen Disambiguierungsverfahren berücksichtigt werden. In einer Stichprobe aus Zeitungsartikeln wurde beispielsweise in mehr als 20% aller Fälle das Wort *Maus* als Bestandteil einer Redewendung oder als Eigename verwendet.

Wir haben bereits festgestellt, dass sich die korrekte Lesart einer Instanz des Zielwortes oft aus anderen Wörtern ableiten lässt, die im selben Satz vorkommen. Die meisten automatischen Disambiguierungsverfahren stützen ihre Entscheidung daher auf solche **Kontextwörter**, wobei bisweilen auch größere oder kleinere Umgebungen als ein Satz verwendet werden. Meistens werden Funktionswörter und andere sehr unspezifische Wörter (z.B. *neu, riesig*) als **Stoppwörter** entfernt. Für deutsche Texte ist darüber hinaus eine **Lemmatisierung** der Kontextwörter üblich, während englische Verfahren oft mit den ursprünglichen Wortformen arbeiten. Auf diese Weise extrahiert man aus den Beispielsätzen (3.152a) – (3.152e) folgende Kontextwörter:

- (3.153) a. Computer₂, Klick, Monitor₂, Video, zaubern
 b. begeistern, Comicfigur, jagen, Publikum, Schädling₁, weltweit,
Zeichentrickfigur
 c. ersetzen, Touchpad
 d. anknabbern, Kabel₂
 e. finden, Schulkameradin, Zuhause

Eine wesentliche Aufgabe der automatischen Disambiguierung besteht nun darin herauszufinden, welche dieser Kontextwörter gute **Indikatorwörter** für eine der betrachteten Lesarten des Zielwortes sind. Da ohnehin ein Wörterbuch oder Wortnetz zur Festlegung des Lesarteninventars benötigt wird, liegt es nahe, solche Indikatorwörter automatisch aus den dort aufgeführten Bedeutungsdefinitionen zu gewinnen. Auf dieser Idee basiert eines der ältesten und bekanntesten Disambiguierungsverfahren, der sogenannte **Lesk-Algorithmus** (Lesk 1986). Aus den Definitionen (3.150) and (3.151) ergeben sich nach Lemmatisierung und Entfernen von Stopwörtern folgende Indikatorwörter:

- (3.154)
- a. **Maus**₁: Behausung, Feld, grau, klein, leben, menschlich, Nagetier, Schädling, Schnauze, spitz, Wald
 - b. **Maus**₂: bewegen, Computer, Cursor, Gerät, gleiten, Kabel, Markierungssymbol, Monitor, PC, Rolle, steuern, Tisch, verbinden

In (3.153) sind diese Indikatorwörter durch Unterstrichen markiert und mit der jeweiligen Lesart versehen. Mit Hilfe dieser Informationen können die Beispielsätze (3.152a) und (3.152b) korrekt den Lesarten *Maus*₂ bzw. *Maus*₁ zugeordnet werden. An den übrigen Beispielen zeigt sich jedoch ein erheblicher Nachteil dieser einfachsten Variante des Lesk-Algorithmus. Da aus den Wörterbuchdefinitionen nur wenige Indikatorwörter gewonnen werden können, liegen für viele Sätze keine verwertbaren Hinweise vor (vgl. 3.153c und 3.153e). Bisweilen kommt es auch zu irreführenden „Zufallstreifern“, z.B. *Kabel* in (3.153d), wodurch der Lesk-Algorithmus die falsche Lesart *Maus*₂ zuweisen würde. Ein weiteres Problem liegt in der unterschiedlichen Länge der Bedeutungsdefinitionen. Kann für eine Lesart eine wesentlich größere Anzahl von Indikatorwörtern bestimmt werden, so erhöht sich die Wahrscheinlichkeit von Zufallstreifern für diese Lesart entsprechend.

Aus diesem Grund ergänzt Lesk (1986) die Liste der Kontextwörter mit zusätzlichen Einträgen, die analog zu den Indikatorwörtern aus Bedeutungsdefinitionen der Kontextwörter gewonnen werden. Wir veranschaulichen dieses Vorgehen am Beispiel (3.153c). In Duden (2006) finden wir folgende Definitionen:

- (3.155)
- ersetzen**₁: für jmdn./etw. Ersatz schaffen; jmdn./etw. an die Stelle von jmdm./etw. setzen; für jmdn./etw. ein Ersatz sein; an die Stelle von jmdm./etw. treten
 - ersetzen**₂: erstatten, wiedergeben, für etw. Ersatz leisten

- (3.156) ⁴ **Touchpad:** *auf Fingerdruck reagierendes, im Computer integriertes Zeigegerät zur Steuerung des Cursors anstelle einer Maus*

Ist ein Kontextwort wie bei (3.155) selbst mehrdeutig, so wird die „passende“ Bedeutungsdefinition gewählt, die zur größten Überlappung mit den Indikatorwörtern der jeweiligen Lesart des Zielwertes führt. In unserem Beispiel wählen wir die Bedeutung *ersetzen₁* und erhalten so für (3.152c) die **erweiterten Kontextwörter**

- (3.157) c. *Computer₂, Cursor₂, Ersatz, ersetzen, Fingerdruck, integriert, Maus, reagieren, schaffen, setzen, Stelle, Steuerung, Touchpad, treten, Zeigegerät*

Durch Vergleich mit (3.154a) und (3.154b) kann nun die Überlappung mit Indikatorwörtern der beiden Lesarten von *Maus* bestimmt werden. Diese sind in (3.157c) durch Unterstrichen markiert. Um die dabei zu erwartenden Zufallstreffer angemessen zu berücksichtigen, wird anstelle der reinen Anzahl von Übereinstimmungen der **Dice-Koeffizient** (3.158) als Entscheidungsgrundlage benutzt.

$$\text{Dice} = \frac{2 \cdot |\text{Indikatorwörter} \cap \text{erweiterte Kontextwörter}|}{|\text{Indikatorwörter}| + |\text{erweiterte Kontextwörter}|} \quad (3.158)$$

Im Zähler von (3.158) steht dabei die Anzahl von Übereinstimmungen, im Nenner die Gesamtanzahl der Indikatorwörter und erweiterten Kontextwörter. In unserem Beispiel ergeben sich für *Maus₁* keine Übereinstimmungen, also $\text{Dice}(1) = 0$. Für *Maus₂* finden wir in (3.157c) zwei Übereinstimmungen und erhalten damit

$$\text{Dice}(2) = \frac{2 \cdot 2}{13 + 15} = \frac{4}{28} = \frac{1}{7}.$$

Wegen $\text{Dice}(2) > \text{Dice}(1)$ entscheidet sich der Lesk-Algorithmus also für die korrekte Lesart *Maus₂*.

Weitere Verbesserungen des Lesk-Algorithmus können durch die Verwendung von im Wörterbuch aufgeführten Beispielsätzen zusätzlich zu den Bedeutungsdefinitionen erzielt werden, sowie durch die Gewichtung von Indikator- und Kontextwörtern mit aus Korpora ermittelten Häufigkeitsinformationen (Kilgarriff und Rosenzweig 2000). Andere Erweiterungen werden von Vasilescu et al. (2004) beschrieben und verglichen.

Bei Einsatz eines Wortnetzes können darüber hinaus semantische Ähnlichkeiten zwischen den Kontextwörtern und einer Lesart des Zielwertes berechnet werden. Üblicherweise wird hierzu der kürzeste Pfad im Wortnetz bestimmt, der von der betrachteten Lesart zu einem Kontextwort führt, wobei die einzelnen Kanten oft mit einem statistischen Maß gewichtet sind (Resnik 1995; Lin 1998). Ist ein Kontextwort selbst mehrdeutig, so wird entweder dessen „ähnlichste“ Lesart herangezogen oder über alle Lesarten gemittelt. Auf diese Weise kann ein

⁴Diese Bedeutungsdefinition wurde aus didaktischen Gründen leicht erweitert.

wesentlich größerer Teil der Kontextwörter zur Lesartendisambiguierung genutzt werden als beim Lesk-Algorithmus.

Neuere Arbeiten betrachten die Lesartendisambiguierung meist als ein allgemeines Klassifikationsproblem und wenden verschiedene **maschinelle Lernverfahren** wie Naive Bayes, Entscheidungsbäume oder Support Vector Machines an (Mitchell 1997; Bishop 2006). Hierzu müssen von Hand disambiguierte Beispielsätze als Trainingsdaten vorliegen. Ein Vorteil dieses Ansatzes ist, dass nahezu beliebige Kontextmerkmale für die Disambiguierung genutzt werden können. Beispielsweise kommt *Maus*₂ selten im Plural vor, so dass in (3.152b) vermutlich die erste Lesart vorliegt. Auch bestimmte syntaktische Konstruktionen und die Argumentstruktur von Verben können nützliche Hinweise liefern: ist wie in (3.152d) *Maus* das Subjekt eines Verbs, das ein Lebewesen als Agens erwartet, so handelt es sich ebenfalls mit hoher Wahrscheinlichkeit um die Lesart *Maus*₁. Darüber hinaus können die maschinellen Lernverfahren unterschiedliche Häufigkeiten der Lesarten berücksichtigen. Ist bekannt, dass sich mehr als die Hälfte aller Verwendungen von *Maus* auf das Nagetier beziehen, so bietet es sich an, in Zweifelsfällen die Lesart *Maus*₁ zuzuweisen.

Ein erheblicher Nachteil maschineller Lernverfahren ist, dass für jedes zu disambiguierende Wort eigene Trainingsdaten benötigt werden. Das automatische Verfahren lernt aus diesen manuell annotierten Sätzen Entscheidungskriterien, die an das Zielwort und sein Lesarteninventar angepasst sind. Es ist aber nicht in der Lage, die gelernten Regeln auf andere Wörter zu übertragen, selbst wenn diese eine ähnliche Bedeutung haben. Hat das Disambiguierungsverfahren z. B. gelernt, zwischen den verwandten Lesarten von *Bank* als Unternehmen und als Gebäude zu unterscheiden, so kann es trotzdem nicht ohne weiteres die analoge Unterscheidung für Wörter wie *Polizei* und *Schule* treffen (als Institution vs. Gebäude).

Ein vielversprechender Ansatz, dieses Problem zu mildern ist das sogenannte **Bootstrapping**. Dabei wird für jedes Zielwort nur eine kleine Menge manuell annotierter Trainingsdaten benötigt. Das maschinelle Lernverfahren gewinnt aus ihnen zunächst wenige, noch unsichere Entscheidungskriterien, die auf eine große Anzahl unannotierter Beispielsätze angewendet werden. Alle Sätze, für die eine klare Entscheidung getroffen werden kann, werden mit ihren automatisch zugewiesenen Lesarten zu den Trainingsdaten hinzugefügt. In mehreren Iterationen entsteht so eine zunehmend große Trainingsmenge und es werden immer mehr und bessere Entscheidungskriterien ermittelt. Eine bekannte Implementierung des Bootstrapping-Ansatzes ist der **Yarowsky-Algorithmus** (Yarowsky 1995). Er verwendet ausschließlich Kollokationen des Zielwertes als Kontextmerkmale und benötigt als Ausgangspunkt für das Training lediglich eine einzige, sinnvoll gewählte Kollokation für jede Lesart. Wesentlich für den Erfolg dieses Verfahrens sind zwei Heuristiken, die auch in viele neuere Arbeiten Einzug gefunden haben: *one sense per collocation* (d.h. Kollokationen sind lesartenspezifisch) und *one sense per discourse* (d.h. innerhalb eines zusammenhängenden Textes wird im Allgemeinen nur eine Lesart eines mehrdeutigen Wortes verwendet).

Die Evaluation statistischer Verfahren zur Lesartendisambiguierung verwendet die üblichen Methoden und Gütemaße für Klassifikationsprobleme, die in Unter-

kapitel 2.4, Abschnitt 2.4.3 ausführlich beschrieben werden. Insbesondere werden Precision und Recall für jede Lesart des Zielwertes berechnet. Ein direkter Vergleich zwischen aktuellen Disambiguierungsverfahren findet bei dem seit 1998 regelmäßig veranstalteten SENSEVAL-Wettbewerb statt, seit 2007 unter dem Namen SEMEVAL (Kilgarriff und Palmer 2000; SENSEVAL 2009). Als „Baseline“ für solche Evaluationen dient meist eine Variante des Lesk-Algorithmus. Interessant sind nur Verfahren, die eine wesentliche, statistisch signifikante Verbesserung gegenüber dieser Baseline erreichen. Beim ersten SENSEVAL-Wettbewerb konnte die korpusbasierte Erweiterung des Lesk-Algorithmus allerdings nur von wenigen Verfahren übertrffen werden (Kilgarriff und Rosenzweig 2000).

Das „Generative Lexikon“

In einer Reihe von Veröffentlichungen hat James Pustejovsky ein populär gewordenes Verfahren zur Bestimmung von Wortbedeutungen vorgestellt, das unter dem Stichwort „Generatives Lexikon“ bekannt ist (Pustejovsky 1991, Pustejovsky 1995). Das Generative Lexikon hat mit den bereits vorgestellten Verfahren zur Lesartendisambiguierung nichts gemein. Die hier formulierte Methode zur Bestimmung von Wortbedeutungen kann vielmehr als Beispiel für die Klasse von Verfahren gelten, die mit unterspezifizierten lexikalischen Repräsentationen arbeiten und Typverletzungen als Anstoß für entsprechende Uminterpretationen vornehmen.

In der ausführlichsten Darstellung des Ansatzes (Pustejovsky 1995) werden vier Interpretationsebenen eingeführt, die bei der Semantikkonstruktion interagieren:

Die Argumentstruktur: Sie gibt die Anzahl der Argumente einer lexikalischen Einheit sowie den logischen Typ der Argumente an. Für die Anbindung der Argumentstruktur an die Ebenen der Qualia- und Ereignisstrukturen (siehe unten) unterscheidet Pustejovsky zwischen echten Argumenten, Default-Argumenten, impliziten Argumenten und Adjunkten. Während echte Argumente syntaktisch realisiert werden müssen, sind Default-Argumente Parameter, die sich auf die Qualia-Strukturen beziehen, syntaktisch aber nicht notwendigerweise realisiert werden müssen. Ein Beispiel ist:

(3.159) *Peter baut das Wohnhaus aus Steinen.*

Die Präpositionalphrase *aus Steinen* ist ein Default-Argument, denn Wohnhäuser werden typischerweise aus Steinen gebaut.

Implizite Argumente hingegen sollten sprachlich nur dann ausgedrückt werden, wenn es sich bei ihnen um Subtypen oder spezielle Anforderungen aus dem Diskurs handelt. Man vergleiche:

(3.160) *?Peter trat seinen Bruder mit seinem Bein.*

(3.161) *Peter trat seinen Bruder mit seinem steifen Bein.*

Die Semantik von *treten* impliziert, dass ein Bein hierfür verwendet wird. Steife Beine stellen jedoch einen Subtypen des impliziten Arguments von *treten* dar.

Als Adjunkte werden schließlich alle anderen syntaktischen Arten der Modifikation bezeichnet.

Argumentstrukturen werden als Merkmalsstrukturen (siehe hierzu Unterkapitel 2.3) repräsentiert. So erhält z.B. *treten* die folgende (vereinfachte) Argumentstruktur. Die Werte der Merkmale sind semantische Typen und werden hier fett dargestellt.

treten	
ARGSTR	
	ARG1 belebtes _ individuum
	ARG2 physikalisches _ objekt
	S-ARG1 bein

Die Ereignisstruktur: Dies ist eine Definition der Ereignissorten, die eine lexikalische Einheit repräsentiert. Ereignissorten basieren auf einer Sortenhierarchie für Zustände, Prozesse und Übergänge. Auch Ereignisstrukturen lassen sich als Merkmalsstrukturen darstellen, bei denen die jeweiligen Ereignisse, deren Ereignistypen, eine Ordnung über diesen Ereignissen, sowie das relevanteste Teileereignis in der Ereignisstruktur bzgl. der Bedeutung des Verbs angegeben werden. So kann für das Verb *bauen* folgende Ereignisstruktur in Form einer Merkmalsstruktur angegeben werden:

bauen	
EREIGNISSTR	
	E1 prozess
	E2 zustand
	RESTR $<_\alpha$
	KOPF E1

Etwas zu bauen, bedeutet demnach, dass ein Prozess stattfindet, der in einem speziellen Zustand endet. Diese beiden Ereignisse sind durch die Relation $<_\alpha$ geordnet, die im Wesentlichen besagt, dass E1 dem E2 vorangeht und beide zusammen Teil des *bauen*-Ereignisses sind.

Die Qualiastruktur: Dies ist eine Repräsentation, die mittels vierer Rollen wesentliche Aspekte der jeweiligen Wortbedeutung spezifiziert. Diese Rollen werden als FORMAL, KONSTITUTIV, TELISCH und AGENTIV bezeichnet. Die Werte der Rolle FORMAL geben Eigenschaften an, die das denotierte Objekt von anderen Objekten unterscheidet. Unter der Rolle KONSTITUTIV wird die Relation zwischen einem Objekt und seinen konstituierenden Teilen angegeben. Die Werte der Rolle TELISCH geben den Zweck und die Funktion der denotierten Entität an, und die Werte der Rolle AGENTIV geben Faktoren an, die in der Entstehung der Entität involviert sind.

Übrigens existiert eine enge Beziehung zwischen Wissensrepräsentationsformalismen (vgl. Unterkapitel 4.6) und Qualiastrukturen. So entspricht die FORMAL-Rolle der **is_a**-Relation in Wissensrepräsentationssprachen. Die KONSTITUTIV-Rolle entspricht der **part_of**-Relation, die TELISCH-Rolle einer **purpose**-Relation und die AGENTIV-Rolle der **cause**-Relation. In dieses Bild passen auch Erweiterungen der Qualiastruktur, bei denen nicht nur vier Rollen angegeben werden, sondern jede Art von Wissen angegeben werden kann. Dann besitzen die Qualiastrukturen eine starke Ähnlichkeit mit semantischen Netzen. Solche Erweiterungen werden im nächsten Abschnitt bei der Angabe lexikalischer Regeln verwendet.

Die Werte der vier Rollen sind Defaults, d.h. sie geben typisches Wissen bzgl. dieser Rollen an. So kann für das Nomen *Tür* folgende Qualiastruktur angegeben werden:

$\begin{bmatrix} \text{tür} \\ \dots \\ \text{QUALIA} \end{bmatrix}$	$\begin{bmatrix} \text{FORMAL} & \text{phys_obj}(X) \\ \text{KONSTITUTIV} & \text{öffnung}(X) \\ \text{TELISCH} & \text{hindurchgehen}(T, Z, Y) \\ \text{AGENTIV} & \text{artefakt}(X) \end{bmatrix}$
--	--

Die Alternation zwischen der Bedeutung als physikalisches Objekt und der Bedeutung als Öffnung wird mittels der KONSTITUTIV-Rolle bzw. der FORMAL-Rolle angegeben. Auch die Lexikoneinträge von Verben und Adjektiven werden mit Qualiastrukturen ausgestattet, um deren Polysemie zu repräsentieren.

Insbesondere die Werte der FORMAL-Rolle sind typisiert. Diese Typen sind einfach oder komplex. Im obigen Beispiel wurde der einfache Typ **phys_obj** verwendet. Komplexe Typen werden mittels des **Typkonstruktors** \bullet gebildet und daher auch „gepunktete Typen“ genannt. Ein Beispiel für einen komplexen Typen wäre **information** \bullet **phys_obj**. Ein komplexer Typ **a** \bullet **b** denotiert Mengen von Objektpaaren $\langle a', b' \rangle$, wobei a' vom Typ **a** ist und b' vom Typ **b**. Eine Bedingung $a' \mathbf{R} b'$ schränkt diese Paarmenge auf diejenigen Objekte ein, für die eine Relation **R** gilt. **R** selber ist eine Relation zwischen Objekten vom Typ **a** und Objekten vom Typ **b**.

Definition 3.6.6

Die Semantik des Operators \bullet lautet:

$$\llbracket \mathbf{a} \bullet \mathbf{b} \rrbracket = \{ \langle a', b' \rangle \mid a' \in \mathbf{a} \text{ und } b' \in \mathbf{b} \text{ und } a' \mathbf{R} b' \text{ und } \langle a', b' \rangle \text{ ist ein komplexes Objekt} \}$$

□

Gepunktete Typen werden benötigt, um komplexe Objekte als n -Tupel in einen Diskurs einzuführen, auf deren Elemente im Diskurs jedoch separat anaphorisch Bezug genommen werden kann. Ein Beispiel liefert das Objektpaar, das durch den komplexen Typen **information** \bullet **phys_obj** eingeführt wird. Dieses

Paar wird im Diskurs als die komplexen Objekte $\langle \mathbf{x:information}, \mathbf{y:phys_obj} \rangle$ adressiert. Dieser komplexe Typ ist z. B. für die Polysemie lexikalischer Einheiten wie *Buch* einschlägig.

Beispiel 3.6.7

In dem folgenden Satz wird mittels des Pronomens *es* auf den Referenten für das Buchobjekt verwiesen, der vorher aber nicht explizit eingeführt wurde. Vielmehr führte das Nomen *Buch* die Lesart als Buch-Information in den Satz ein. Dass dennoch Referenz auf das Buchobjekt möglich ist, wird durch den obigen komplexen Typen gewährleistet.

- (3.162) *Nachdem er das Buch gelesen hatte (Information), legte er es ins Regal zurück. (Objekt)*

□

Aber nicht immer werden sämtliche Lesarten eines polysemen Nomens in einen Diskurs eingeführt. Nomen wie *Tür*, *Fenster* usw. referieren systematisch auf ein konkretes Objekt sowie auf eine Öffnung. In einem Diskurs kann nicht simultan auf beide Lesarten Bezug genommen werden, wie das folgende Beispiel zeigt:

- (3.163) ??*Hans ging durch die Tür, die er gestern lackiert hatte.*

Dies bedeutet, dass es polyseme Einheiten gibt, die nicht sämtliche Bedeutungsvarianten in einem Diskurs verfügbar machen. Um diese Unterschiede in der Polysemie zu berücksichtigen, führt Buitelaar (1998) einen zusätzlichen Typkonstruktor \circ ein, der zwar analog zum \bullet -Operator semantische Typen konstruiert, diese konstruierten Typen aber – im Gegensatz zum \bullet -Operator – nicht komplexe Objekte denotieren:

Definition 3.6.7

Die Semantik des Operators \circ lautet:

$$[\![\mathbf{a} \circ \mathbf{b}]\!] = \{ \langle a', b' \rangle \mid a' \in \mathbf{a} \text{ und } b' \in \mathbf{b} \text{ und } a' \mathbf{R} b' \}$$

□

Kombinationen in der Anwendung der Typkonstruktoren sind möglich. Beispielsweise kann dem Nomen *Zeitung* der Typ $(\mathbf{information} \bullet \mathbf{phys_obj}) \circ \mathbf{organisation}$ zugeordnet werden. Dieser semantische Typ gibt an, dass das polyseme Nomen *Zeitung* drei Interpretationen besitzt, von denen die Interpretation als Organisation nicht simultan mit den beiden anderen in einen Diskurs eingeführt wird. Dies zeigen die folgenden Sätze (3.164) und (3.165).

- (3.164) *Die interessante Zeitung (Information) liegt auf dem Tisch.
(Objekt)*

- (3.165) ??*Die interessante Zeitung (Information), die den Reporter feuerte (Organisation), liegt auf dem Tisch. (Objekt)*

Neben den Ebenen der Argumentstruktur, der Ereignisstruktur und der Qualiastruktur bildet die vierte Ebene schließlich die **lexikalische Vererbung**: Das Lexikon ist global als Verband strukturiert, so dass mittels Vererbung lexikalische Objekte Eigenschaften von übergeordneten Objekten erhalten können (siehe Unterkapitel 2.3 zu den Begriffen des Verbandes und der Vererbungshierarchie).

Diese vier Ebenen werden durch die folgenden zwei Operationen in Beziehung zueinander gesetzt:

1. **Typerzwingung**: Eine lexikalische Einheit oder Phrase wird durch eine regierende lexikalische Einheit in eine spezifische semantische Bedeutung umgeformt, ohne dass sich der syntaktische Typ dieser Einheit ändert. Dies ist ein Prozess, der eine Bedeutung erzeugt, die in der Qualiastruktur repräsentiert ist.
2. **Selektive Bindung**: Eine lexikalische Einheit oder eine Phrase operiert über eine (Teilstruktur einer) Phrase, ohne den semantischen Typ zu ändern.

Die Anwendung dieser Operationen wird durch den syntaktischen und semantischen Kontext der entsprechenden Phrase kontrolliert.

Kompositionnalität wird im logischen Paradigma mittels Funktoren realisiert, die typisierte Argumente nehmen. Dabei kann der Funktor nur dann erfolgreich auf die Argumente angewandt werden, wenn diese Argumente den Typ besitzen, den der Funktor auch verlangt. Flexibilität bzgl. der Typisierung wird durch regelbasierte Typanhebung oder Typverschiebung erlangt. Solche Flexibilisierungen der Typzuweisung werden z. B. in Partee (1992) vorgestellt. Die **Typerzwingung** ist eine Art **Typverschiebung** aus Sicht des Funktors; er zwingt den Argumenten einen bestimmten Typ auf. Die Menge der Verschiebungsooperatoren, die mit einem Ausdruck α verbunden ist, wird Σ_α genannt. Die Regel lautet (Pustejovsky 1995, S. 111):

Wenn α vom Typ c ist, und β vom Typ $\langle a, b \rangle$, dann gilt:

- (i) Ist $c = a$, so ist $\beta(\alpha)$ von Typ b , sonst
- (ii) wenn es ein $\sigma \in \Sigma_\alpha$ gibt, so dass $\sigma(\alpha)$ in einem Ausdruck von Typ a resultiert, dann ist $\beta(\sigma(\alpha))$ von Typ b , sonst
- (iii) produziere einen Typfehler.

Beispiel 3.6.8

Ein Beispiel für Typerzwingung liefert das Verb *wollen*. In den Sätzen (3.166) – (3.168) besitzt das Verb *wollen* unterschiedliche Argumente in Objektposition, nämlich einen Satz vom Typ t , ein intransitives Verb vom Typ $\langle e, t \rangle$, und eine NP vom Typ $\langle \langle e, t \rangle, t \rangle$.

(3.166) *Helmut will, dass Alfons geht.*

(3.167) *Helmut will gehen.*

(3.168) *Helmut will ein Bier.*

□

Die Idee ist nun, dass sämtliche Argumente Aussagen, und somit vom Typ t , sein sollen. Die Verschiebung der eigentlichen Typen zum Typ t geschieht dann unter Rückgriff auf implizite Information. So soll Satz (3.167) als Satz (3.169) interpretiert werden und Satz (3.168) wie in (3.170):

(3.169) $\text{Helmut}_e \text{ will}_{\langle t, \langle e, t \rangle \rangle} [\text{Helmut}_e \text{ [gehen]}_{\langle e, t \rangle}]_t$

(3.170) $\text{Helmut}_e \text{ will}_{\langle t, \langle e, t \rangle \rangle} [\text{Helmut}_e [[\text{trinken}]_{\langle \langle e, t \rangle, \langle e, t \rangle \rangle} \text{ [ein} \\ \text{Bier]}_{\langle \langle e, t \rangle, t \rangle}]_{\langle e, t \rangle}]_t$

Die Verschiebung in (3.170) vom semantischen Typ für NPs hin zum Typ t geschieht unter Rückgriff auf die Qualiastruktur von *Bier*. Die TELISCH-Rolle des Nomens stellt das Ereignis des Trinkens für die Interpretation zur Verfügung. Diese Information wird für die Typverschiebung verwendet.

Die **selektive Bindung** ist insbesondere für die Polysemie von Adjektiven relevant. Adjektive besitzen die Eigenschaft, unterschiedliche Aspekte der Nomen zu modifizieren, wie die folgenden Beispiele zeigen:

(3.171) *ein schnelles Auto*

(3.172) *eine schnelle Zeitung*

(3.173) *ein schneller Autor*

(3.174) *eine schnelle Verkäuferin*

Phrase (3.171) besitzt zwei Bedeutungen: Ein schnelles Auto ist eines, das schnell fahren kann (aufgrund seiner Motorenstärke) oder eines, das momentan schnell fährt. Die Phrasen (3.172) – (3.174) besitzen eine Default-Interpretation, die Bezug auf inhärente Eigenschaften nimmt (publizieren, schreiben, bedienen), aber im entsprechenden Kontext kann auch eine andere Bedeutung relevant sein. So kann z.B. bei einem Marathon einer Betriebsmannschaft die Siegerin auch als *schnelle Verkäuferin* bezeichnet werden, wenn der Sprecher annimmt, dass der Hörer weiß, dass sie beruflich Verkäuferin ist. Dann bezieht sich *schnell* auf die momentane Eigenschaft des Laufens. Okkasionelle Eigenschaftszuweisungen dieser Art können mit Qualiastrukturen nicht erfasst werden, denn Qualias geben – wie bereits erwähnt – Defaultwissen an.

Die Idee ist nun, einem Adjektiv wie *schnell* zu erlauben, nur auf spezielle Information in der Qualiastruktur zuzugreifen. So kann *schnell* nur auf Information der telischen Rolle zugreifen, da es ein Ereignis modifiziert, das dort angegeben ist, während das Adjektiv *teuer* sich auf die FORMAL-Rolle in der Qualiastruktur bezieht.

Zusammenfassend besteht die Idee hinter dem „generativen Lexikon“ darin, prototypische Information mit hoher Salienz im Lexikon anzugeben und über verschiedene Mechanismen diese Information für die Satzkonstruktion verfügbar zu machen. Gesteuert wird der gesamte Mechanismus über die Kombinierbarkeit semantischer Typen bzw. Typanhebungs- und Verschiebungsoperationen.

3.6.6 Literaturhinweise

Die Montague-Semantik wird in Dowty et al. (1981) detailliert dargestellt. Das Standard-Einführungsbuch in die DRT ist Kamp und Reyle (1993). Beide Bücher sind allerdings nicht für Computerlinguisten geschrieben worden, d.h. Methoden der computerlinguistischen Umsetzung dieser formalsemantischen Ansätze werden dort nicht erläutert. Blackburn und Bos (2005) stellt anhand der Programmiersprache Prolog verschiedene Methoden vor, Konzepte der formalen Semantik für die Computerlinguistik nutzbar zu machen; van Deemter und Peters (1996) ist eine Sammlung mit Forschungsartikeln zur Unterspezifikation.

Einen guten Einstieg in grundlegende Phänomene in der lexikalischen Semantik liefern Cruse (1986) und Saeed (2008). Eine kompakte, gut lesbare Darstellung verschiedener Varianten des Lesk-Algorithmus und anderer Verfahren zur Lesartendisambiguierung findet sich bei Jurafsky und Martin (2009, S. 638–651). Im Anschluss werden dort überblicksartig die wichtigsten semantischen Ähnlichkeitsmaße besprochen (Jurafsky und Martin 2009, S. 652–667). Eine gute Quelle für Veröffentlichungen zu aktuellen Disambiguierungsverfahren und deren Evaluationsergebnisse stellen die SENSEVAL- bzw. SEMEVAL-Wettbewerbe dar (SENSEVAL 2009). Allgemeine Fragen der lexikalischen Semantik werden aus computerlinguistischer Sicht von Jurafsky und Martin (2009, S. 611–635) diskutiert. Die Beiträge in Pustejovsky und Bergler (1992) sowie in Bouillon und Busa (2001) zeigen unterschiedliche Methoden, Weltwissen mit lexikalisch-semantischem Wissen zu kombinieren, um Äußerungsbedeutungen der lexikalischen Einheiten zu bestimmen. Für einen Überblick über Arbeiten, die den Erwerb lexikalischer Bedeutungen thematisieren, sei auf Matsumoto (2003) verwiesen.

3.7 Pragmatik

Die Computersemantik setzt – wie im vorigen Unterkapitel 3.6 dargestellt – die aus den linguistischen, formalen Semantiktheorien bekannten Konzepte ein, um systematisch eine Bedeutungsrepräsentation für sprachliche Ausdrücke zu konstruieren. Allerdings blendet die Computersemantik die meisten kontextuellen Parameter bei der Konstruktion der Bedeutungsrepräsentation aus.

Ein Beispiel soll dies verdeutlichen. Angenommen, Sie rufen bei Ihrer Bank an, weil Sie Ihre Aktien der Firma *NoProfits* verkaufen wollen und Sie geraten an ein telefonisches Auskunftssystem. Sie fragen: *Kann ich meine NoProfits-Aktien verkaufen?* und ein gutes Auskunftssystem würde dann z. B. die Antwort zurückgeben: *Die NoProfits-Aktien können Sie für 30 Cent pro Aktie verkaufen.* Für diese Antwort muss das Auskunftssystem die Entscheidungsfrage als eine Aufforderung interpretieren, die Aktien zu einem möglichst guten Kurs zu verkaufen. Die Semantik kann die Überführung der Entscheidungsfrage in diese Aufforderung aber nicht leisten, da diese Frage in einem anderen Kontext durchaus wörtlich zu interpretieren ist. Zudem bleibt die Semantik von *können* unterspezifiziert: drückt *können* eine Fähigkeit aus, so dass die obige Frage zu interpretieren ist als: hat der Sprecher die Fähigkeit, die *NoProfits*-Aktien zu verkaufen? Oder ist gemeint, dass die Möglichkeit besteht, dass der vom Satz ausgedrückte Sachverhalt des Aktien-Verkaufs eintritt?

Die Pragmatik beschäftigt sich mit den unterschiedlichen kontextuellen Einflüssen auf die Interpretation. Die Eingangsfrage an das Auskunftssystem muss z. B. mittels pragmatischer Interpretationsprinzipien als eine Aufforderung interpretiert werden. Die Frage selbst wird auf der Basis von Information ausgewertet, die stillschweigend vorausgesetzt wird. So drückt das Pronomen *meine* aus, dass der Anrufer in einer Besitzens-Relation zu den Aktien steht, die in bestimmten Kontexten noch weiter spezifiziert werden müsste (Aktien, die der Anrufer erworben/geerbt/gestohlen/... hat), und das Verb *verkaufen* teilt unter anderem ebenfalls mit, dass der Anrufer die Aktien besitzt. Diese stillschweigend vorausgesetzte Information wird als Präsupposition bezeichnet und muss ebenfalls im Diskurskontext mittels pragmatischer Regeln verarbeitet werden (mehr hierzu in Abschnitt 3.7.3). Neben den Präsuppositionen gelten als Kernbereiche der Pragmatik die Deixis, Implikaturen, Sprechakte und Konversationsstrukturen. Levinson (2000) gibt einen guten Überblick über diese und weitere pragmatische Gebiete.

Pragmatische Phänomene werden in der Computerlinguistik schon seit längerer Zeit untersucht, obwohl sie bislang nicht den Stellenwert erlangten wie z. B. die Syntax oder die Semantik. Von den vielfältigen computerlinguistischen Methoden zur Behandlung pragmatischer Phänomene werden in diesem Unterkapitel aus Platzgründen drei vorgestellt. Der erste Abschnitt 3.7.1 dient dabei vorab der Klärung der grundlegenden Begriffe *Text*, *Diskurs* und *Dialog* und der Darstellung, wie größere Äußerungseinheiten mit Hilfe rhetorischer Relationen strukturiert werden können.

Der zweite Abschnitt 3.7.2 stellt unterschiedliche Methoden zur automatischen Bestimmung des Antezedens einer Anapher vor. Anaphern sind sprach-

liche Ausdrücke, deren Bedeutung erst durch ihre Referenz auf eine vorherige Äußerungseinheit, das Antezedens, festgelegt werden kann. Dabei tritt in der Regel das Problem auf, dass mehrere Einheiten als potentielle Antezedenzen existieren und das korrekte Antezedens bestimmt werden muss. Pronomina sind prototypische Anaphern, aber es existieren noch weitere Klassen anaphorischer Ausdrücke.

Der dritte Abschnitt 3.7.3 beschäftigt sich mit den Präsuppositionen sowie mit Implikaturen. Präsuppositionen sind – wie erwähnt – Voraussetzungen, die mit einer sprachlichen Äußerung gesetzt werden. Da Präsuppositionen sowohl einen semantischen als auch einen pragmatischen Bedeutungsanteil besitzen, stellt der Abschnitt über Präsuppositionsverarbeitung das Bindeglied zwischen dem Unterkapitel zur Semantik und diesem Pragmatik-Unterkapitel dar. Implikaturen hingegen sind Aspekte der vom Sprecher intendierten Bedeutung, die nicht explizit vom Sprecher genannt wurden. Implikaturen können daher als pragmatisch motivierte Schlüsse aufgefasst werden.

Der letzte Abschnitt 3.7.4 dieses Pragmatik-Unterkapitels stellt Methoden der Benutzermodellierung vor, die für die Computerlinguistik relevant sind. Sprachliche Anweisungen an Benutzer von Computersystemen sollten idealerweise an die Fähigkeiten und Interessen des jeweiligen Benutzers angepasst sein. Zum Beispiel sollte ein medizinisches Informationssystem, das nicht nur Ärzten, sondern auch medizinischen Laien zur Verfügung steht, dem Laien die gewünschte Information sprachlich anders präsentieren als dem Fachmann.

3.7.1 Text, Diskurs und Dialog

Helmut Horacek

Menschen kommunizieren in natürlicher Sprache in ganz verschiedener Weise, im persönlichen Gespräch, über Telefon, durch das Lesen von Zeitungen oder Büchern. In allen Fällen bestehen dabei die sprachlichen Äußerungen aus mehreren Sätzen, sei es von einem Autor oder einer Gruppe von Autoren gemeinsam, wie bei Zeitungsartikeln, oder von verschiedenen Leuten, wie bei Gesprächen. Die ursprüngliche, nicht analysierte Form dieser Äußerungen wird dabei als **Text** angesehen: „Text ist eine vortheoretisch intuitive, weder quantitative noch qualitative definierte Kategorie sprachlicher Äußerungen von mehr als einem Satz, die sich vorwiegend auf schriftliche Erzeugnisse unterschiedlichster Form und Funktion bezieht“ (Bußmann 1990). Wenn an der Kommunikation mindestens zwei Personen beteiligt sind, die sich sowohl selbst äußern, als auch die Äußerungen der anderen unmittelbar wahrnehmen, spricht man von einem **Dialog**, einer Wechselrede zwischen mindestens zwei Sprechern.

Ein Text besteht also grundlegend aus mehreren Sätzen. Zur formalen Beschreibung und Analyse von Sätzen werden in diesem Buch in mehreren Kapiteln diverse Verfahren vorgestellt. Diese Verfahren kategorisieren einen Satz typischerweise als korrekt oder als fehlerhaft, und sie liefern eine der Syntax und Semantik entsprechende Repräsentation dieses Satzes, bzw. sie erzeugen einen

Satz, der einer solchen Repräsentation entspricht. Der Zusammenhang von mehreren Sätzen und die Bezüge zwischen ihnen werden dadurch nicht erfasst. Gerade diese Aspekte sind jedoch für das Verständnis eines Textes bzw. für die Erzeugung eines geeigneten Textes unerlässlich. Reale Texte sind mehr als eine bloße Aneinanderreihung von Sätzen. Sie weisen einen syntaktischen, semantischen und pragmatischen Zusammenhang auf, der einen Text erst zu einem **kohärenten**, also zusammenhängenden, Text macht. Ein kohärenter Text wird auch als **Diskurs** bezeichnet, so dass jede vorkommende Äußerung durch die Kenntnis des vorausgegangenen und folgenden Kontexts interpretierbar wird.

Textkohärenz wird unter anderem durch syntaktische Eigenschaften unterstützt, wie die Verwendung von deiktischen Elementen, Pronomen, Tempusabfolge und konjunktionalen Verknüpfungen. Darüber hinaus bestimmen auch semantisch-pragmatische Indizien die Kohärenz eines Diskurses. Diese umfassen etwa Verträglichkeitsbeziehungen, Präspositionen, Implikaturen (vgl. den Abschnitt 3.7.3) sowie Referenzbeziehungen.

Zur Beschreibung des Aufbaus und anderer Eigenschaften von Diskursen sind in der Linguistik eine Reihe von Theorien vorgeschlagen worden. Eine der bekanntesten dieser Theorien, die dementsprechend in der Diskursplanung häufig eingesetzt wird, ist die **Rhetorical Structure Theory (RST)**, Mann und Thompson 1988), auf die wir uns in der Folge beziehen wollen, wobei die beschriebenen Techniken jedoch auch allgemeiner einsetzbar sind.

In der RST drücken **Diskursrelationen** Bezüge zwischen aneinander grenzenden Textsegmenten aus. Manche dieser Bezüge beschreiben untereinander gleichberechtigte Relationen (**Multi-Nucleus** Relationen), während die meisten Relationen Bezüge zwischen einem dominanten Textsegment (dem **Nucleus**) und einem untergeordneten (dem **Satelliten**) festlegen. Die Bedeutung der Relationen wird durch ihren Effekt im Diskurs und durch Bedingungen an den Inhalt von Nucleus und Satellit angeben.

Um Diskursrelationen zu operationalisieren, werden sie als **Planoperatoren** über Adressateninferenzen und Sprecherzielen ausgedrückt, so dass das Zusammensetzen mehrerer Textsegmente damit gesteuert werden kann. Ein solcher Planoperator hat Eingangsbedingungen, die festlegen, wann er angewendet werden kann und einen Effekt, der das Ergebnis nach der Anwendung beschreibt. Anforderungen an Nucleus und Satellit werden als semantische Bedingungen an die zu kombinierenden Inhalte formuliert. Außerdem können die an Stelle von Nucleus oder Satellit befindlichen Textsegmente durch Einfügen zusätzlicher Inhalte mittels anderer Relationen erweitert werden (diese Stellen werden **growth points** genannt).

Die häufig vorkommenden Relationen **SEQUENCE**, **ELABORATION**, **RESULT** und **BACKGROUND** können als Operatoren wie folgt ausgedrückt werden (*N* steht für das Nucleus- und *S* für das Satellit-Textsegment):

	Bedingungen
SEQUENCE	<i>N</i> kommt zeitlich/räumlich vor <i>S</i>
ELABORATION	<i>S</i> ist eine Detailinformation über <i>N</i>
RESULT	<i>S</i> ist eine Konsequenz von <i>N</i>
BACKGROUND	<i>S</i> ist eine Hintergrundinformation zu <i>N</i>

Der Effekt dabei ist folgender: Der Sprecher und der Hörer glauben beide, dass obige Relation zwischen den Eingabeentitäten besteht (z.B., dass *N* zeitlich/räumlich vor *S* vorkommt oder dass *S* Detailinformation zu *N* angibt).

Diese Operatoren können nun zur schrittweisen Planung von Diskursen verwendet werden. Dies kann in unterschiedlicher Weise erfolgen, je nach Art des Vorliegens der zu vermittelnden Information. In für den Einsatz von Planungsverfahren mit Diskursstrukturrelationen typischen Umgebungen liegen Informationen in Form einer Menge von elementaren Aussagen vor. Durch das Testen anwendbarer rhetorischer Relationen, jeweils auf Paare dieser Aussagen, können bei erfüllten Bedingungen einer Relation schrittweise umfangreichere Strukturen bottom-up aufgebaut werden. Eine andere Vorgehensweise besteht in der expliziten Vorgabe eines Ziels, das durch eine Aussage erfüllt werden kann. Durch Anwendung der Operatoren wird dieses Ziel dann solange zerlegt, bis die einzelnen Teilziele durch elementare Aussagen erfüllt werden können.

Während die RST zunächst ausschließlich zur Generierung eingesetzt wurde, gibt es seit einigen Jahren auch Ansätze für die Analyse. Erzeugt werden in beiden Fällen RST-Bäume, im Fall der Generierung aus konzeptuellen Spezifikationen, im Fall der Analyse aus semantischen Repräsentationen von Äußerungen. Während in der Generierung typischerweise nur eine solche Struktur aufgebaut wird, die durch lokale Präferenzen bestimmt wird, werden in der Analyse mehrere RST-Bäume aufgebaut, die mögliche Interpretationen eines Texts darstellen. Diesen Bäumen liegen meist unterschiedliche Intentionen zugrunde, so dass die plausibelste Interpretation durch Abgleich der Intentionen mit einem gegebenen Kontext ermittelt werden kann.

Das derzeit beste und am weitesten entwickelte Verfahren (Marcu 2000) ist sogar in der Lage, rhetorische Strukturen direkt aus Oberflächentexten aufzubauen. Wegen der Unzuverlässigkeit von Analyseverfahren bei satzübergreifenden Phänomenen wie der Interpretation von Diskursmarkern und dem Skopus rhetorischer Relationen verwendet Marcu eigene, aufeinander abgestimmte Teillprozesse zur Interpretation relevanter Diskurseigenschaften.

Zunächst werden die alternativen lokalen Interpretationen von Diskursrelationen gesammelt (*rhetorical grounding*). Diese Interpretationen ergeben sich aus Annahmen über Satzgrenzen basierend auf Interpunktions-, aus möglichen Bedeutungen und strukturelle Eigenschaften von Diskursmarkern, sowie aus Schlüssen über sehr einfache kohäsive Mittel, wie etwa Vorkommen von Pronomen, die das Vorhandensein einer **ELABORATION** nahelegen. Im zweiten Abschnitt werden die lokal möglichen Interpretationen zu konsistenten Diskusbäumen zusammengefügt (*rhetorical structure derivation*). Basierend auf einem mathematischen Modell von 12 Regeln zur kohärenten Bildung von Diskusbäumen werden die sich ergebenden Bedingungen propagiert und bottom-up

korrekte Diskursbäume erzeugt, wobei viele lokal mögliche Interpretationen ausgeschlossen werden können. Die entstehenden globalen Interpretationen werden dann nach empirisch motivierten Präferenzkriterien geordnet, etwa durch Bevorzugung rechtseingebetteter Diskursstrukturbäume.

Die Methode von Marcu wurde an unterschiedlichen Texten evaluiert, mit Erfolgsraten in der Größenordnung von 60 Prozent, was durch verbesserte empirische Grundlagen kontinuierlich gesteigert wird. Wenn man bedenkt, daß die Übereinstimmung menschlicher Experten bei der Diskursinterpretation selten 90 Prozent übersteigt, sind die Ergebnisse des Verfahrens von Marcu durchaus beachtlich.

Generell erlauben die vorhandenen Techniken bereits die korrekte Analyse einfacher Texte mit einer beschränkten Anzahl rhetorischer Relationen. Zu einer breiteren Abdeckung sind jedoch noch intensivere Ausarbeitungen in Bezug auf das Repertoire an rhetorischen Relationen, textuell verfügbarer Information als Evidenz für das Vorhandensein rhetorischer Relationen, sowie die Kombination verschiedener Evidenzen zur Ermittlung der plausibelsten rhetorischen Relation vonnöten.

3.7.2 Anaphernresolution

Michael Strube

Viele text- oder dialogverstehende Systeme müssen in der Lage sein, die Entitäten (d.h. abstrakte oder konkrete Gegenstände) zu identifizieren, auf die sich ein Sprecher oder Verfasser durch **Referenzausdrücke** bezieht. Die Entität, auf die sich ein Sprecher bezieht, wird **Referent** genannt. Gehört sie zum **Diskursmodell** eines Diskursteilnehmers, wird sie **Diskursentität** genannt. Wenn zwei Referenzausdrücke dazu verwendet werden, denselben Referenten zu bezeichnen, dann werden sie als **koreferent** bezeichnet. **Anaphern** sind die Referenzausdrücke, die dazu verwendet werden, auf einen schon in den Diskurs eingeführten Referenten zu verweisen. Die automatische Auflösung von Anaphern wird **Anaphernresolution** genannt. Diskursteilnehmer (Sprecher oder Verfasser) führen den Akt des **Referierens** durch das Äußern eines Referenzausdrucks aus (Sidner 1983). Zu den wichtigsten Referenzausdrücken zählen definite Nominalphrasen, Eigennamen, Personal- und Demonstrativpronomen (s. Beispiel (3.175), Übersetzung aus dem Brown Corpus cn03).

- (3.175) (a) *Der Posten war nicht tot.*
 (b) *Er zeigte, in der Tat, Lebenszeichen ...*
 (c) *Er war teilweise mit einer Kavallerieuniform bekleidet.*
 (d) *Mike zog ihm diese aus und legte sie selbst an.*
 (e) *Er fesselte und knebelte den Mann ...*

Zu den Anwendungen, die von einer Anaphernresolutionskomponente profitieren können, gehören u.a. das Informationsmanagement (Unterkapitel 5.3), natürlichsprachliche Dialogsysteme (Unterkapitel 5.5) und maschinelle Übersetzung (Unterkapitel 5.7). Die Generierung angemessener Referenzausdrücke dagegen ist für natürlichsprachliche Generierungssysteme wichtig (Unterkapitel 3.8 und 5.6).

Auflösung von Pronomen und definiten NPs

Das Bestimmen des richtigen Antezedenzials eines anaphorischen Ausdrucks wird durch Filter und Präferenzen geleistet. Filter dienen dazu, die Menge möglicher Antezedenzials zu beschränken. Als Filter werden bezeichnet:

- Kongruenz mit dem Antezedenzials in Numerus und Genus bei Pronomen, in Numerus bei definiten Nominalphrasen;
- syntaktische Restriktionen für satzinterne Anaphern, wie sie etwa durch die Bindungstheorie von Grammatiktheorien zur Verfügung gestellt werden;
- Kongruenz in Bezug auf semantische Typen oder Selektionsrestriktionen (Unterkapitel 3.6).

Präferenzen dagegen dienen dazu, die verbleibenden möglichen Antezedenzen aufgrund ihrer **Salienz** (engl. *salience*) zu ordnen. Mit Salienz bezeichnet man, wie prominent oder aktiv eine Diskursentität im Diskursmodell ist. Verschiedene Faktoren tragen zur Salienz bei, u.a.:

- die Distanz zwischen Anapher und möglichem Antezedenzen;
- die grammatischen Funktionen von Anapher und möglichem Antezedenzen (Präferenz für Subjekt) sowie deren Parallelismus;
- die Realisierung des Antezedenzen (Pronomen, definite NP, etc.);
- die Länge anaphorischer Ketten;
- für definite Nominalphrasen und Eigennamen die Identität oder die Ähnlichkeit auf Zeichenkettenebene.

Alle im Folgenden beschriebenen Ansätze greifen auf diese oder ähnliche Faktoren zurück, um Präferenzen zu bestimmen. Bei manchen Ansätzen sind diese Faktoren explizit, bei anderen im Algorithmus verborgen.

Linguistische Ansätze

Erste auf linguistischen Methoden aufbauende Anaphernresolutionsalgorithmen wurden Ende der 1970er Jahre vorgestellt. Der **Hobbs-Algorithmus** (Hobbs 1978) implementiert sowohl Filter als auch Präferenzen, um Pronomen aufzulösen. Er arbeitet lediglich auf der Basis syntaktischer Strukturen und ist als sog. *tree walk* implementiert. Da der Algorithmus eine vollständige syntaktische Analyse voraussetzt, sind Bindungsrestriktionen berücksichtigt. Der Algorithmus führt eine Breitensuche von links nach rechts durch, um Antezedenzen innerhalb des Satzes zu finden. Ist der Algorithmus innerhalb des Satzes erfolglos, werden alle weiteren Sätze von rechts nach links vorgehend auf die gleiche Weise durchsucht. Der Hobbs-Algorithmus bezieht Distanzmaße indirekt durch die Suchstrategie mit ein. Der Algorithmus realisiert eine Präferenz für Antezedenzen in Subjektposition, indem er bei der Suche zuerst auf die herausragende Position des Subjekts in der syntaktischen Struktur trifft. Obwohl der Hobbs-Algorithmus weder Semantik noch Weltwissen berücksichtigt, weist er für englische Texte sehr gute Ergebnisse auf und gilt auch heute noch als Vergleichsmaßstab (engl. *baseline*) für neue Verfahren (Tetreault 2001).

Parallel zum Hobbs-Algorithmus wurden das **Fokus-Modell** (Sidner 1983) und das **Centering-Modell** (Grosz et al. 1995; Walker et al. 1998) entwickelt. Diese präferenzbasierten Modelle sind kognitionswissenschaftlich motiviert und basieren auf Methoden aus der Künstlichen Intelligenz. Das Fokus-Modell wurde in erster Linie für die Anaphernresolution entwickelt und eingesetzt (Sidner 1983, Suri et al. 1999). Das Centering-Modell dagegen wurde als allgemeines Modell **lokaler Kohärenz** entwickelt. Es wurde auf eine Vielzahl linguistischer Phänomene in verschiedenen Sprachen angewendet.

Das Centering-Modell ist dazu vorgesehen, die Beziehung zwischen dem **Fokus der Aufmerksamkeit** (engl. *center of attention*), der Bestimmung der Form der Referenzausdrücke und der Kohärenz des Diskurses zu modellieren. Das Modell besteht vor allem aus zwei Konstrukten, dem **backward-looking center (Cb)** und der Liste der **forward-looking center (Cf)** (das erste Element der Cf-Liste wird auch **preferred center (Cp)** genannt). Beide Konstrukte sind an **Äußerungen** U_i (von engl. *utterances*, d.h. Sätze oder Teilsätze in Texten) gebunden. Mit Hilfe von Cb und Cf kann der Übergang zwischen einzelnen Äußerungen beschrieben werden (s. Tabelle 3.11). Die Übergangstransition *continue* drückt aus, dass aufeinanderfolgende Äußerungen von demselben Gegenstand handeln, da dieser Gegenstand in beiden Äußerungen eine prominente Stellung einnimmt. Die *retain*-Transition zeigt ebenfalls, dass beide Äußerungen denselben Gegenstand zum Thema haben. Gleichzeitig deutet diese Transition durch Wortstellung und/oder syntaktische Realisierung an, dass ein Wechsel zu einem anderen Gegenstand unmittelbar bevorsteht. Die beiden *shift*-Transitionen beschreiben den Wechsel zu einem anderen Gegenstand, *smooth-shift* einen Wechsel, der durch ein *retain* eingeleitet wurde, *rough-shift* dagegen beschreibt einen abrupten Wechsel. Die Transitionen treten häufig in bestimmten Mustern auf (etwa *continue – retain – smooth-shift*), die zur Beschreibung der lokalen Kohärenz von Texten dienen können.

	$Cb(U_i) = Cb(U_{i-1})$ OR no $Cb(U_{i-1})$	$Cb(U_i) \neq Cb(U_{i-1})$
$Cb(U_i) = Cp(U_i)$	CONTINUE	SMOOTH-SHIFT
$Cb(U_i) \neq Cp(U_i)$	RETAIN	ROUGH-SHIFT

Tabelle 3.11: Centering-Übergänge nach *Brennan et al.* (1987)

Das Centering-Modell gibt zwei Regeln vor, die sich auf die Realisierung von Referenzausdrücken und die lokale Kohärenz des Diskurses beziehen.

Regel 1: Wenn ein Element der $Cf(U_i)$ durch ein Pronomen realisiert ist, dann muss der Cb auch durch ein Pronomen realisiert sein.

Regel 2: Die Centering-Transitionen haben folgende Ordnung: CONTINUE, RETAIN, SMOOTH-SHIFT, ROUGH-SHIFT.

Brennan et al. (1987) beschreiben einen Pronomenresolutionsalgorithmus, den sog. **BFP-Algorithmus**, der für jede Äußerung im Wesentlichen drei Schritte ausführt:

1. Erzeuge alle möglichen Cb - Cf -Kombinationen für jede mögliche Belegung von Pronomen mit Antezedenten.
2. Filtere die erzeugten Kombinationen durch syntaktische Restriktionen (Bindungstheorie), Selektionsrestriktionen, Centering-Regeln.
3. Ordne die Kombinationen durch die Ordnung der Transitionen.

Brennan et al. (1987) ordnen die *Cf* nach grammatischen Funktionen (Subjekt > Objekt > andere). Unter dieser Annahme wird Beispiel (3.175) vom BFP-Algorithmus analysiert, wie vereinfacht in Tabelle 3.12 gezeigt. Weil das Textfragment mit Äußerung (3.175a) beginnt, existieren hier weder *Cb* noch eine Transition. Das Pronomen *er* in (3.175b) ist koreferent mit dem ersten Element der *Cf*(3.175a), deshalb wird diese Äußerung mit CONTINUE versehen. Das Gleiche gilt für Äußerung (3.175c). In Äußerung (3.175d) wird die Diskursentität MIKE eingeführt. Da die Diskursentität POSTEN durch das Pronomen *ihm* realisiert ist, bleibt POSTEN *Cb* von (3.175d). Da aber *Mike* Subjekt von (3.175d) ist, steht die Diskursentität an der ersten Stelle der *Cf*(3.175d). Damit erhält diese Äußerung eine RETAIN-Transition. In Äußerung (3.175e) entsteht eine Ambiguität, da das Pronomen *er* sowohl auf MIKE als auch auf POSTEN verweisen kann. Da die Transition SMOOTH-SHIFT der Transition ROUGH-SHIFT vorgezogen wird, bestimmt der Algorithmus MIKE als Antezedenzen von *er*.

(3.175a)	<i>Cb</i> : –	<i>Cf</i> : [POSTEN: <i>Posten</i>]	–
(3.175b)	<i>Cb</i> : POSTEN: <i>er</i>	<i>Cf</i> : [POSTEN: <i>er</i> , ZEICHEN: <i>Zeichen</i>]	CONTINUE
(3.175c)	<i>Cb</i> : POSTEN: <i>er</i>	<i>Cf</i> : [POSTEN: <i>er</i> , UNIFORM: <i>Uniform</i>]	CONTINUE
(3.175d)	<i>Cb</i> : POSTEN: <i>ihm</i>	<i>Cf</i> : [MIKE: <i>Mike</i> , POSTEN: <i>ihm</i>]	RETAIN
(3.175e)	<i>Cb</i> : MIKE: <i>er</i> <i>Cb</i> : MIKE: <i>Mann</i>	<i>Cf</i> : [MIKE: <i>er</i> , POSTEN: <i>Mann</i>] <i>Cf</i> : [POSTEN: <i>er</i> , MIKE: <i>Mann</i>]	SMOOTH-SHIFT ROUGH-SHIFT

Tabelle 3.12: Beispiel (3.175) analysiert mit dem BFP-Algorithmus

Das Centering-Modell konnte mit Erfolg auf die Pronomenauflösung in verschiedenen Sprachen angewendet werden (u.a. Englisch: Brennan et al. 1987, Tetreault 2001, Japanisch: Walker et al. 1994, Deutsch: Strube und Hahn 1999). Die Anpassung auf andere Sprachen konnte gelingen, weil das Modell Präferenzen explizit macht. Die *Cf*-Präferenzordnung kann so an die Gegebenheiten der jeweiligen Sprache angepasst werden. Das Centering-Modell konnte auch auf andere Fragestellungen angewendet werden, etwa auf die Bestimmung der Wortstellung, die Generierung von Referenzausdrücken oder die Bestimmung lokaler Kohärenz.

Die Flexibilität des Centering-Modells ist gleichzeitig auch sein größter Nachteil. Wichtige Teile sind nicht spezifiziert, so dass unterschiedliche Interpretationen des Modells entstanden (u.a. Brennan et al. 1987, Strube und Hahn 1999). Als wichtigstes Problem erwies sich, dass der Begriff der Äußerung nicht spezifiziert ist (Kameyama 1998). Dies führte zu Varianten, in denen die Äußerung als (in)finiter Teilsatz definiert ist, und zu anderen, in denen sie als vollständiger Satz definiert ist. Das eine Extrem kann gut mit satzinternen Anaphern umgehen, das andere mit satzübergreifenden.

Kehler (1997) stellt fest, dass der BFP-Algorithmus das Versprechen des Centering-Modells, ein kognitives Modell zu sein, nicht einlöst, da es inkrementelle Verarbeitung nicht erlaubt. Strube (1998) stellt eine inkrementelle Variante vor, die zugleich die Verarbeitung satzinterner und satzübergreifender Anaphern vereinheitlicht (s. auch Tetreault 2001).

Heuristiken

Parallel zum eher allgemein gehaltenen Centering-Modell wurden spezialisierte Anaphernresolutionsalgorithmen entwickelt, die auf Heuristiken beruhen. Statt auf linguistischen Regeln basieren diese Algorithmen meist auf Gewichten, die der Entwickler bestimmten Faktoren zuweist. Da die Gewichte manuell vergeben werden, können sie für einen Korpus oder eine Sprache optimiert werden. Der wichtigste Vertreter heuristischer Anaphernresolutionsalgorithmen ist der Algorithmus von Lappin und Leass (1994) für die Pronomenauflösung.

Der **Lappin & Leass-Algorithmus** benötigt als Eingabe eine vollständige syntaktische Analyse. Er beruht auf einer Reihe von **Salienzfaktoren**, zu denen die grammatische Funktion des Antezedenzials gehört, der Parallelismus grammatischer Funktionen von Antezendent und Anapher, die Anzahl der Elemente in einer anaphorischen Kette, die Distanz zwischen Antezendent und Anapher. Lappin & Leass' Faktoren und die dazugehörigen Gewichte sind in Tabelle 3.13 dargestellt.

Salienzfaktor	Gewicht
Sentence recency	100
Subject emphasis	80
Existential emphasis	70
Accusative emphasis	50
Indirect object and oblique complement emphasis	40
Head noun emphasis	80
Non-adverbial emphasis	50

Tabelle 3.13: Faktoren und Gewichte im Lappin & Leass Algorithmus

Die Berechnung der Gewichte für eine Anapher-Antezedenzials-Relation wird ausgelöst, wenn der Algorithmus eine Anapher erreicht. Antezedenzials im unmittelbar vorhergehenden Satz erhalten für den Faktor *recency* einen Wert von *100*, Antezedenzials im davorliegenden Satz nur noch einen Wert von *50*. Ist der Antezendent Subjekt eines Satzes, erhält er für den Faktor *subject emphasis* den Wert *80*, ansonsten *0*. Das Gewicht eines Antezedenzials errechnet sich aus der Addition der Einzelgewichte. Der Algorithmus verwendet (vorwiegend grammatische) Restriktionen und (pragmatische) Präferenzen:

- Restriktionen
 - satzinterner syntaktischer Filter (Bindungstheorie);
 - morphologischer Filter (Kongruenz);
 - Identifikation pleonastischer (d.i. nicht-referentieller) Pronomen;
 - Algorithmus zur Bindung von Reflexivpronomen;
- Präferenzen
 - Zuweisung der Salienzfaktoren.

Auch dieser Algorithmus benötigt keine semantische Analyse und kein Weltwissen. Nur die Notwendigkeit, auf einer vollständigen syntaktischen Analyse aufzubauen, verhindert die Anwendung des Lappin & Leass-Algorithmus auf Texte unterschiedlichster Herkunft. Der Gedanke, auf die vollständige syntaktische Analyse zu verzichten, liegt nahe und wurde schon von Kennedy und Boguraev (1996) realisiert. Sie verließen sich lediglich auf einen POS-Tagger und Nominalphrasenidentifikation. Ihre Ergebnisse für die Pronomenauflösung waren so ermutigend, dass in der Folge eine ganze Reihe von Wissenschaftlern heuristische Ansätze verfolgten, die mit beschränktem Wissen auskamen (Baldwin 1997; Mitkov 1998). Heuristiken wurden aber auch auf die Auflösung anderer Arten von Anaphern angewendet, etwa auf die Auflösung von definiten NPs (Vieira und Poesio 2000).

Heuristiken werden im Bereich der Anaphernresolution meist auf bestimmte, eher eingeschränkte Domänen angewendet und weisen dort gute Ergebnisse auf. Dies liegt daran, dass die Verfahren im Allgemeinen recht robust sind und die Gewichtung der Parameter an die Domäne angepasst werden können. Da die Bestimmung der Gewichte in der Regel manuell geschieht, unterliegen sie einer gewissen Beliebigkeit. Die Bestimmung optimaler Gewichte dürfte wegen des großen Suchraums manuell kaum möglich sein. Die Interaktion von Faktoren können Heuristiken nicht berücksichtigen.

Machine learning-Ansätze, Baseline

Die Notwendigkeit, Anaphern in heterogenen Dokumentsammlungen aufzulösen, legte seit Mitte der 90er Jahre den Einsatz von **korpus-basierten Methoden** und **Methoden des maschinellen Lernens (ML)**nahe. Diese Methoden sind in der Regel robuster als linguistische und heuristische Ansätze und entdecken häufig statistische Zusammenhänge in den Daten, die der menschlichen Analyse verborgen bleiben. Die meisten dieser Ansätze sind überwacht (engl. *supervised*) und benötigen einen Korpus annotierter Daten, in denen anaphorische Relationen markiert sind. Im Bereich der Pronomenauflösung scheinen ML-Ansätze herkömmlichen Methoden überlegen zu sein (Ge et al. 1998). Die Auflösung von definiten NPs benötigt auch bei diesen Methoden ein gewisses Maß an Domänen- oder Weltwissen. Deshalb ist es sinnvoll, auf die lexikalische Datenbasis **WordNet** (Unterkapitel 4.3) oder vergleichbare Wissensquellen zuzugreifen. Das linguistische Wissen des Entwicklers fließt nicht in einen Anaphernresolutionsalgorithmus, sondern in die Bestimmung relevanter Faktoren ein.

Soon et al. (2001) beschreiben einen ML-Ansatz zur Anaphernresolution, der sich zum Vergleichsmaßstab für neuere Arbeiten entwickelt hat. Sie verwenden die Korpora, die für die *Message Understanding Conference* (MUC6, MUC7) erstellt wurden. Die Korpora sind mit Koreferenzrelationen annotiert und bestehen aus Trainings- und Testdatensätzen. Die Eingabe in das System von Soon et al. (2001) besteht aus unannotierten Texten. Um Referenzausdrücke zu bestimmen, setzen sie eine Kette von sprachverarbeitenden Komponenten ein: Tokenisierung, Satzsegmentierung, Morphologie, POS-Tagging, NP-Identifikation, *Named Entity Recognition*, Bestimmung der semantischen Klasse. Mit Hilfe dieser NLP-

Komponenten bestimmen sie Referenzausdrücke und weisen den Faktoren aus Tabelle 3.14 (in Klammern sind die Werte angegeben, die ein Faktor annehmen kann, *T* steht für *true*, *F* für *false*) Werte zu.

1	Distance	Distanz in Sätzen zwischen Ana und Ante (0, 1, 2, ...)
2	i-Pronoun	ist Ante Pronomen? (T, F)
3	j-Pronoun	ist Ana Pronomen? (T, F)
4	String Match	sind Ana und Ante identisch? (T, F)
5	Definite Noun Phrase	ist Ana definite NP? (T, F)
6	Demonstrative Noun Phrase	ist Ana definite NP mit demonstrativem Artikel? (T, F)
7	Number Agreement	sind Ana und Ante kongruent in Bezug auf Numerus? (T, F)
8	Semantic Class Agreement	sind Ana und Ante kongruent in Bezug auf die folgenden in einer <i>isa</i> -Hierarchie angeordneten semantischen Klassen <i>female</i> , <i>male</i> , <i>person</i> , <i>organization</i> , <i>location</i> , <i>date</i> , <i>time</i> , <i>money</i> , <i>percent</i> , <i>object</i> ? (T, F)
9	Gender Agreement	sind Ana und Ante kongruent in Bezug auf Genus? (T, F)
10	Both-Proper-Names	sind Ana und Ante Eigennamen? (T, F)
11	Alias	sind Ana und Ante ähnlich (<i>substring match</i> , Abkürzungen)? (T, F)
12	Appositive	ist Ana Apposition von Ante? (T, F)

Tabelle 3.14: Faktoren von Soon et al. (2001)

Um einen ML-Algorithmus auf das Problem der Anaphernresolution anzuwenden, muss die Aufgabe des Bestimmens des Antezedenzien einer Anapher entsprechend umformuliert werden. Dies geschieht häufig dadurch, dass das Problem als **binäre Klassifikation** ausgedrückt wird, womit es für viele auf Klassifikation beruhende ML-Algorithmen zugänglich wird. Dazu werden Paare aus Anapher und möglichem Antezedenzien gebildet, die als positiv (T) klassifiziert werden, wenn sie in der Tat koreferent sind, als negativ (F), wenn sie nicht koreferent sind. Tabelle 3.15 zeigt zwei Trainingsinstanzen, die je einen Vektor (engl. *feature vector*) darstellen. Die Anapher ist jeweils das Pronomen *er* aus Beispiel (3.175e), die Antezedenzien der Eigenname *Mike* und das Pronomen *ihm* aus Beispiel (3.175d).

Ante	Ana	1	2	3	4	5	6	7	8	9	10	11	12	Zielkonzept
<i>Mike</i>	<i>er</i>	1	F	T	F	F	F	T	T	T	F	F	F	T
<i>ihm</i>	<i>er</i>	1	T	T	F	F	F	T	T	T	F	F	F	F

Tabelle 3.15: Feature Vector

Wenn man über ausreichend annotierte Daten verfügt, dann weist ein ML-Verfahren gegenüber den anderen beschriebenen Ansätzen zur Anaphernresolution mehrere Vorteile auf. Abhängig von den Faktoren, die man verwendet, ist ein solcher Ansatz domänen- oder sogar sprachunabhängig. Je nach dem ML-Verfahren, das man verwendet, wird die Interaktion zwischen einzelnen Faktoren berücksichtigt. Der quantitative Beitrag einzelner Faktoren zum Gesamtergebnis kann bestimmt werden. Schließlich wurden sehr gute Ergebnisse für die Auflösung von Pronomen und Eigennamen berichtet, während die Auflösung von definiten NPs mehr (Domänen- oder Welt-)Wissen benötigt als in den Faktoren in Tabelle 3.14 enthalten ist.

Machine learning-Ansätze, weitere Entwicklungen

Das System von Soon et al. (2001) hat sich als Vergleichsmaßstab etabliert, weil es relativ gute Ergebnisse mit einem einfachen Modell des maschinellen Lernens auf der Basis einer überschaubaren Menge von Faktoren erzielt. Eine Schwäche dieses Systems ist, daß es ohne Kenntnis des Kontexts lokale Entscheidungen trifft. Deshalb wurden in der Folge ML-Methoden eingesetzt, die global optimale Entscheidungen treffen können. Eine weitere Schwäche des Systems von Soon et al. (2001) ist der Mangel an linguistischem Wissen, an semantischem Wissen und an Weltwissen.

Ein erster Versuch, Kontext in ein maschinelles Lernverfahren für die Koreferenzresolution einzubeziehen, ist das *Twin candidate*-Modell von Yang et al. (2008). In diesem System bestehen Lerninstanzen aus der potentiellen Anapher und je einem positiven und negativen Antezedenzkandidaten. In einem zweiten Schritt werden dann alle zu einer bestimmten potentiellen Anapher gehörenden Instanzen wie in einem Turnier miteinander verglichen, so dass sich am Ende eine Präferenz für einen Antezedenten ergibt.

Luo et al. (2004) repräsentieren Koreferenzresolution durch einen *Bell tree*, der alle möglichen Koreferenzmengen darstellt. Wegen der exponentiellen Komplexität des *Bell tree* müssen Luo et al. (2004) den Baum frühzeitig beschneiden und effiziente Suchheuristiken anwenden. Deshalb ist nicht sichergestellt, ob dieses Modell eine global optimale Lösung findet. Ng (2005) stellt ein *Reranking*-Modell für Koreferenzresolution vor, bei dem einige Dutzend verschiedene Systemvariationen miteinander kombiniert werden. Schließlich stellen Yang et al. (2008) ein Modell vor, in dem die Instanzen statt Referenzausdrücken Diskursentitäten repräsentieren. Das Modell nutzt *Inductive Logic Programming*, um globale Beschränkungen zu formulieren.

Ein weiterer Weg, Koreferenzresolution globales Wissen zur Verfügung zu stellen, stellen unüberwachte ML-Ansätze (engl. *unsupervised*) dar. Schon Cardie und Wagstaff (1999) beschreiben einen *Clustering*-Ansatz, der ohne annotierte Trainingsdaten auskommt. Während die Ergebnisse von Cardie und Wagstaff (1999) weit hinter denen überwachter Lernverfahren zurückliegen, kann der unüberwachte Ansatz von Haghighi und Klein (2007) mit überwachten Lernverfahren konkurrieren.

Die meisten der beschriebenen Ansätze benutzen Faktoren, die auf dem System von Soon et al. (2001) beruhen. Deshalb ist ein weiteres Feld, das Fortschritte verspricht, die Entwicklung linguistisch motivierter Faktoren und der Einbezug von semantischem und Weltwissen. Schon Ng und Cardie (2002) beschreiben eine Erweiterung der Faktoren. Der unkontrollierte Einbezug aller Faktoren allerdings verwirrt das System. Erst eine manuelle Auswahl der besten Faktoren bewirkt eine moderate Verbesserung der Ergebnisse. Bengtson und Roth (2008) bestimmen eine sehr leistungsfähige Menge von Faktoren, die auch dazu dienen, Anaphorizität zu lernen. Kehler et al. (2004), Yang et al. (2005) und Ponzetto und Strube (2006) ergänzen Koreferenzresolutionssysteme um Wissen über Selektionsrestriktionen, die eine Verbesserung für die Pronomenauflösung bewirken. Weltwissen dagegen hilft bei der Auflösung definiter Nominalphrasen – jenes Typs anaphorischer Ausdrücke, der ML-Ansätzen noch die größten Schwierigkeiten bereitet. Harabagiu et al. (2001) greifen auf WordNet zurück, während Ponzetto und Strube (2006) Wissen aus der *Online-Enzyklopädie Wikipedia* extrahieren und der Koreferenzauflösung als Faktor zur Verfügung stellen.

Während fast alle beschriebenen Verfahren für englischsprachige Texte entwickelt wurden, gibt es nur wenige Arbeiten, die Verfahren des maschinellen Lernens für die Koreferenzauflösung in deutschen Texten anwenden. Strube et al. (2002) entwickeln ein Verfahren für die Analyse deutscher Texte aus dem Tourismusbereich, während Versley (2007) ein System zur Analyse des **TüBa-D/Z-Korpus** (Telljohann et al. 2004) vorstellt.

Neben der Anwendung auf Textkorpora gibt es einige wenige Arbeiten, die sich auch mit der Auflösung von Anaphern in gesprochener Sprache beschäftigen, wobei diese in der Regel eine weitaus höhere Dichte an anaphorischen Ausdrücken aufweist als Texte. Außerdem werden Pronomen in gesprochener Sprache viel häufiger dafür verwendet auf abstrakte Dinge zu verweisen, was dazu führt, dass die Antezedenzien nicht nur aus nominalen Konstituenten bestehen, sondern auch aus Verbalphrasen, (Teil-)Sätzen und ganzen Diskurssegmenten (Eckert und Strube 2000). Byron (2002) stellt ein symbolisches Verfahren vor, während Strube und Müller (2003) ein Verfahren beschreiben, das auf maschinellem Lernen beruht. Müller (2007) entwickelt eine vollständig automatische Methode, die *it*, *this* und *that* auflöst.

Daten, Evaluierung, Systeme

Während der MUC-Korpus in den ersten Jahren der Anwendung maschineller Lernverfahren auf Koreferenzresolution als Referenzkorpus diente, haben nun die Daten der ACE-Initiative (*Automatic Content Extraction Evaluation*) diese Rolle übernommen. Für die deutsche Sprache steht die um Koreferenzrelationen erweiterte TüBa-D/Z-Baumbank zur Verfügung.

Im Gegensatz zu syntaktisch annotierten Korpora (etwa *Penn Treebank*, Unterkapitel 4.2) stehen für die Anaphernresolution nur sehr wenige annotierte Korpora zur Verfügung. Deshalb muss für eine neue Anwendung häufig erst ein Korpus mit anaphorischen Relationen annotiert werden. Während bei früheren annotierten Korpora (etwa den MUC-Korpora) die Annotation *inline* hinzuge-

fügt wurde, gibt es jetzt Annotationsschemata und -tools, die die **Standoff-Annotation** (Thompson und McKelvie 1997) unterstützen. Hierbei werden die Basisdaten von den Annotationen getrennt. Ein Tool für die Annotation von anaphorischen Relationen in Texten und (multimodalen) Dialogen ist *MMAX2* (Müller und Strube 2006, aktuelle Version unter <http://mmax2.sourceforge.net>). Bei der Annotation ist darauf zu achten, dass die Annotationen von anderen Annotierern reproduziert werden können (Passonneau 2004).

Bei der **Evaluierung** von Koreferenzresolutionssystemen kann man davon abstrahieren, den korrekten Antezedenten für eine Anapher zu finden. Vielmehr geht es darum, die Diskursentität zu bestimmen, auf die mit dem Referenzausdruck verwiesen werden soll. Damit kann die Evaluierung so umformuliert werden, dass das Ziel der Anaphernresolution ist, einen Referenzausdruck einer Menge zuzuordnen, deren Elemente dieselbe Diskursentität beschreiben. Vilain et al. (1995) beschreiben einen modelltheoretischen Algorithmus zur Evaluierung von Anaphernresolutionsalgorithmen. Er bildet Mengen von referenzidentischen Referenzausdrücken und berechnet die minimale Anzahl an fehlenden Verbindungen, um ein von der Anaphernresolution erzeugtes Ergebnis in eine von einer Annotation dargestellte Vorlage zu überführen.

Der Algorithmus von Vilain et al. (1995) hat sich als Standard im Bereich der Anaphernresolution durchgesetzt und ist anderen Maßen (wie etwa der Anzahl korrekt aufgelöster Anaphern) vorzuziehen. Da das Evaluierungsmaß von Vilain et al. (1995) in Grenzfällen nicht verlässlich ist, sollte parallel dazu immer entweder das *B-cubed*-Maß von Bagga und Baldwin (1998) oder das *CEAF*-Maß von Luo (2005) angegeben werden.

Literaturhinweise

Das Centering-Modell gilt als wichtiges Modell lokaler Kohärenz. Neben der Originalarbeit (Grosz et al. 1995) sind im Bereich der Anaphernresolution besonders Brennan et al. (1987), Walker et al. (1994), Strube und Hahn (1999) und Tetreault (2001) zu nennen. Im Bereich der Pronomenauflösung dient der Hobbs-Algorithmus (Hobbs 1978) immer noch als wichtiger Vergleichsmaßstab, Lappin und Leass (1994) beschreiben einen wichtigen heuristischen Ansatz. Eine gute Beschreibung der Anwendung von Methoden des maschinellen Lernens auf die Anaphernresolution geben Soon et al. (2001).

Frei verfügbare Software:

JavaRAP: Reimplementierung des Pronomenauflösungssystems von Lappin und Leass (1994) für Englisch von Qiu et al. (2004)
<http://www.comp.nus.edu.sg/~qiul/NLPTools/JavaRAP.html>

OpenNLP: Koreferenzresolutionskomponente des OpenNLP-Systems basierend auf einem *Maximum Entropy Classifier*
<http://opennlp.sourceforge.net/>

GuiTAR: Modulares Koreferenzresolutionssystem, das verschiedene Strategien für verschiedene Arten von Referenzausdrücken implementiert (Poesio und

Kabadjov 2004)

<http://cswww.essex.ac.uk/Research/nle/GuiTAR/gtarNew.html>

BART: Modulares Koreferenzresolutionssystem, Reimplementierung von Soon et al. (2001), erlaubt andere Wissenquellen und andere *Machine learning*-Verfahren zu integrieren (Versley et al. 2008)
<http://www.assembla.com/wiki/show/bart-coref>

Annotierte Korpora:

MUC: Korpora, die für die *Message Understanding Conference* (MUC6, MUC7) erstellt wurden
http://www.itl.nist.gov/iaui/894.02/related_projects/muc/

ACE: Korpora, die für die ACE-Initiative erstellt wurden; die Annotation unterscheidet sich stark von der der MUC-Daten
<http://www.nist.gov/speech/tests/ace/>

TüBa-D/Z: Koreferenzannotation deutscher Zeitungstexte in der TüBa-D/Z-Baumbank
http://www.sfs.uni-tuebingen.de/en_tuebadz.shtml

3.7.3 Implikaturen und Präspositionen

Gerhard Jäger

Die computationelle Semantik (und Pragmatik) befasst sich in erster Linie mit der Frage, was das **Folgerungspotential** eines sprachlichen Zeichens bzw. einer sprachlichen Äußerung ist. Eine wichtige Klasse von Inferenzen eines Satzes werden durch die Zuweisung einer wahrheitskonditionalen Semantik (etwa im Sinne der DRT) implizit vorausgesagt. Angelehnt an den Folgerungsbegriff der formalen Logik (s. Unterkapitel 2.1) sagt man, dass der Satz *A* den Satz *B* impliziert, wenn es kein Szenario gibt, in dem *A* wahr und *B* falsch ist.

Diese Charakterisierung erfasst den intuitiven Folgerungsbegriff nur zum Teil. Man betrachte beispielsweise den folgenden Diskurs:

- (3.176) Abends im Restaurant.
- (a) A: *Willst du auch etwas essen?*
 - (b) B: *Ich habe schon gegessen. ...*
 - (c) *... Allerdings war das gestern.*

Im Kontext der Frage (3.176a) würde man zunächst unterstellen, dass aus Satz (b) folgt, B habe am selben Tag schon zu Abend gegessen. Diese Folgerung „verschwindet“ jedoch, wenn man Satz (c) mit in Betracht zieht. Anders gesagt, ist die fragliche Folgerung **anfechtbar**. Daher kann es sich nicht um eine normale Implikation handeln, da sich die kompositional (siehe Unterkapitel 3.6) determinierte wahrheitskonditionale Bedeutung von (b) ja durch die Äußerung von (c) nicht ändert.

Die Begriffe **Implikatur** und **Präsposition** fassen eine Reihe von Folgerungstypen zusammen, die vom wahrheitskonditionalen Implikationsbegriff nicht abgedeckt werden. Sie werden gemeinhin dem Gebiet der Pragmatik zugerechnet, weil sie nicht ausschließlich aus der kompositionalen Semantik ableitbar sind, sondern auch vom Äußerungskontext und den Absichten und Annahmen der Diskursteilnehmer abhängen.

Implikaturen

Der deutsche Begriff **Implikatur** ist eine Lehnübersetzung des engl. *implicature*. Hierbei handelt es sich um ein Kunstwort, das der englische Philosoph Paul Grice kreierte, um die Abgrenzung von der Implikation (*implication*) deutlich zu machen (vgl. Grice 1975). Es leitet sich vom Verb *to implicate* (jmd. mit etwas in Zusammenhang bringen) ab (was als pragmatischer *terminus technicus* üblicherweise als *implikatieren*, manchmal auch als *implizieren* übersetzt wird). Im weiteren Sinne bezeichnete Grice damit alle Folgerungen, die nicht direkt wahrheitskonditional sind. Hier unterscheidet er **konventionelle Implikaturen** von **konversationellen Implikaturen**. Konventionelle Implikaturen sind Folgerungen, die durch die konventionalisierte Bedeutung eines sprachlichen Zeichens festgelegt sind, ohne jedoch Teil der Wahrheitsbedingungen zu sein. Ein

typisches Beispiel ist etwa der Unterschied zwischen den Konjunktionen *und* und *aber*. Die Sätze

- (3.177) (a) *Gregor ist reich und gesund.*
 (b) *Gregor ist reich aber gesund.*

haben die selben Wahrheitsbedingungen, sind aber nicht wirklich synonym. Satz (3.177b) transportiert außer den Wahrheitsbedingungen die Information, dass zwischen Gregors Reichtum und seiner Gesundheit ein Kontrast besteht (was ohne weitere Hintergrundinformation abwegig wirkt). Dieser Aspekt der Bedeutung von (b) wäre nach Grice eine konventionelle Implikatur.

Konventionelle Implikaturen sind, wie Implikationen, Teil der lexikalisch fundierten und kompositionale berechneten Bedeutung eines Satzes. Davon zu unterscheiden sind die **konversationellen Implikaturen**. Dabei handelt es sich um Folgerungen, die sich aus der Annahme ergeben, dass sich die Diskursteilnehmer rational verhalten. Diese Idee lässt sich gut mit einem von Grice' Beispielen illustrieren:

- (3.178)
 (a) Autofahrer zu einem Passanten: *Mir ist das Benzin ausgegangen.*
 (b) Passant: *Um die Ecke ist eine Tankstelle.*

Die Antwort des Passanten impliziert, dass die besagte Tankstelle geöffnet ist. Dieser Sachverhalt folgt nicht aus der Bedeutung von (b) als solcher, sondern aus der Annahme, dass der Passant dem Autofahrer die Bedeutung von (b) in kooperativer, also nicht irreführender Absicht mitteilen möchte.

Grice unterstellt, dass Kommunikation eine kooperative gemeinsame Aktivität von Sprecher und Hörer ist. Daher kann man davon ausgehen, dass die Diskursteilnehmer dem **Kooperationsprinzip** folgen:

„Mache deinen Gesprächsbeitrag jeweils so, wie es von dem akzeptierten Zweck oder der akzeptieren Richtung des Gesprächs, an dem du teilnimmst, gerade verlangt wird.“

(Grice 1975, in der Übersetzung von A. Kemmerling, zitiert nach Meggle 1979, S. 248)

Daraus ergeben sich, Grice zufolge, vier **Konversationsmaximen** (ebenfalls zitiert nach Meggle 1979, S. 249/250):

1. Maxime der Quantität:

- Gestalte deinen Beitrag so informativ wie (für die gegebenen Gesprächszwecke) nötig.
- Mache deinen Beitrag nicht informativer als nötig.

2. Maxime der Qualität:

- Sage nichts, ws du für falsch hältst.
- Sage nichts, wofür dir angemessene Gründe fehlen.

3. **Maxime der Relevanz:** Sei relevant.

4. **Maxime der Modalität:** Sei klar.

Insbesondere:

- Vermeide Dunkelheit des Ausdrucks.
- Vermeide Mehrdeutigkeit.
- Sei kurz (vermeide unnötige Weitschweifigkeit).
- Der Reihe nach!

Die Implikatur des Satzes (3.178b), wonach die besagte Tankstelle geöffnet ist, ergibt sich beispielsweise aus der Annahme, dass der Sprecher das Kooperationsprinzip befolgt (dem Autofahrer also helfen will), und dass er die Relevanzmaxime befolgt (der Verweis auf eine geschlossene Tankstelle wäre irrelevant).

In vielen Situationen werden eine oder mehrere Maximen verletzt. Der Hörer kann aber dennoch davon ausgehen, dass der Sprecher das Kooperationsprinzip befolgt. Somit wird impliziert, dass es für die Maximenverletzung gute Gründe gibt. Wenn z. B. der Mitarbeiter der Telekom-Hotline dem Kunden mitteilt: *Unser Mitarbeiter kommt zwischen 8 und 12 Uhr zu Ihnen*, dann wird damit die Maxime der Quantität verletzt – der Beitrag ist weniger informativ, als es für den Gesprächszweck nötig wäre. Wenn man Kooperativität und Befolgung der anderen Maximen unterstellt, ergibt sich die Implikatur, dass eine präzisere Angabe die Maxime der Qualität verletzen würde. Es wird also impliziert, dass der Sprecher den genauen Zeitpunkt, an dem der Servicemitarbeiter den Kunden aufsucht, nicht kennt.

Im letzten Beispiel wurde eine Maxime verletzt, um dadurch eine schwerwiegendere Verletzung einer anderen Maxime zu vermeiden. Es gibt auch Situationen, in denen eine Maximenverletzung durch gewichtigere Anforderungen erklärt ist, die nicht durch die Maximen erfasst sind. Hierzu gehören etwa Anforderungen der Höflichkeit. Wenn man auf die Frage *Wie fandest du den neuen Bond?* antwortet: *Bonds Jacke sah echt gut aus!*, verletzt man die Maximen der Relevanz und der Quantität. Die direktere Antwort *Ich fand ihn ziemlich öde*. würde die Maximen möglicherweise eher erfüllen, könnte aber unhöflich erscheinen. Daraus ergibt sich die Implikatur, dass dem Antwortenden der neue Bond nicht besonders gefallen hat.

Eine besonders gut untersuchte Klasse von konversationellen Implikaturen sind **skalare Implikaturen**. Ein typisches Beispiel hierfür ist die Verwendung von Zahlwörtern und Quantoren, wie in

- (3.179) (a) *Ich habe schon fünfzehn Bond-Filme gesehen.*
 (b) *Ich habe fast alle Bond-Filme gesehen.*

Es gibt gute Gründe für die Annahme, dass (3.179a) auch dann wahr ist, wenn der Sprecher mehr als fünfzehn Bond-Filme gesehen hat. (Man stelle sich z. B. einen Bond-Fanclub vor, in den man nur aufgenommen wird, wenn man fünfzehn Bonds gesehen hat. Jemand, der zwanzig Bonds gesehen hat, hätte natürlich auch Anspruch auf Aufnahme.) Der Satz impliziert jedoch, dass der Sprecher

nicht mehr als fünfzehn Bonds gesehen hat. Das ergibt sich aus folgenden Überlegungen: Wenn der Sprecher sechzehn Bonds gesehen hat, hätte die Maxime der Quantität ihn verpflichtet, das zu sagen. Die anderen Maximen hätten dem nicht entgegengestanden. Daraus folgt, dass er eben keine sechzehn Bonds gesehen hat, eine entsprechende Äußerung also die Qualitätsmaxime verletzt hätte. Durch ein analoges Argument ergibt sich für (3.179b) die Implikatur, dass der Sprecher nicht alle Bond-Filme gesehen hat.

Dieses Inferenzschema wirkt einleuchtend, ist aber nicht ohne Probleme. So könnte man auf analoge Weise argumentieren, dass der Satz *Ich habe einen Bond-Film gesehen* impliziert, dass der Sprecher nicht *James Bond jagt Dr. No* gesehen hat – andernfalls hätte er das aufgrund der Quantitätsmaxime auch sagen müssen. Gleichermassen impliziert der Satz, dass der Sprecher nicht *Liebesgrüße aus Moskau, Goldfinger, …, Casino Royale, Ein Quantum Trost* gesehen hat. Der Satz impliziert also letztendlich seine eigene Negation!

Entscheidend hier ist, dass bei der Berechnung der Implikaturen bestimmte Ausdrucksalternativen in Betracht gezogen werden, die der Sprecher nicht gewählt hat. Wenn zu viele Alternativen herangezogen werden, führt das zu absurdem Ergebnissen. Vielmehr ist es so, dass für jeden Ausdruck nur eine wohldefinierte Menge von geordneten Ausdrucksalternativen, auch **Skala** genannt, eine Rolle spielen (vgl. Horn 1968). Für Determinatoren wie *fast alle* wären das z. B. die Skala *kein, ein, einige, viele, fast alle, alle*. Wenn ein Sprecher sich für ein Element dieser Skala entscheidet, impliziert er damit, dass alle stärkeren Elemente der Skala zu falschen Aussagen führen würden. Für Numeralia wie *fünfzehn* wären dementsprechend alle anderen Numeralia Ausdrucksalternativen.

Das Wissen um die Skalenzugehörigkeit lexikalischer Ausdrücke ist Teil des konventionalisierten semantischen Wissens. Die Trennlinie zwischen konventionalisierter semantischer und aus Rationalitätsüberlegungen deduzierbarer pragmatischer Information ist also nicht wirklich scharf.

Eine weitere wichtige Klasse sind die **klausalen Implikaturen**. Dabei handelt es sich um Quantitäts-Implikaturen, die die Einstellung des Sprechers zum Wahrheitsgehalt von Teilsätzen des geäußerten Satzes betreffen. Betrachten wir ein Beispiel.

(3.180) *Wenn der Zug Verspätung hat, erreichen wir ihn noch.*

Dieser Satz impliziert, dass der Sprecher nicht weiß, ob der Zug Verspätung hat. Wenn er sich nämlich sicher wäre, dass der Zug Verspätung hat, hätte er den kürzeren und informativeren Satz *Wir erreichen den Zug noch* äußern können. Wenn er sich aber sicher wäre, dass der Zug keine Verspätung hat, wäre der Satz als Ganzes eine Verletzung der Relevanz-Maxime.

Grice unterscheidet im weiteren zwischen **partikularisierten** und **generalisierten konversationellen Implikaturen**. Partikularisierte Implikaturen treten nur in speziellen Kontexten auf und hängen von spezifischen Merkmalen dieses Kontexts ab. Die Implikatur des Satzes (3.176b), wonach der Sprecher schon am selben Tag zu Abend gegessen hat, hängt z. B. von der vorangegangenen Frage und der Äußerungssituation ab. Skalare Implikaturen wie die Verstärkung

von *fast alle* zu *fast alle, aber nicht alle* treten nahezu in allen Kontexten auf. Daher handelt es sich um generalisierte Implikaturen.

Im Unterschied zu Implikationen sind konversationelle Implikaturen **anfechtbar**. Wenn man Satz (3.179b) fortsetzt mit *eigentlich sogar alle*, entsteht kein Widerspruch, sondern allenfalls ein Eindruck von Inkohärenz. Ein weiteres wichtiges Merkmal konversationeller Implikaturen ist ihre **Abtrennbarkeit**. Damit ist gemeint, dass zwei synonyme Ausdrücke von vergleichbarer Komplexität die gleichen Implikaturen auslösen. Wenn man z. B. in (3.179b) den Ausdruck *fast* durch *nahezu* ersetzt, entsteht immer noch die Implikatur, dass der Sprecher nicht alle Bond-Filme gesehen hat.

Die Grundideen des Griceschen Programms sind in der modernen Pragmatikforschung weitgehend unkontrovers. Wichtig ist vor allem die von Grice eingeführte Unterscheidung zwischen dem **Gesagten**, also der wörtlichen Bedeutung eines Satzes, und dem **Gemeinten**, also der Information, die tatsächlich absichtsvoll kommuniziert wird. Es wird auch weithin akzeptiert, dass sich das Gemeinte systematisch aus dem Gesagten, kontextueller Information sowie den Prinzipien rationaler Kommunikation berechnen lässt.

Über die genaue Ausgestaltung dieses Programm hat sich zum gegenwärtigen Zeitpunkt noch kein Konsens herausgebildet. Man kann grob zwei Denkrichtungen unterscheiden. Auf der einen Seite gibt es die neo-Gricesche Schule, zu deren wichtigsten Vertretern Stephen Levinson gehört (siehe z. B. Levinson 2000). Er ersetzt die Griceschen Maximen durch drei Prinzipien (das Q-, I-, und M-Prinzip), die gemeinsam in etwa die Effekte der Quantitäts-, Qualitäts-, und Modalitätsmaxime abdecken. Dem steht die relevanztheoretische Schule um Dan Sperber, Deirdre Wilson und Robyn Carston gegenüber, die pragmatische Inferenzen ausschließlich aus einer verallgemeinerten Variante der Relevanz-Maxime abzuleiten versuchen (vgl. Sperber und Wilson 1986; Carston und Uchida 1998).

Ein besonders problematischer Aspekt des ursprünglichen Griceschen Programms ist die Annahme, dass das Gemeinte erst berechnet werden kann, wenn das Gesagte bekannt ist. Die Pragmatik kann sozusagen erst anfangen, wenn die Semantik mit ihrer Arbeit fertig ist. Weiterhin wird Semantik bei Grice mit Wahrheitsbedingungen identifiziert. Es ist aber so, dass schon in die Wahrheitsbedingungen eines Satzes pragmatische Information einfließt. So lässt sich argumentieren, dass es in Satz (3.176b) tatsächlich Teil des Gesagten ist (bzw. vor der Einbeziehung von Satz (3.176c) zu sein scheint), dass der Sprecher am selben Tag schon zu Abend gegessen hat. Dafür spricht, dass derartige Information im Skopus von wahrheitskonditionalen Operatoren stehen kann:

- (3.181)
- (a) *Ich habe noch nicht gegessen.*
 - (b) *Hans hat wahrscheinlich gegessen.*
 - (c) *Wenn Hans schon gegessen hat, können wir eigentlich anfangen.*

In allen drei Fällen wird die Information, dass der Sprecher bzw. Hans kurz vorher gegessen hat, in der selben Skopussposition verrechnet wie die kompositionale determinierten Bedeutungsaspekte des jeweiligen Teilsatzes. In (3.181a)

steht diese Information im Skopuss der Negation, in (b) im Skopuss des Modalwortes *wahrscheinlich*, und in (c) im Skopuss des Konditional-Operators.

In der relevanztheoretischen Tradition nennt man pragmatisch determinierte Aspekte des Gesagten **Explikaturen**. In der neo-Griceschen Tradition spricht man bei Interaktion derartiger Informationen mit skopalen Elementen von **eingebetteten Implikaturen** (die es nach Grice eigentlich nicht geben dürfte). Interessanterweise ist es so, dass nicht alle Implikaturen bei der Einbettung eines Satzes unter einen skopalen Operator erhalten bleiben.

- (3.182) (a) A: *Hat Hans aufgehört zu rauchen?*
- (b) B: *Er trägt Nikotinpflaster.*
- (b') B: *Er trägt kein Nikotinpflaster.*

Der Satz (3.182b) impliziert, dass Hans aufgehört hat zu rauchen. Wenn man den Satz negiert wie in (b'), wird diese Implikatur jedoch nicht mitnegiert; sie wird in diesem Fall gar nicht generiert.

Diese Beispiele sollen illustrieren, dass die Berechnung pragmatischer Inferenzen aus der kompositionally bestimmten konventionalisierten Bedeutung, kontextueller Information und den Prinzipien rationaler Kommunikation auf komplexe Art mit der kompositionalen Bedeutungsberechnung selbst interagiert. Die genaue Natur dieses Zusammenspiels ist derzeit ein wichtiger Untersuchungsgegenstand. Nicht zuletzt sind hier in naher Zukunft wichtige Aufschlüsse aus psycholinguistischen Befunden und ihrer computationellen Modellierung zu erwarten.

Präsuppositionen

Intuitiv gesprochen sind die Präsuppositionen eines Satzes die impliziten Hintergrundannahmen, die mit einer Äußerung dieses Satzes üblicherweise verbunden sind. Sie sind häufig konventionell mit bestimmten sprachlichen Mitteln verknüpft und lassen sich kompositionally berechnen. Diese Merkmale teilen sie mit semantischen Inhalten im engeren Sinne. Andererseits interagieren Präsuppositionen auf komplexe Weise mit dem Redehintergrund und mit den pragmatischen Angemessenheitsbedingungen einer Äußerung, so dass sie letztendlich dem Gebiet der Pragmatik (und nicht der Semantik) zuzurechnen sind.

Wie auch Implikaturen sind Präsuppositionen eine bestimmte Art von Folgerungen. Im Unterschied sowohl zu gewöhnlichen Implikationen als auch zu Implikaturen können Präsuppositionen eines eingebetteten Satzes jedoch an den Matrixsatz vererbt werden. Das sei anhand des folgenden Beispiels illustriert. (3.183a) impliziert sowohl (b) als auch (c), aber nur (b) wird präsponniert.

- (3.183) (a) *Hans verschüttet wieder den Kaffee.*
- (b) *Hans hat schon einmal den Kaffee verschüttet.*
- (c) *Hans verschüttet etwas.*

In (3.184) erscheint (3.183a) eingebettet in verschiedene syntaktische Kontexte. Die Folgerung (3.183b) bleibt in allen Fällen erhalten, (3.183c) jedoch nicht.

- (3.184) (a) *Es stimmt nicht, dass Hans wieder den Kaffee verschüttet.*
 (b) *Velleicht verschüttet Hans wieder den Kaffee.*
 (c) *Wenn Hans wieder den Kaffee verschüttet, kriegt er Ärger.*

Die Vererbung von Präsuppositionen von eingebetteten auf Matrix-Konstruktionen wird **Präsuppositionssprojektion** genannt.

Die meisten syntaktischen Kontexte sind durchlässig für Präsuppositionssprojektion. Karttunen (1973) bezeichnet derartige Kontexte als **Löcher** (engl. *holes*). Daneben gibt es eine Reihe von satzeinbettenden Verben, die undurchlässig für Projektion sind (in Karttunens Terminologie **Stöpsel**, engl. *plugs*). In diese Kategorie fallen *verba dicendi* wie *sagen*, *behaupten*, *erwähnen* usw. Aus *Hans sagt, dass er wieder den Weihnachtsmann getroffen hat* etwa folgt nicht, dass Hans schon einmal den Weihnachtsmann getroffen hat.

Am interessantesten sind die Kontexte aus der dritten Kategorie, Karttunens **Filter** (engl. *filters*). Hiermit bezeichnet er Kontexte, die selektiv manche Präsuppositionen projizieren, andere jedoch nicht. Beispielsweise präsponiert (3.185a) in Isolation sowohl, dass gerade jemand den Kaffee verschüttet, als auch, dass Hans schon einmal den Kaffee verschüttet hat. Nur die erste dieser beiden Präsuppositionen wird jedoch in (b), (c) und (d) auf den Gesamtsatz projiziert.

- (3.185) (a) *Es ist wieder Hans, der gerade den Kaffee verschüttet.*
 (b) *Hans hat das letzte Mal den Kaffee verschüttet, und es ist wahrscheinlich wieder Hans, der gerade den Kaffee verschüttet.*
 (c) *Wenn Hans das letzte Mal den Kaffee verschüttet hat, dann ist es wahrscheinlich wieder Hans, der gerade den Kaffee verschüttet.*
 (d) *Entweder verschüttet Hans nie etwas, oder es ist wieder Hans, der gerade den Kaffee verschüttet.*

Ein Spezialfall der Filterung ist die **Anfechtung** von Präsuppositionen. Der Satz (3.186) z. B. präsponiert, dass Hans raucht oder einmal rauchen wird. In (b) wird diese Präsupposition explizit verneint. Das führt nicht, wie zu erwarten wäre, zu einem Widerspruch, sondern blockiert lediglich die Projektion der Präsupposition auf den Gesamtsatz.⁵

- (3.186) (a) *Hans wird nie aufhören zu rauchen.*
 (b) *Hans wird nie aufhören zu rauchen, denn er raucht gar nicht und wird auch nicht damit anfangen.*

Man kann zwei Arten von Filtern unterscheiden. In (3.185b) folgt die gefilterte Präsupposition des zweiten Konjunkts aus der Semantik des ersten Konjunkts. Analog dazu lässt sich in (3.185c) die gefilterte Präsupposition des *dann*-Satzes aus dem *wenn*-Satz folgern. In Analogie zur Anaphorik (die im Unterkapitel

⁵In dieser Hinsicht scheinen sich Präsuppositionen ähnlich zu verhalten wie konversationelle Implikaturen. Man beachte aber, dass sich nur projizierte Präsuppositionen anfechten lassen, während konversationelle Implikaturen gar nicht projiziert werden.

3.7.2 genauer diskutiert wird) kann man davon sprechen, dass in diesen Konfigurationen die gefilterten Präsuppositionen **gebunden** werden. Bemerkenswert ist, dass Bindung asymmetrisch ist – wenn in einer Konjunktion die Präsupposition des ersten Konjunkts aus dem zweiten Konjunkt folgt, entsteht allenfalls der Eindruck von Redundanz, aber die Projektion wird nicht blockiert. Das gleiche gilt sinngemäß für Konditionalsätze.

Für Bindung ist es ausreichend, dass die Präsupposition des zweiten Konjunkts bzw. des *dann*-Satzes **kontextuell** aus dem ersten Konjunkt (bzw. dem *wenn*-Satz) geschlussfolgert werden kann. Karttunen (1973) erläutert diesen Sachverhalt sinngemäß mit folgendem Beispiel. In (3.187) besteht keine semantische Beziehung zwischen dem jeweiligen ersten Teilsatz und der Präsupposition des zweiten Teilsatzes, wonach Geraldine einmal heilige Unterwäsche getragen hat. In einem normalen Kontext würde diese Präsupposition also projiziert. Nehmen wir aber an, dass es Teil des Redehintergrundes ist, dass Mormonen bis zu einem gewissen Alter heilige Unterwäsche tragen. Dann lässt sich aus der Tatsache, dass Geraldine Mormonin ist, sehr wohl schlussfolgern, dass sie einmal heilige Unterwäsche getragen hat. Diese kontextuelle Folgerung reicht aus, um die Präsupposition zu filtern.

- (3.187) (a) *Geraldine ist Mormonin, und sie hat aufgehört, ihre heilige Unterwäsche zu tragen.*
- (b) *Wenn Geraldine Mormonin ist, hat sie aufgehört, ihre heilige Unterwäsche zu tragen.*

In (3.185d) wird die Präsupposition des zweiten Disjunkts zwar gefiltert, aber nicht gebunden. Entscheidend ist hier, dass eine Disjunktion die klausale Implikatur auslöst, dass der Sprecher sich weder über den Wahrheitsgehalts des ersten noch des zweiten Disjunkts sicher ist. Wenn die Präsupposition, wonach Hans schon einmal Kaffee verschüttet hat, projiziert würde, würde sich der Sprecher aber auf ihre Wahrheit und damit auf die Falschheit des ersten Disjunkts festlegen. Bei einem solchen Konflikt zwischen der Präsupposition eines eingebetteten Satzes und den konversationellen Implikaturen des Matrixsatzes wird die Präsuppositionsprojektion blockiert. Präsuppositionsanfechtung ist ein Spezialfall dieses Mechanismus – (3.186b) löst die Implikatur aus, dass der Sprecher glaubt, dass Hans nicht raucht und auch nie rauchen wird. Damit ist die potentielle Präsupposition, wonach Hans raucht oder einmal rauchen wird, nicht verträglich, und sie wird deshalb nicht projiziert.

Um zu testen, ob eine bestimmte Folgerung eines Satzes eine Präsupposition ist, muss man überprüfen, ob sie (a) aus eingebetteten Kontexten projiziert wird, und wenn ja, ob (b) die Projektion in den genannten Filterkonfigurationen blockiert wird. Der verbreitetste (aber allein nicht völlig zuverlässige) **Präsuppositionstest** ist der **Negationstest**. Dabei wird geprüft, ob ein Präsuppositionskandidat aus einem negierten Kontext projiziert wird. Z. B. ergibt der Negationstest (zutreffenderweise), dass (3.183b) von (3.183a) präsupponiert wird, da (3.183b) sowohl aus (3.183a) als auch aus (3.184a) folgt.

Präsuppositionen sind **konventionell** mit bestimmten grammatischen oder lexikalischen Mitteln verbunden, den **Präsuppositionsauslösern** (engl. *pre-*

supposition triggers). Diese bilden morpho-syntaktisch eine heterogene Klasse. Zu erwähnen wären **definite Deskriptionen** (die NP *der König von Frankreich* löst die Präsposition aus, dass Frankreich genau einen König hat), **faktive Verben** wie *wissen* oder *bedauern* (sie lösen die Präsposition aus, dass ihr Komplementsatz wahr ist), **Phasenübergangsverben** wie *anfangen*, *aufhören* (siehe die Diskussion zu Beispiel (3.186)) und manche **Quantoren** (z. B. löst *alle Delegierte* die Präsposition aus, dass es Delegierte gibt). Nicht zuletzt gibt es auch bestimmte syntaktische Konstruktionen, die Präspositionen auslösen, beispielsweise **Spaltsätze** (vgl. 3.188a) und **Pseudo-Spaltsätze** (wie in 3.188b). Beide Sätze in (3.188) präsupponieren, dass jemand eine Banane gegessen hat.

- (3.188) (a) *Es war Hans, der eine Banane gegessen hat.*
 (b) *Wer eine Banane gegessen hat, war Hans.*

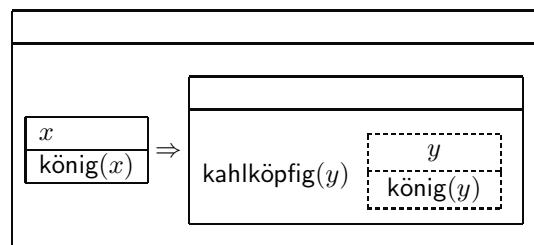
Diese Liste ist bei weitem nicht vollständig.

Präspositionen in der DRT In diesem Abschnitt soll die von van der Sandt (1992) entwickelte Formalisierung der Präspositionsprojektion kurz skizziert werden. Van der Sandt führt gute Argumente dafür an, dass Präspositions- und Anapherninterpretation zwei Aspekte desselben Phänomens sind und deshalb auch technisch einheitlich behandelt werden können. Da die Diskursrepräsentationstheorie (DRT; siehe Unterkapitel 3.6) über einen ausgedehnten Apparat zur Anapherninterpretation verfügt, liegt eine Übertragung auf Präspositionspheomene nahe.

Präspositionen werden als subordinierte DRSen formalisiert. Der modifizierte DRS-Konstruktionsalgorithmus bildet einen Satz zunächst auf eine DRS ab, in der jede Präsposition auf derselben Einbettungsebene erscheint wie ihr Auslöser. Das sei anhand des folgenden Beispiels illustriert.

- (3.189) *Wenn es einen König gibt, dann ist der König kahlköpfig.*

Der Präspositionsauslöser *der König* steht im *dann*-Satz eines Konditionalsatzes. Das korrespondiert zum zweiten Argument der Implikation in der DRS. Deshalb erscheint die zu der ausgelösten Präsposition (wonach es einen König gibt – die Einzigkeitspräsposition wird der Einfachheit halber hier ignoriert) korrespondierende Sub-DRS eingebettet in diese Teil-DRS. Präsponierte DRSen werden hier durch gepunktete Umrandung gekennzeichnet, um sie von normalen Sub-DRSen zu unterscheiden.



Präsupponierte DRSen sind zunächst uninterpretierbar. Sie müssen in einem weiteren Verarbeitungsschritt **resolviert** werden. Die präferierte Methode hierfür ist **Bindung** (im technischen Sinne). Hierfür wird

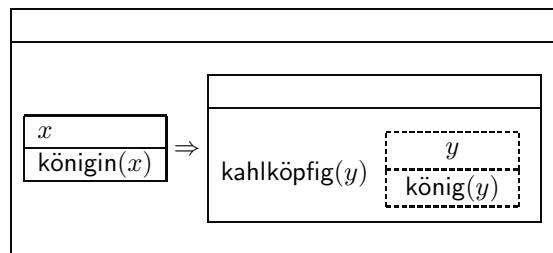
1. eine für die zu resolvierende DRS K **zugängliche** (siehe Definition 3.6.5 in Unterkapitel 3.6) DRS K' ausgewählt,
2. das Universum von K wird durch eine ein-eindeutige Abbildung f auf das Universum von K' abgebildet und alle Vorkommen von Diskursreferenten d aus dem Universum von K werden nach $f(d)$ umbenannt, und
3. die Konditionen von K werden nach der Umbenennung zu K' hinzugefügt, und K selbst wird getilgt.

Im laufenden Beispiel ist das erste Argument der Implikation für die Präsupposition zugänglich, deshalb kann gebunden werden, indem y auf x abgebildet wird. Als Resultat erhalten wir eine DRS, in der die Präsupposition verschwunden ist und die Kondition im zweiten Teil der Implikation zu $kahlköpfig(x)$ umbenannt wurde. Anaphernbindung kann als ein Spezialfall hiervon aufgefasst werden – die präsupponierte DRS enthält hier nur einen Diskursreferenten und keine Konditionen.

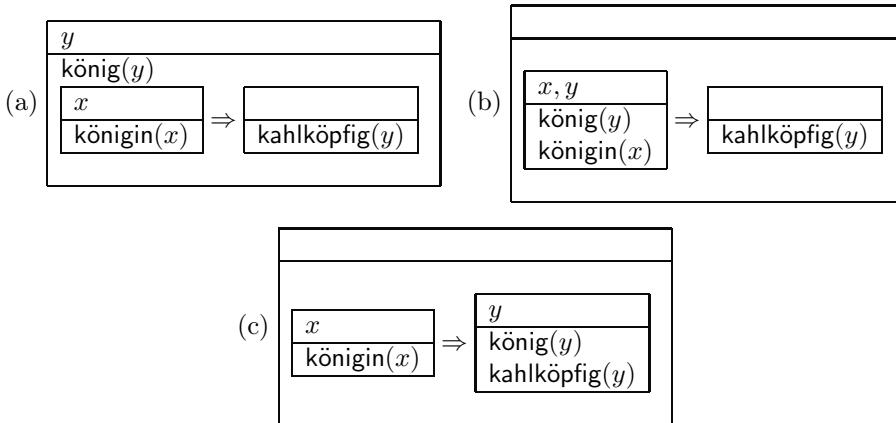
Wenn Bindung nicht möglich ist, können präsupponierte DRSen u.U. einfach ohne Umbenennung der Diskursreferenten einer zugänglichen DRS hinzugefügt werden. Diese Operation heißt **Akkommodation**. Zur Illustration ändern wir das letzte Beispiel leicht ab:

(3.190) *Wenn es eine Königin gibt, dann ist der König kahlköpfig.*

Dem würde zunächst die folgende DRS entsprechen:



Bindung würde zu einer inkonsistenten DRS führen, da niemand gleichzeitig König und Königin sein kann. Deshalb wird akkommodiert. Für die Präsupposition sind alle anderen Sub-DRSen zugänglich, die Matrix-DRS sowie die beiden Argumente der Implikation. Deshalb gibt es drei Optionen für Akkommodation – die Präsupposition wird zu einer der zugänglichen DRSen hinzugefügt. Technisch bedeutet das, dass sowohl das Universum als auch die Konditionen des Akkommodationsziels mit der entsprechenden Komponente der präsupponierten DRS mengentheoretisch vereinigt werden. Als Resultate erhalten wir



Die natürlichsprachlichen Paraphrasen dieser drei potentiellen Lesarten sind:

- (3.191) (a) *Es gibt einen König, und wenn es eine Königin gibt, dann ist er kahlköpfig.* (globale Akkommmodation)
- (b) *Wenn es einen König und eine Königin gibt, dann ist der König kahlköpfig.* (intermediäre Akkommmodation)
- (c) *Wenn es eine Königin gibt, dann gibt es einen König, und er ist kahlköpfig.* (lokale Akkommmodation)

Wenn wie hier mehrere Optionen für Akkommmodation existieren, gilt die Präferenzordnung „Akkommodiere so hoch wie möglich!“ (im Sinne der Zugänglichkeitsrelation, also K_1 ist höher als K_2 gdw. K_1 für K_2 zugänglich ist). Demnach ist hier Akkommmodation in die Matrix-DRS (entspricht (3.191a)) die präferierte Option, die somit als einzige Lesart von (3.190) vorausgesagt wird. Es gibt eine ganze Reihe von Beschränkungen für Resolution, die hier nur stichpunktartig erwähnt werden können:

1. Jede Sub-DRS ist in ihrem lokalen Kontext konsistent und informativ. (Für die Matrix-DRS ist der lokale Kontext der Redehintergrund.)
2. Jeder Diskursreferent, der in einer DRS-Kondition vorkommt, kommt auch in einem zugänglichen Universum vor (ist also gebunden).
3. Bindung ist besser als Akkommmodation.
4. Hohe Akkommmodation ist besser als tiefe Akkommmodation.

Präsuppositionenprojektion wird in diesem System als Akkommmodation in die Matrix-DRS rekonstruiert. Filterung tritt auf, wenn globale Akkommmodation nicht die präferierte Resolutionsstrategie ist – sei es, dass Bindung möglich und damit präferiert ist, sei es, dass eine der genannten pragmatischen Beschränkungen verletzt würde. Beispielsweise würde globale Akkommmodation der Präsupposition in (3.185d) dazu führen, dass das erste Disjunkt uninformativ wird.

Deshalb ist hier nur lokale Akkommodation möglich, es findet also keine Projektion statt. In (3.186b) hingegen würde globale Akkommodation den zweiten Teilsatz inkonsistent machen, was ebenfalls ausgeschlossen ist.

Literaturhinweise

Wie im Text erwähnt, ist der Standardtext zum Implikaturbegriff Grice (1975). Eine lehrbuchartige Darstellung findet sich in Levinson (1983). Zur aktuellen Debatte zwischen neo-Gricescher und relevanztheoretischer Auffassung siehe Carston und Uchida (1998), Levinson (2000) sowie Recanati (2004).

Zum Thema Präspositionen wären neben den genannten Arbeiten von Karttunen und van der Sandt Gazdar (1979) Karttunen (1974), Stalnaker (1973) und Stalnaker (1974) sowie Heim (1990) als wichtige primäre Quellen zu erwähnen. In Beaver (1997) findet sich ein umfassender Überblick sowohl über den Phänomenenbereich als auch über den Stand der Theoriediskussion. Geurts (1999) ist eine gut lesbare ausführliche Darstellung und Weiterentwicklung von van der Sandts Theorie. Blackburn und Bos (1999) befasst sich mit ihrer Implementierung in Prolog.

3.7.4 Benutzermodellierung

John Bateman und Cécile Paris

Alle Systeme, die mit Menschen interagieren, haben irgendeine Vorstellung von ihrem Benutzer. Dies ist notwendig, weil man nicht davon ausgehen kann, dass sich Benutzer hinsichtlich ihrer Erwartungen, ihres Wissensstandes, ihrer Interessen und Fähigkeiten gleichen. Wissen über den Benutzer kann dazu dienen, besser zu bestimmen, welche Informationen das System anbieten sollte, wie es das tun sollte, wie der Interaktionsstil gestaltet werden sollte, usw. In vielen Systemen ist diese Vorstellung von dem Benutzer jedoch implizit. Der Systemautor hatte hier einen bestimmten Benutzer im Sinn. Wenn dagegen ein System explizit diese Vorstellung repräsentiert und dadurch sein Verhalten dem Benutzer anpasst, spricht man von **Benutzermodellierung** (engl. *user modelling*). Benutzermodellierung bezeichnet die Methoden, die interaktive Software-Systeme in die Lage versetzen, ihr Verhalten an ihren jeweiligen Benutzer anzupassen durch Erstellung und Ausnutzung eines **Benutzermodells**, das die Eigenschaften des Benutzers beinhaltet. Mit Hilfe von Benutzermodellierung sollen Computersysteme benutzerfreundlicher werden, wodurch andererseits wiederum die Benutzer ihre Ziele besser erreichen können.

Ein Großteil der frühen Arbeiten zur Benutzermodellierung fand innerhalb der Computerlinguistik statt. Dafür gibt es verschiedene Gründe. Zuerst einmal hat sich herausgestellt, dass Benutzer, die Fragen in natürlicher Sprache an ein System stellen, dazu neigen, dem System menschliche Eigenschaften zuzuschreiben und erwarten, dass das System in derselben Weise antwortet, wie ein menschliches Wesen es tun würde. Dazu kommt, dass zwischenmenschliche Kommunikation schon immer als ein geeignetes Modell für die Mensch-Maschine-Kommunikation angesehen wurde (z. B. Winograd und Flores 1986). Es ist offensichtlich, dass Menschen in gegenseitiger Kommunikation normalerweise ein Modell ihres Hörers (beim Sprechen) oder ihres Lesers (beim Schreiben) benutzen. Deshalb schien es wünschenswert, Verfahren zur Nachahmung dieses beobachteten menschlichen Kommunikationsverhaltens in Computersystemen einzubauen. Wenn die Ausgabe des Computersystems natürlichsprachlichen geschriebenen oder gesprochenen Text enthält, wird der Zuschnitt solcher Texte auf das Benutzermodell hin auch als **Tailoring** bezeichnet.

Heutzutage ist Benutzeranpassung allgemein üblich. Die Anpassung von Dokumenten an den Leser ist inzwischen in den meisten Web- und Mobilfunkdienstleistungen eingebaut. Nutzer werden aufgefordert, bestimmte Sachgebiete anzugeben, die bei der Informationspräsentation, die sie bekommen, besonders berücksichtigt werden sollen. Die Inferenzen für diese Benutzeranpassung können mehr oder weniger komplex sein. Im einfachsten Fall nennt ein Benutzer seine Präferenzen direkt, und das System filtert die Information mit Hilfe von Word-Matching. Auch der web-basierte Buchversand Amazon zeigt Werbung für Bücher, von denen das System annimmt, dass sie der jeweilige Nutzer mit höherer Wahrscheinlichkeit kaufen wird. Solche Anpassung oder **Individuali-**

sierung wird immer häufiger verwendet. Die Arbeit an Benutzermodellierung ist deshalb heute weit über die Computerlinguistik hinausgewachsen und wird in anderen Disziplinen wie z. B. Informationretrieval, adaptive multimodale Web-Präsentation oder mobile Touristenführungssysteme fortgeführt.

Innerhalb der Computerlinguistik verwenden jetzt die meisten Systeme irgend ein Benutzermodell. Auch wenn die Benutzeranpassung kein Hauptaugenmerk eines Systems ist, wird trotzdem das Systemverhalten durch Einsatz eines Benutzermodells beeinflusst. Außerdem haben mit der Hinwendung der Computerlinguistik zur Multimedia- und Hypertext-Generierung Systeme bereits begonnen, Benutzermodelle zur Entscheidung über das zu verwendende Medium, die zu platzierenden Links und die anzubietenden Navigationshilfen heranzuziehen.

Ziele der Benutzermodellierung

Die Grundannahmen der Benutzermodellierung sind:

1. dass ein System Wissen über seinen Benutzer – sei es ein Einzelindividuum oder ein Repräsentant einer typischen Gruppe – ausnutzen kann, um sein Verhalten an den Benutzer anzupassen, und
2. dass eine solche Anpassung für den Benutzer von Vorteil ist.

Diese Annahmen haben eine Vielzahl von benutzeradaptiven Verhaltensweisen in Computersystemen hervorgebracht. Verschiedene Aspekte eines Systems und seines Verhaltens können benutzerabhängig gestaltet werden: der Inhalt der präsentierten Information kann mehr oder weniger benutzerspezifisch gemacht werden, der Interaktionsstil des Systems kann angepasst werden, und das Interface selbst kann variieren. Manche Systeme kombinieren mehrere dieser Möglichkeiten. In jedem Fall ist es jedoch wichtig, dass die Vorteile für den Benutzer empirisch nachprüfbar sind.

Anpassung des Inhalts an den Benutzer: Ein wissensbasiertes System hat eine große Menge von Informationen zur Verfügung. Es ist unmöglich, all diese Informationen einfach dem Benutzer zu präsentieren. Zu einem bestimmten Zeitpunkt ist ein Großteil der Informationen für den Benutzer mindestens irrelevant, wenn nicht sogar unverständlich – einmal weil sie zu komplex sind oder weil sie in einem Abstraktionsgrad gegeben werden, der dem Benutzer nicht entspricht.

Frühere Beispiele für dieses Phänomen finden sich in Erklärungen, die **Expertensysteme** ausgeben. Man hatte erkannt, dass Benutzer eher den Aussagen von Expertensystemen vertrauen, wenn das System die Schlussfolgerungen erklären kann, die zu einer Aussage geführt haben. Aber die Generierung solcher Erklärungen wurde zum Problem. Wenn Expertensysteme einfach die Schlusskette verbalisierten, wurden die Erklärungen nicht verstanden, da sie sich auf einer für den Endnutzer falschen Abstraktionsebene befanden und oft auch für den Nutzer irrelevante Informationen enthielten, z. B. Details über Regeln zur

programminternen Effektivität. Ein aktuelleres Beispiel ist die Informationsbeschaffung durch das Web. Zu oft enthält die Antwort auf eine Suchanfrage viele Links, die für das Anliegen des Benutzers irrelevant sind. Wenn das System in der Lage wäre, Wissen über den Benutzer anzusammeln, damit nur die für den Benutzer brauchbaren Informationen präsentiert werden, würde das die Nützlichkeit eines Informationretrieval-Systems sehr verbessern.

Es gibt zwei wichtige Arten von Wissen für die Anpassung des Inhalts an einen Benutzer: die möglichen Ziele des Benutzers und sein Wissensstand. Um das erstere zu veranschaulichen, betrachten wir ein Szenario in einem **Auskunfts- system** zum öffentlichen Verkehr, das von einem Benutzer gefragt wird: *Fährt die Fähre zur Stadt am Sonntagmorgen zur gleichen Zeit wie werktags?* Falls überhaupt keine Fähren am Sonntag verkehren, wäre die direkte Antwort *Nein*. Eine solche Antwort würde jedoch mindestens als unkooperativ, wenn nicht gar als irreführend empfunden. Ein System, das eine solche Frage als indirekte Frage erkennen und schlussfolgern kann, dass der Benutzer vorhat, am Sonntag in die Stadt überzusetzen, wäre in der Lage, eine kooperativere Antwort zu geben wie z. B.:

- (3.192) *Die Fähren zur Stadt verkehren am Sonntag nicht. Sie können aber einen Bus nehmen.*

Ein solches Verhalten kann jedoch nicht unabhängig von Kenntnissen über den jeweiligen Benutzer vorprogrammiert werden. Es handelt sich nicht einfach um mehr Information, wie die folgenden alternativen Antworten aus einem Dialog mit einem automatischen Buchungssystem (Morik 1989, S. 380) verdeutlichen:

- (3.193) BENUTZER: *Hat das Hotel eine Nachtbar?*
 (a) SYSTEM: *Nein, das Hotel ist ruhig.*
 (b) SYSTEM: *Nein, aber es gibt eine Nachtbar in der Nähe.*

Beide Antworten geben zusätzliche Informationen. Sie **überbeantworten** die Frage, ein für kooperative Agenten typisches Verhalten. Aber welche Zusatzinformation gegeben wird, hängt entscheidend von den Annahmen über den Benutzer, d.h. vom Benutzermodell, ab. Hier zeigt sich, dass die Verwendung eines Benutzermodells, das es erlaubt, die **Intentionen** des Benutzers zu erkennen, ein System kooperativer machen kann.

Wissen über den Wissensstand des Benutzers kann gleichermaßen für das Informationsangebot eines Systems wichtig sein. Automatisch generierte Vergleiche bieten z. B. ein sinnvolles und komplexes Mittel zur Informationspräsentation an, aber ihre Informativität hängt entscheidend vom **Vorwissen** des Benutzers ab. Die zum Vergleich herangezogenen Objekte müssen dem Benutzer bekannt sein, wie aus dem folgendem Kontrast ersichtlich wird:

- (3.194) (a) *Wie der Igel hat das Stachelschwein viele Stacheln.*
 (b) *Wie das Stachelschwein hat der Igel viele Stacheln.*

Für den Benutzer, der sich mit Igeln auskennt, ist die Variante (a) informativer, für einen Benutzer, der Stachelschweine besser kennt, Variante (b). Wenn ein

Benutzer keine dieser beiden Tiere kennt, dann sind beide Alternativen nicht adäquat. Wenn Information bzgl. des Vergleichsobjektes bereits früher im Text gegeben worden ist, ändert sich die Situation genauso. Um solchen Fällen gerecht zu werden, muss sich die Benutzermodellierung mit dem Konzept des **Diskursmodells** (s. Unterkapitel 3.8 sowie Abschnitt 3.7.1) auseinandersetzen. Aus dem sich entfaltenden Text gewonnene Information kann wie Vorwissen im Benutzermodell gespeichert werden und danach mit in den Präsentationsentscheidungsprozess einfließen.

Diese zwei Aspekte der Benutzermodellierung – Erkennen der Intention des Benutzers und der Wissensstand dieses Benutzers – gehören zu den ersten Problemen, die die Benutzermodellierungsforschung in der Computerlinguistik behandelt hat, weil beide Input für effektive Text- und Dialogproduktion liefern. Heutige für die Computerlinguistik relevante Forschung in der Benutzermodellierung behandelt immer noch vorwiegend diese beiden Aspekte, wird jedoch stets erweitert, um mit neuen Aspekten der Informationspräsentation Schritt zu halten.

Anpassung des Navigationsstils: Aufgrund der Eigenschaften von Hypertexten können Multimediapräsentationen sehr effektiv sein, aber es ist auch leichter für den Benutzer, die Orientierung zu verlieren. Daher wird besonders bei Multimediapräsentationen die Frage der Navigation wichtig. Hier kann eine Benutzermodellierung eingesetzt werden, um gezielt eine auf den Benutzer zugeschnittene Navigationshilfe anzubieten. Weil ein Benutzermodell Kenntnis über den Wissensstand des Benutzers hat, kann das Modell in zweierlei Hinsicht ausgenutzt werden. Einmal wieder hinsichtlich der Auswahl der zu präsentierenden Informationen und zum zweiten hinsichtlich dessen, welche dieser Informationen direkt und welche als Hyperlink angeboten werden.

Benutzermodelle finden auch Verwendung für die Anpassung von Benutzeroberflächen in der Mensch-Maschine-Kommunikation (engl. *Human Computer Interaction: HCI*). In einigen Systemen werden z. B. Menüs in Abhängigkeit vom Benutzerverhalten neu arrangiert: Funktionen, die der Benutzer am häufigsten aufruft, gelangen an die Spitze der Menüliste, während selten benutzte Funktionen an das Ende der Liste zurückfallen. In anderen Fällen gibt das Interface zusätzliche Hilfen für neue Benutzer. Heute werden Benutzermodelle auch dazu verwendet, um Nutzern von Web-Interfaces die geeignete Interaktion zu offerieren. Z. B. kann ein System auf verschiedene Abkürzungen und Links verweisen, die aus beobachteten typischen Navigationsabläufen gewonnen wurden.

Anpassung des Präsentationsstils an den Benutzer: Besonders relevant sind in diesem Bereich derzeit Fragen bzgl. der dem Benutzer angepassten Multimodalität. Hier wird das äußere Erscheinungsbild einer Präsentation durch die Wahl eines passenden Mediums (z. B. Text versus Diagramm) auf der Basis der in dem Benutzermodell eingetragenen Interpretations- und Wahrnehmungsfähigkeiten des Benutzers bestimmt. Solche Verfahren überschneiden sich mit denen für adaptive Mensch-Machine-Schnittstellen sowie mit denen, die eine adäquate

Berücksichtigung von **e-Accessibility** und **e-Inclusion** anstreben. Wenn z. B. bekannt ist, dass der Benutzer schlecht sehen kann, bietet sich eine automatische Anpassung der Schriftgröße oder ein völliger Verzicht auf Text an.

Anpassung der Sprache an den Benutzer: Die Form der Sprache, die ein Informationssystem zur Interaktion auswählt, kann auch in sinnvoller Weise an den Benutzer angepasst werden. In multilingualen Systemen finden grobe Stereotype über die vom Benutzer bevorzugte Sprache Verwendung. Auf einer feineren Ebene kann Wissen über den Wissensstand oder die Art des Benutzers verwendet werden, um Wortwahl und Stil zu steuern. Dies reicht von grober Subsprachenauswahl – z. B. differiert die Sprache eines Schifffahrts-Wetterberichts von der in den Tagesnachrichten – bis hin zu kleinen Variationen in Abhängigkeit vom Erfahrungshintergrund und den Interessen des Benutzers.

In einer frühen Studie zu letzterem Gebiet zeigen Bateman und Paris (1989) und Paris (1993, S. 170)), wie eine Textgenerierungskomponente (s. Unterkapitel 3.8) konstruiert werden kann, die die folgenden alternativen Paraphrasierungen aus ein- und derselben Wissensbasis in Abhängigkeit vom Benutzermodell generieren kann:

Systementwickler: *The system is faulty, if there exists a O in the set of output terminals of the system such that the expected value of the signal part of O does not equal the actual value of the signal part of O and for all I in the set of the input terminals of the system, the expected value of the signal part of I equals the actual value of the signal part of I .*

Fortgeschrittener Benutzer: *The system is faulty, if all of the expected values of its input terminals equal their actual values and the expected value of one of its output terminals does not equal its actual value.*

Naiver Benutzer: *The system is faulty, if its inputs are fine and its output is wrong.*

Die erste Variante ist eine direkte Darstellung der zugrunde liegenden Regel aus dem Expertensystem. Die beiden anderen Varianten sind von dem spezifizierten Grad der Expertise des Benutzers abhängige Reformulierungen dieser Regel. Für das Design dieses Systems bauten Bateman und Paris auf Ergebnisse aus der funktionalen Linguistik auf, insbesondere auf dem Begriff des **Registers**. Die Registertheorie besagt, dass eine systematisch darstellbare Abhängigkeit zwischen Eigenschaften der benutzten Sprache und Eigenschaften des Kontexts besteht. Die hergestellten Reformulierungen folgen daher einem allgemeinen Mechanismus, in dem kontextabhängige Variation systematisch entsteht (Biber 1988; Bateman und Paris 1991). Dieser Zusammenhang zwischen Sprache und Kontext bietet immer noch ein solides Fundament für die Definition von Benutzermodellierung an sich. Die Registertheorie hat mit zunehmender Verfügbarkeit großer, elektronischer Korpora in den letzten Jahren immer mehr an Bedeutung gewonnen. Für die Zukunft sind noch wesentlich feinere Ergebnisse zur Beziehung zwischen Sprache und Situation zu erwarten.

Weitere Prototypen in der Textgenerierungen haben sich in letzter Zeit auch mit der Anpassung der Sprache an die *Sprachfähigkeiten* der intendierten Leser auseinandergesetzt. Williams und Reiter (2005) stellen z. B. ein System vor, das Texte für Leser mit Leseschwäche generiert. Solche Variation könnte in Zukunft einen sinnvollen zusätzlichen Beitrag zu e-Inclusion leisten. Darüber hinaus versuchen jetzt einige Systeme, sich sogar dem *emotionalen* oder *affektiven* Zustand des Benutzers anzupassen: sogenanntes „affective user modelling“.

Empirische Grundlagen

Die Empirie spielt in der Benutzermodellierung in zweierlei Hinsicht eine Rolle: Erstens ist es empfehlungswert empirisch festzustellen, welche Arten von Anpassungen beim Systemdesign genau anzustreben sind. Zweitens ist es im voraus nicht einfach vorherzusagen, welche Anpassungen tatsächlich einem Benutzer helfen. Manche angeblichen „Vorteile“ können weniger günstige Konsequenzen haben. Untersuchungen zum menschlichen Verhalten sowie zur Interaktion zwischen Menschen und Systemen, die mit einer Benutzermodellierung versehen sind, sind daher unabdingbar.

Systemdesign: Beim Systemdesign muss die erwünschte Variation, die durch eine Benutzeranpassung zu steuern ist, identifiziert werden. Ein wesentliches Werkzeug für solche Untersuchungen ist die Korpusanalyse (s. Unterkapitel 4.1). Z. B. kann ein Korpus von Erklärungen für Nutzer mit verschiedenem Erfahrungshintergrund gesammelt und auf linguistische Eigenschaften hin untersucht werden, die von dem intendierten Benutzerkreis abzuhängen scheinen. Darüber hinaus muss für das System herausgearbeitet werden, welche vermuteten Eigenschaften der beteiligten Dialogpartner oder des Lesers dafür zuständig sind, dass eine bestimmte Menge von linguistischen Eigenschaften favorisiert wird und nicht eine andere. Ohne dies ist eine angemessene Kontrolle durch das Benutzermodell kaum möglich.

Um herauszufinden, welche Benutzereigenschaften das Verhalten eines Systems beeinflussen, können auch Experten befragt werden. Dies ist besonders angebracht bei der Entwicklung von Expertensystemen und Lernsystemen. Schäfer und Weyrath (1997) benutzten z. B. Interviews mit Angestellten der Feuerwehr, um die für das Einstufen der Dringlichkeit eines Notrufs heranzuziehenden Faktoren zu identifizieren. In einem Lehrkontext verwenden in ähnlicher Weise Carberry und Clarke (1997) die Einschätzungen chirurgischer Experten zur Herausarbeitung der Faktoren, die dazu beitragen, einen medizinischen Fall passenden Schwierigkeitsgrades für einen Studenten auszuwählen. In beiden Fällen sehen wir eine Beziehung zwischen **Wissensakquisition** und der Definition von Benutzermodellstruktur und -inhalt.

Ein relativ neuer Ansatz für die automatische Anpassung des Stils von generierten Texten an einem ausgewählten Korpus ist die pCRU-Architektur (Belz 2007). Diese Generierungsarchitektur betrachtet den gesamten Generierungsprozess als eine Menge von Entscheidungspunkten; jeder Entscheidungspunkt wird durch eine kontextfreie Phrasenstrukturregel dargestellt. Diverse maschi-

nelle Lernverfahren können dann angesetzt werden, um den Regeln Wahrscheinlichkeiten zuzuordnen. Das Ergebnis ist ein automatisch angepasstes System, dass in der Lage ist, Texte zu generieren, die die gleichen Verteilungen von linguistischen Merkmalen aufweisen wie die des Zielkorpus. Weitere Forschung wird hier benötigt, um die Relevanz und Möglichkeiten dieses Verfahrens für die Benutzermodellierung auszuarbeiten.

Evaluierung von Systemen: Dass Sprecher ihren Sprachgebrauch an ihren Zuhörer anpassen, ist eine Selbstverständlichkeit. Aber es ist nicht ratsam sofort daraus abzuleiten, dass Variation von Systemverhalten in Bezug auf den Benutzer automatisch von Vorteil sein muss. Obwohl vieles in der Benutzermodellierung auf der Beobachtung und Nachahmung menschlichen Verhaltens basiert, müssen noch die Auswirkungen solcher Variation bei der Mensch-Maschine-Interaktion sorgfältig evaluiert werden. Nachdem durch eine Korpusanalyse (s. Unterkapitel 4.1) oder durch andere aus natürlichen Interaktionskontexten gewonnene linguistische Daten gewisse Variationen in der Sprache zwischen Menschen belegt werden, muss immer noch die Umsetzung dieser Variation im Systemverhalten auf seine Zweckmässigkeit hin überprüft werden.

Dies kann durch Experimente mit dem System und Feedback von den Systembenutzern untersucht werden. Solche Evaluierungen können informell (durch z. B. Abfragen der Benutzer nach Benutzerzufriedenheit, Paris et al. 2003) oder formal und umfassend (durch Messung der Systemleistung in Zeit und Qualität über eine grosse Benutzeranzahl) sein. In letzterem Fall hat es sich bis jetzt als überraschend schwierig erwiesen, Vorteile bei der Benutzeranpassung statistisch zu belegen.

Beispielhaft für dieses Problem ist die von Reiter und Kollegen durchgeführte grosse empirische Studie, die die Effektivität der Benutzeranpassung bei sogenannten „smoking cessation letters“ untersucht hat. Solche Briefe werden in Großbritannien regelmäßig von Ärzten an Raucher geschrieben, um vorzuschlagen, dass sie mit dem Rauchen aufhören sollen. Untersucht wurde, ob eine höhere Erfolgsrate durch Texte bewirkt werden könnte, die mit einem automatisch generierten genauen Tailoring auf den Addresaten bzgl. der Häufigkeit des Rauchens, des Familienstands, der allgemeinen Gesundheit u.a. hergestellt wurden. Die erhofften Ergebnisse blieben aber aus: kein statistisch signifikanter Unterschied war feststellbar (Reiter et al. 2003).

Positive Tendenzen sind manchmal doch zu erkennen; aber weitere Studien sind dringend notwendig, um den nicht unerheblichen Aufwand für die Benutzermodellierung zu rechtfertigen. Die vielleicht ersten statistisch belegten positiven Ergebnisse sind die von Colineau und Paris (2007). Hier konnten für den Bereich der Informationsauskunft und Dokumentengenerierung signifikante Verbesserungen bei den Erledigungen von Aufgaben nach der Lieferung von angepassten Texten nachgewiesen werden. Weitere Information zu dem schwierigen Thema der Evaluierung in der Computerlinguistik im Allgemeinen ist in Kapitel 6 zu finden.

Inhalt des Benutzermodells

Aus der obigen Diskussion ist ersichtlich, dass ein Benutzermodell in verschiedener Weise und zu verschiedenen Zwecken eingesetzt werden kann. Daraus folgt, dass das Modell selbst eine Menge diverser Informationen über den Benutzer enthalten kann, abhängig von der jeweiligen Situation und von der Art und Weise, wie das Modell im System benutzt werden soll. Folgende Informationen sind häufig in einem Benutzermodell enthalten:

- Präferenzen, Interessen und Einstellungen: Dazu gehören Präferenzen des Benutzers in Bezug auf die Modalität der Informationspräsentation (gesprochen, graphisch, usw.), auf eine Lern- oder Problemlösungsstrategie, seine Einstellung zu einem Sachgebiet oder seine Interessen in Bezug auf bestimmte Themenbereiche.
- Benutzerwissen und -annahmen: Dazu gehören die Kenntnis von bestimmten Konzepten eines Sachgebietes, Annahmen des Benutzers oder der Grad seiner Vertrautheit mit einem Sachgebiet.
- Kognitive Fähigkeiten des Benutzers.
- Nichtkognitive Fähigkeiten des Benutzers: Dazu gehören perzeptuelle und motorische Fähigkeiten.
- Persönliche Charakteristika: Dazu gehören spezifische Eigenschaften des Benutzers wie Beruf, Persönlichkeitstyp, Ausbildungsabschlüsse, Wohnort, Geschlecht usw.
- Geschichte der Interaktion des Benutzers mit dem System: Dazu gehören bisherige Interaktionsmuster und alle Informationen, die das System dem Benutzer bereits präsentiert hat.

Viele dieser Dimensionen haben eine Entsprechung in der **Registertheorie** (siehe oben), wo ihre Korrelation mit bestimmten sprachlichen Mustern bereits eingehend untersucht worden ist. Bis jetzt sind diese linguistischen Kenntnisse jedoch in der Benutzermodellierung noch relativ wenig eingesetzt worden.

Aufbau und Verwaltung von Benutzermodellen

Für die Erstellung eines Benutzermodells werden zwei Methoden verwendet. Bei der ersten Methode fragt das System den Benutzer explizit nach den Informationen, die notwendig sind, um sein Verhalten an den Benutzer anpassen zu können. Dies findet meistens direkt statt, z. B. mit Hilfe von Selbsteinschätzungstechniken. Diese Methode hat zwei Vorteile: Sie ist einfach und offen, d.h. der Benutzer weiß, was das Modell über ihn enthält. Nachteilig ist, dass diese Methode von dem Benutzer leicht als aufdringlich und lästig beurteilt wird, da er eine Reihe von Fragen beantworten muss. Trotzdem wird diese Methode ihrer Einfachheit und Transparenz wegen oft verwendet, und in einigen Anwendungsbereichen (z. B. im Bereich der individuell angepassten Patienteninformation) sind die Benutzer schon mit dem notwendigen Fragebogen-Ansatz vertraut.

USER: *Can you tell me how to find out the inode of a file?*

TRACE: asserting *USER* knows INODE
 INODE has difficulty COMPLEX, so deducing:
 ruling out *USER* = NOVICE
 ruling out *USER* = BEGINNER
 USER is SOMEWHAT-UNLIKELY to be INTERMEDIATE
 USER is SOMEWHAT-LIKELY to be EXPERT
 asserting *USER* does not know UNIX-LS-I-COMMAND
 UNIX-LS-I-COMMAND has difficulty COMPLEX, so deducing:
 USER is SOMEWHAT-LIKELY to be INTERMEDIATE
 USER is SOMEWHAT-UNLIKELY to be EXPERT.

SYSTEM: *Type: ,ls -i‘*

USER: *What does rwho do?*

UNIX-RWHO-COMMAND has difficulty MUNDANE, so deducing:
 ruling out *USER* = EXPERT
 only one candidate left, so asserting:
 USER = INTERMEDIATE

SYSTEM: *rwho is like who, except rwho is for all users on the network*

Abbildung 3.52: Protokoll des Unix Consultant-Systems (Chin 1989, S.96).

In der zweiten Methode überwacht das System die Aktionen des Benutzers einschließlich seiner Antworten auf Fragen und zieht daraus Schlussfolgerungen über ihn. Größtenteils können die so gewonnenen Informationen als die in Benutzeraussagen enthaltenen Präsuppositionen (s. Abschnitt 3.7.3) aufgefasst werden. Ein sehr frühes Beispiel dieses Verfahrens ist das in Abbildung 3.52 angegebene Systemprotokoll (engl. *program trace*) aus dem Unix Consultant System (Chin 1989). Das Protokoll zeigt, wie das System auf Basis der von dem Benutzer gestellten Fragen ein für ihn zu erwartendes Wissensniveau ausarbeitet. Verschiedene Konzepte sind im System verschiedenen Niveaus zugeordnet als Ergebnis der Phase der Wissensakquisition. Daraus kann ein Wissensniveau des Benutzers auf der Basis der in den Fragen des Benutzers als bekannt gekennzeichneten vorkommenen Konzepte ermittelt werden.

In dem Protokoll wird ersichtlich, dass das System schließlich dem Nutzer den Status *intermediate* (mittlerer Wissenstand) zuordnet. Die generierte Antwort auf dieselbe Frage *What does rwho do?* ist ganz anders, falls das System vorher dem Benutzer den Status Neuling zuordnet – nämlich:

SYSTEM: *rwho is used to list all users on the network, list their tty, and list their login name.*

Wie bei der bereits diskutierten Generierung von Vergleichen, sehen wir hier, dass die für *intermediate*-Benutzer verwendete analoge Beschreibung für Anfänger nicht geeignet wäre.

Solche indirekten Methoden sind nicht aufdringlich und können daher für den Benutzer als vorteilhaft angesehen werden. Sie erfordern aber im Allgemeinen kompliziertere Algorithmen, um Inferenzen auf der Basis der vorhandenen Daten zu ziehen. Außerdem ist diese Methode für den Benutzer nicht transparent, d.h. der Benutzer weiß nicht, dass er beobachtet und modelliert wird. Ethische Fragen sind hier relevant, werden aber immer noch zu selten thematisiert. In Dialogsystemen ist dies möglicherweise weniger problematisch, weil es für den Benutzer aus den Dialogbeiträgen des Systems oft gut erkennbar ist, wie er eingestuft wird. Darüber hinaus können konkrete linguistische Eigenschaften der Äußerungen des Benutzers nicht nur Wissenszustände signalisieren, sondern auch präferierte Interaktionstile (Fischer und Bateman 2006).

Natürlich ist keine der beiden Methoden garantiert zuverlässig. Bei der Selbst-einschätzungs-methode kann es vorkommen, dass der Benutzer die Fragen nicht ehrlich beantwortet (insbesondere bei der Beurteilung von Fähigkeiten und Wissen) oder gar nicht in der Lage ist, sich selbst genau einzuschätzen. Bei der zweiten Methode können die aus dem gezeigten Benutzer-verhalten gezogenen Schlussfolgerungen u.U. inkorrekt sein. Um das Gewinnen von Information über den Benutzer zu beschleunigen und ihre Konsistenz leichter zu überprüfen, werden Fakten nicht vereinzelt behandelt, sondern als strukturierte Mengen von zusammengehörenden Eigenschaften. Daraus ergibt sich der Begriff von **Benutzerstereotypen**. Wenn ein Benutzer einem bestimmten Stereotyp zugeordnet wird, dann werden alle Eigenschaften dieses Stereotyps als für ihn zutreffend angenommen. Diese Methode wurde zum ersten mal in Rich (1979) angewendet und ist seitdem zu komplexeren Methoden weiterentwickelt worden. Ein Beispiel für einen Stereotyp eines Geschäftsmannes bzw. einer Geschäftsfrau aus einem späterem System von Rich ist in Abbildung 3.53 zu sehen. Dieser Stereotyp beinhaltet Anforderungen an Hotelreservierungen für einen Geschäftsmann bzw. eine Geschäftsfrau, wenn er oder sie in zwei verschiedenen Angelegenheiten unterwegs ist. Eine Sammlung von solchen *if-then*-Regeln bildet zusammen einen Stereotyp.

```

if      goal = plan(business-trip)
then    subgoal = get-flight(first-class)
        subgoal = get-hotel(first-class,quiet,close-to(meeting-place))
        subgoal = get-car(high-quality,small)
        subgoal = get-restaurant-list(expensive)

if      goal = plan(pleasure-trip)
then    subgoal = get-flight(coach)
        subgoal = get-hotel(moderate,adjoining-rooms,pool)
        subgoal = get-car(intermediate,family-size)
        subgoal = get-restaurant-list(moderate,take-children)

```

Abbildung 3.53: Stereotyp für Geschäftsmann/-frau nach Rich (1979, S. 38)

Die Annahme, dass ein gewisser Stereotyp zutrifft, oder dass gewisse Teile eines Stereotyps zutreffen, ist im Regelfall keine einfache ja/nein-Entscheidung. Die Zuweisung ist immer punktuell und änderbar, wenn neue Hinweise über den Benutzer vom System erkannt werden. Zu jedem Zeitpunkt der Interaktion kann das System herausfinden, dass einige oder alle stereotypen Annahmen für einen bestimmten Benutzer falsch waren und ersetzt werden müssen. Z. B. kann das System erfahren, dass eine bestimmte Person nicht gerne fliegt. Das konkrete Benutzermodell für diese Person erbt dann die allgemeineren Eigenschaften des Stereotyps nur entsprechend eingeschränkt. Deshalb werden für die Verwaltung von solchen Benutzermodellen häufig nicht-monotonen Logiken sowie Fuzzy-Logik (s. Unterkapitel 2.1), maschinelle Lernalgorithmen, neuronale Netzwerke und probabilistische oder statistische Verfahren einschließlich Bayes'scher Methoden (s. Unterkapitel 2.4) eingesetzt. Querverbindungen zur komplexen formalen Modellierung von Wissen und Annahmen (engl. *Belief Management Systems*) lassen sich auch leicht ziehen (s. Abschnitt 3.7.1).

Ein konkretes Beispiel: das System TAILOR

Abschließend illustrieren wir einige der oben genannten Punkte und Methoden anhand eines konkreten Systems: das klassische von Paris (1993) entwickelte System TAILOR. In TAILOR hat Paris gezeigt, wie ein Modell des Fachwissens des Benutzers von einem natürlichsprachlichen Generierungssystem (s. Unterkapitel 3.8) verwendet werden kann, um die generierten Texte angemessen zu beeinflussen. TAILOR erzeugt benutzergerechte Beschreibungen von Haushaltsgegenständen für jede Art von Benutzer; vom Anfänger bis zum Experten.

Für das Design von TAILOR hat Paris den oben beschriebenen korpusbasier-ten Ansatz gewählt. Sie verglich die Einträge zur Definition von Haushaltsgegenständen in Enzyklopädien für Erwachsene und für Kinder. Hierbei zeigten sich zwei Hauptstrategien zur Gegenstandsdefinition:

1. eine Aufzählung der Teile und Eigenschaften (Farbe, Größe usf.) des Ge- genstands – die strukturelle Strategie, und
2. eine Beschreibung, wie der Gegenstand funktioniert – die funktionale Stra- tegie.

Die Korpus-Studie zeigt außerdem, dass eine enge Korrelation zwischen der gewählten Strategie und dem angenommenen Wissensstand des Lesers besteht. Unter der Annahme, dass der Leser den Gegenstand kennt, war offensichtlich die strukturelle Beschreibung ausreichend, während für einen unbedarften Leser immer die funktionale Beschreibung gegeben wurde.

Die beiden gefundenen Strategien wurden im TAILOR-System als zwei ver- schiedene Textorganisationsschemata implementiert (s. a. Unterkapitel 3.8):

- Das **Konstituenten-Schema**, das der strukturellen Strategie entspricht, beschreibt einen Gegenstand mit Hilfe seiner Teile und Eigenschaften.
- Das **prozedurale Schema**, das der funktionalen Strategie entspricht, be- schreibt einen Gegenstand durch Erläuterung der Mechanismen, die es dem

Gegenstand ermöglichen, seine Funktion auszuüben. Statt Einzelteile und Eigenschaften zu beschreiben konzentriert sich diese Strategie auf kausale Beziehungen.

Die Konzepte in TAILORs Wissenbasis teilen sich in zwei Klassen auf, spezifische Konzepte (lokales Sachwissen) und Basiskonzepte. Beispiele für spezifische Konzepte sind *Mikrofon*, *Sender* oder *Telefon*. Zu den Basiskonzepten gehören z. B. *Magnetismus* und *Spannung*. Das Benutzermodell wurde beim Aufruf des Systems durch eine Abfrage des Benutzers, welche Konzepte er kennt, aufgebaut. Ein naiver Benutzer wurde modelliert als einer, der kein lokales Sachwissen hat und auch keine Basiskonzepte kennt. Ein Experte wurde modelliert als jemand, der alle Basiskonzepte sowie die meisten spezifischen Konzepte aus der Wissensbasis kennt. Mit Hilfe der Menge der bekannten Konzepte konnte dann jede Art von Benutzer auf dem Spektrum von *naiv* bis *Experte* repräsentiert werden. Zur Aktualisierung des Benutzermodells wurde jedesmal, nachdem eine Beschreibung generiert worden war, das beschriebene Konzept mit in die Menge der bekannten Konzepte aufgenommen.

Die Wahl der passenden Diskursstrategie erfolgte mit Hilfe von TAILORs Benutzermodell. Wenn der Gegenstand dem Benutzer bereits bekannt war, wurde das Komponenten-Schema gewählt. Ansonsten wurde das prozedurale Schema instantiiert. Für die in dem zu erzeugenden Text vorkommenden Teile wurde seinerseits eine Beschreibung erstellt. TAILOR konnte dabei die beiden Strategien auch kombinieren. Wenn z. B. der Benutzer das Objekt *Telefon* nicht kannte, wurde die prozedurale Strategie gewählt. Wenn dabei die Teile *Mikrofon* und *Lautsprecher* vorkamen, und der Benutzer mit diesen Konzepten vertraut war, dann wurden sie strukturell erklärt.

Der folgende von TAILOR generierte Text zur Beschreibung eines Telefons zeigt diese Kombination. Der Benutzer ist im Allgemeinen mit einem Telefon vertraut, kennt aber die Mikrofon-Transmitter-Komponente nicht. Der Text wird deshalb primär mit dem Komponentenschema erzeugt, abgesehen von dem Abschnitt über den Transmitter (im Text kursiv gezeigt).

The telephone is a device that transmits soundwaves. The telephone has a housing that has various shapes and various colors, a transmitter that changes soundwaves into current, a curly-shaped cord, a line, a receiver to change current into soundwaves and a dialing-mechanism. *The transmitter is a microphone with a small diaphragm. A person speaking into the microphone causes the soundwaves to hit the diaphragm of the microphone. The soundwaves hitting the diaphragm causes the diaphragm to vibrate. The vibration of the diaphragm causes the current to vary.* The receiver is a loudspeaker with a small aluminum diaphragm. The housing contains the transmitter and it contains the receiver. The housing is connected to the dialing-mechanism by the cord. The line connects the dialing-mechanism to the wall. (Paris 1993, S. 97)

Im Detail läuft der Entscheidungsprozess bei TAILOR folgendermaßen ab: Zuerst gibt TAILOR den Oberbegriff von dem zu beschreibenden Konzept als Text

(*a device that transmits soundwaves*) aus. Dann überprüft das System für jede Komponente des Konzepts (*housing, transmitter, usw.*), ob und wie das entsprechende Konzept schon im Benutzermodell bekannt ist. Ist das Konzept bekannt, verwendet TAILOR das Komponenten-Schema rekursiv weiter, um die Komponente zu beschreiben. Falls das Konzept dem Benutzer nicht bekannt ist, generiert das System mit dem prozeduralen Schema weiter. Im obigen Beispiel geschieht dies bei dem Konzept *Sender* (*transmitter*), und es wird ein Textfragment generiert, das die kausalen Beziehungen, die die Funktion des Teils beschreiben, hervorhebt.

Entscheidend ist hierbei, dass eine Vielfalt von solchen dem Benutzerniveau entsprechenden Texten aus ein und derselben Wissensbasis erzeugt werden kann. Die Fähigkeit, Informationen so auf einen bestimmten Benutzerkreis zuzuschneiden, hat einen enorm wertsteigernden Effekt für eine Wissensbasis. Einmal aufgebaut, kann eine solche Wissensbasis von einem wesentlich breiteren Publikum benutzt werden als ohne Benutzermodell. Diese Wertsteigerung ist eine der wichtigsten Motivationen für den Einbau von Benutzermodellen in praktische Anwendungen.

Zukunftsansichten der Benutzermodellierung

Forschung zur Benutzermodellierung wird zur Zeit in den folgenden Gebieten durchgeführt:

- Präsentationsanpassung beim Information Retrieval,
- Präsentationsanpassung für Hypermedianavigation und Mediumauswahl,
- Computerlinguistik: insbesondere für Informationspräsentation (inklusive Hypertext und Hypermedia) und Dialogmanagement,
- Mensch-Maschine-Kommunikation: Bereitstellung adaptiver Interfaces,
- QA-Systeme,
- Unterricht und Lehre: insbesondere für computergestützte Lehr- und Lernsysteme und für eine dem Benutzerniveau angepasste Auswahl von Übungen und Erklärungen.

Mit dem Aufkommen von Dialogsystemen auf der Basis von gesprochener Sprache und der wachsenden Individualisierung, die nun auch von persönlich angepassten (engl. *customised*) Webseiten und dem massiven Wachstum der computergestützten Informationsbeschaffung angetrieben wird, besteht ein anhaltendes Interesse an weiterer Forschung auf dem Gebiet der Benutzermodellierung, auch in der Computerlinguistik. Insbesondere müssen Systeme in der Lage sein, sehr schnell in einem Dialog ein Benutzermodell zu erstellen und Antworten auf der Basis dieses Modells zu generieren. Die meisten Customization- und Individualisierungsstrategien in heutigen praktischen Anwendungen sind allerdings noch recht einfach. Komplexeres Verhalten muss sich an vergangenen und aktuellen Forschungsergebnissen zur Benutzermodellierung orientieren. Die dort

vorgeschlagenen Verfahren müssen jedoch noch zuverlässiger und effektiver werden und sich empirisch verifizieren lassen.

Literaturhinweise

Der Sammelband von Kobsa und Wahlster (1989) stellt immer noch eine gute Einführung in die Benutzermodellierung dar. Für weitere Beschreibungen von Benutzermodellierungsverfahren in der Computerlinguistik sei der interessierte Leser auf die Zeitschrift *User Modeling and User Adapted Interaction* verwiesen, insbesondere die Artikel in der 2001 erschienenen „10 Year Anniversary“ Ausgabe wie z. B. Zukerman und Litman (2001) und Kobsa (2001) sind lesenswert. Darüber hinaus bieten Brusilovsky (2003) und Jameson (2008) Einblicke in aktuelle Forschungsthemen. Die Tagungsbände der *User Modelling International Conference* erscheinen jedes zweites Jahr.

3.8 Textgenerierung

Helmut Horacek

Menschen produzieren Texte in verschiedener Form, und diese unterscheiden sich je nach Gegebenheit der vorliegenden Situation oft erheblich hinsichtlich Inhalt, Detail und Ausdrucksweise. Um diese Fähigkeit zumindest für aufgabenorientierte Texte auf einem Rechner nachzubilden, werden relevante Aspekte einer Situation modelliert, in der die Produktion einer Äußerung angebracht ist. Es wird versucht, durch schrittweises Anreichern und Konkretisieren einer ursprünglich abstrakten und oft vagen Spezifikation einer kommunikativen Absicht, zu einer vollständigen Beschreibung eines Diskursbeitrags zu kommen und im Weiteren zu einer adäquaten sprachlichen Umsetzung dieser Beschreibung. Die Rolle der **Diskursplanung**, die im Wesentlichen den gestalterischen Anteil eines solchen Verfahrens ausmacht, besteht in diesem Kontext im Ausdrücken dieser **kommunikativen Absicht** in Form eines **Textplans**. Die Umsetzung dieses Plans erfordert dann eine Überführung der darin befindlichen Spezifikationen in Text, und zwar mit den vorhandenen linguistischen Mitteln. Betrachten wir als Beispiel dazu eine Wegauskunft, ein beliebter Anwendungsbereich für die Sprachgenerierung (siehe Carstensen 1991). Die kommunikative Absicht, einem Auskunftsuchenden Kenntnis über den Weg von einem Ort A zu einem Ort B zu verschaffen, bedarf bei nicht-trivialer Wegstrecke zu ihrer Umsetzung einer Folge sorgfältig geplanter und realisierter Äußerungen. Darin werden die Abschnitte, aus denen sich ein solcher Weg zusammensetzt, in geordneter Weise und geeigneter Ausführlichkeit angeführt, je nach situativen Gegebenheiten und Informationsbedürfnis des Adressaten, wie etwa in folgendem Beispiel:

- (3.195)
- (a) *Das ist ein kurzer Weg.*
 - (b) *Sie gehen von hier weiter bis zur Kirche und biegen dort links ab.*
 - (c) *Dann kommen Sie an einem großen Haus, dem Rathaus, vorbei und erreichen schließlich B.*

Um einen solchen Text zu erzeugen, muss der gesamte Weg mental in genügend kleine, leicht fassbare Abschnitte zerlegt und es müssen dazu Beschreibungen gebildet werden. Diese Beschreibungen sollten dann in kohärenter Weise zusammengesetzt und zur besseren Orientierung an einigen Stellen mit Referenzen auf auffällige Objekte, sogenannte Landmarken, versehen werden. Schließlich ist eine angemessene sprachliche Umsetzung des Plans in Form von Text erforderlich, von der man plausibel annehmen kann, dass dem Auskunftsuchenden mit der erzeugten Beschreibung gedient ist.

Im weiteren Verlauf dieses Kapitels wollen wir Verfahren zur automatischen Planung solcher Diskurse und deren Umsetzung in Form von Text näher untersuchen. Dazu beschreiben wir zunächst die Aufgaben der Diskursplanung und der Planumsetzung. Danach betrachten wir die Funktionalität dieser Prozesse auf einer etwas formaleren Basis. Im Hauptteil werden die grundlegenden Methoden

der Diskursplanung und Planumsetzung vorgestellt, wesentliche Annahmen dieser Methoden und daraus resultierende Probleme diskutiert sowie ihre Eignung charakterisiert. Abschließend wird die Rolle der wichtigsten mit Generierung befassten linguistischen Theorien erläutert.

3.8.1 Aufgaben der Planung und Umsetzung

Generell besteht die Aufgabe der Diskursplanung und der Umsetzung dieses Plans darin, eine bestimmte kommunikative Absicht bei gegebenem Kontext und Annahmen über den Adressaten zu interpretieren, daraus die Spezifikation eines Textplans zu bilden und schließlich daraus mittels zur Verfügung stehender linguistischer Ressourcen einen Text zu erzeugen, von dem plausibel angenommen werden kann, dass sein Äußern die kommunikative Absicht erfüllt. Um die dabei anfallenden Planungsaufgaben in einem formalen System zu erfüllen, wird eine der Domäne und Anwendung angemessene Wissensbasis herangezogen und in zweierlei Hinsicht verarbeitet:

- Es werden solche Elemente der Wissensbasis gewählt, die für die Erfüllung der kommunikativen Absicht als ausreichend relevant erachtet werden. Dieser Aspekt wird als **Inhaltsfestlegung** bezeichnet.
- Die gewählten Informationen, die in der Wissensbasis in kontextneutraler Weise repräsentiert sind, müssen in eine zusammenhängende, kohärente, den Bedürfnissen des Adressaten angemessene Struktur gebracht werden. Dieser Aspekt wird als **Inhaltsorganisation** bezeichnet.

Wegen der großen Zahl und unterschiedlichen Art von Maßnahmen, die bei der Planung und Planumsetzung berücksichtigt werden müssen, wird der gesamte Verarbeitungsprozess meist in drei Abschnitte gegliedert:

- Die Festlegung, welche Informationen kommuniziert werden, wie sie angeordnet und wie die einzelnen Teile rhetorisch zueinander in Beziehung gesetzt werden, macht in gewissem Sinne die Planung im Großen aus. Dieser Abschnitt wird als Diskursplanung oder **Textplanung** bezeichnet.
- Basierend auf der durch die Textplanung aufgebauten Grobstruktur wird eine feinere Strukturierung vorgenommen, die Satzebene betreffend. Dazu gehören hauptsächlich die Zusammenfassung von Aussagen mit gemeinsamen Teilen (**Aggregation**), Bestimmung der Satzgrenzen und der Topik/Fokus-Struktur, Erzeugung von **Referenzausdrücken** und Wahl der linguistischen Ressourcen zum Ausdrücken konzeptueller Inhalte (**Lexikalisation**). Dieser Abschnitt wird als **Satzplanung** bezeichnet.
- Schließlich werden die abstrakten Satzrepräsentationen in Text umgesetzt. Dieser Abschnitt wird als linguistische **Oberflächenrealisierung** bezeichnet.

Dabei besteht das Zwischenergebnis nach der Diskursplanung in einer teilgeordneten Struktur von Spezifikationen satzartiger Aussagen, mit zusätzlichen Angaben für die rhetorischen Bezüge zwischen diesen Elementen. In der Satzplanung werden diese Strukturen um Spezifikationen für Referenzausdrücke, Konnektive, Topik/Fokus-Struktur und Rolle in der Diskursstruktur (Haupt- oder Nebensatz sowie deren Reihenfolge) angereichert, sowie durch Maßnahmen wie Aggregation und Lexikalisierung teilweise umstrukturiert.

Die Art des Zusammenspiels zwischen Inhaltsfestlegung und Inhaltsorganisation und auch die Einbindung und Abfolge einzelner Maßnahmen der Satzplanung sind jedoch nicht weiter festgelegt. Da sie sich in einer Reihe von Fällen gegenseitig bedingen, ist eine opportunistische Verzahnung dieser beiden Aspekte bzw. der unterschiedlichen Maßnahmen in einem Planungsprozess aus theoretischer Sicht motiviert. Dem tragen auch verschiedene aufwändige, nicht-sequentielle **Architekturtypen** Rechnung wie integrierte, interaktive und revisionsbasierte. In implementierten Systemen und auch in methodischen Verfahren wird hingegen aus praktischen Erwägungen meist eine strikte Abfolge – zunächst Inhaltsfestlegung, dann -organisation, dann Maßnahmen zur Satzplanung – eingehalten, gemäß der mittlerweile als Standardarchitektur geltenden Aufteilung von Reiter (1994, siehe Abbildung 3.54).

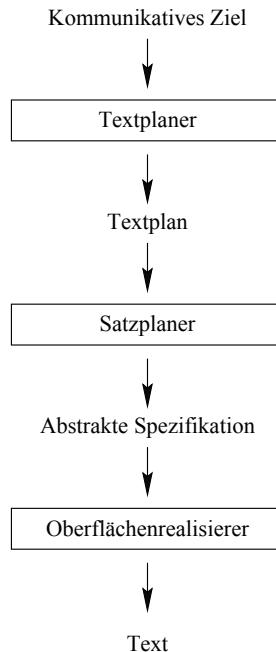


Abbildung 3.54: Die Standard-Generierungsarchitektur

Wie auch immer die Organisation des gesamten Planungsprozesses (Diskurs- und Satzplanung) aufgebaut ist, es muss die Gewährleistung der Umsetzbar-

keit des erzeugten Textplans mit den vorliegenden linguistischen Mitteln sichergestellt sein. Dies ist das Problem der **Generierungslücke** (engl. *generation gap*). Dieses Problem entsteht grundlegend dadurch, dass die Ausgangsspezifikationen für die Textgenerierung im Allgemeinen nicht-linguistischer Natur sind. Deshalb muss im Rahmen von Text- und Satzplanung beim Übergang von nicht-linguistischen zu linguistischen Elementen die Zusammensetzbartigkeit der ausgewählten linguistischen Elemente gesichert sein.

Beispiel 3.8.1

Ein *Beschluss*, der zu einem *wichtigen* Ergebnis geführt hat, kann nur in nominalisierter Form in einer Phrase ausgedrückt werden (*Wir haben einen wichtigen Beschluss gefasst.*) und nicht als Vorgang (**Wir haben wichtig beschlossen.*). □

An vielen Stellen der Diskurs- und Satzplanung bestehen Wahlmöglichkeiten in Bezug auf Ausführlichkeit, Ausdrucksweise oder Anordnung von Textteilen. Dabei sind jedoch die Gründe für die Auswahl zwischen möglichen Varianten im Allgemeinen oft noch wenig verstanden. Meist werden Präferenzen eher auf pragmatisch motivierte Entscheidungen und spezifische Domänenkonventionen zurückgeführt und nicht auf rhetorisch bedingte Faktoren. Im Allgemeinen erweisen sich die linguistischen Kriterien eher als notwendig, um offensichtlich ungeeignete Varianten von Textspezifikationen zu vermeiden, aber nicht als hinreichend, um Präferenzen in einem bestimmten situativen Kontext ableiten zu können.

Für die Wegplanung bedeutet die Inhaltsbestimmung die Festlegung eines Pfades vom Ausgangsort zum Zielort, inklusive der an den entsprechenden Stellen einbezogenen Landmarken und sonstiger ergänzender Information, wie etwa Übersichtshinweise. Die Inhaltsorganisation ergibt sich in Bezug auf die Reihenfolge von Aussagen bereits aus der Folge von Wegsegmenten. In der Satzplanung können allerdings Beschreibungen von mehreren Segmenten fallweise koordiniert und ergänzende Informationen, etwa zu Landmarken, geeignet eingefügt werden. Wahlmöglichkeiten bestehen unter anderem darin, welche Teilstrecken explizit genannt und welche koordiniert werden, welche Landmarken zur Referenz herangezogen werden und wie sie beschrieben werden und welche begleitenden Hinweise in die eigentliche Wegbeschreibung eingefügt werden. Dabei kann durch eine Verzahnung mit der Inhaltsorganisation eine motiviertere Inhaltauswahl, etwa bei Entscheidungen zu Übersichtshinweisen, unterstützt werden, wenn Informationen über Segmentkoordinationen bereits bei der Inhaltauswahl zur Verfügung stehen.

3.8.2 Funktionalität des Planungsprozesses

Die Beschreibung einer Situation, in der die Erzeugung einer Äußerung oder eines Textes angebracht ist, kann im Sinne von Reiter und Dale (2000) als Quadrupel

$$\langle k, c, u, d \rangle$$

aufgefasst werden, wobei

- k die Wissensbasis,
- c die kommunikative Absicht,
- u das Adressatenmodell und
- d die Diskurshistorie

darstellen. Im Rahmen dieses Buchkapitels beschränken wir uns im Wesentlichen auf die Rolle der ersten beiden Komponenten, da das Adressatenmodell im Abschnitt über Benutzermodellierung des Unterkapitels 3.7 beschrieben wird. Wir verweisen nur gelegentlich bei der Auswahl zwischen alternativen Möglichkeiten auf Annahmen aus einem Adressatenmodell oder auf Elemente einer Diskurshistorie, die die damit verbundenen Entscheidungen motivieren.

Bei der Spezifikation einer kommunikativen Absicht und der damit verbundenen Inhalte der Wissensbasis können Diskrepanzen zwischen den Annahmen eines Diskursplanungsverfahrens und den vorliegenden Gegebenheiten einer Anwendungseinbettung auftreten. Während Diskursplanungsverfahren meist vereinfachend annehmen, dass die Propositionen, auf denen sie arbeiten, linguistisch motiviert aufgebaut sind und in Haupt- oder Nebensätzen ausgedrückt werden können, kann in Hintergrundsystemen eine solche Absicht und speziell die damit verbundenen Inhalte in sehr unterschiedlicher, oft nicht-linguistischer Form vorliegen. Dabei kann es sich unter anderem um Tabellen mit numerischen Daten, Relationen einer Datenbank, oder Propositionen aus einem wissensbasierten System handeln. Je nach Inhalt der Wissensbasis muss also unter Umständen eine Interpretation von nicht-linguistischen Daten vorgenommen werden, durch die ihnen Entsprechungen in Form von Propositionen mit sprachlich ausdrückbaren Prädikaten zugeordnet werden. Die Definition solcher Zuordnungen kann durch eine Kombination von Auswahl, Zusammenfassung und Inferenzen auf der Basis der vorliegenden Daten erfolgen und ist in hohem Maße eine anwendungs- und domänenabhängige Aufgabe, deren Systematik noch wenig entwickelt ist. Diese Umstrukturierungen, die die Bildung von durch lexikalisches Material ausdrückbaren Propositionen aus den Elementen der Wissensbasis zum Ziel haben, werden auch als **Lexikalisierung im eigentlichen Sinn** bezeichnet, während die Auswahl zwischen alternativen Lexemen (d. h. die Wortwahl) als **Lexikalisierung im engeren Sinn** angesehen wird. Diese Kategorisierung geht auf Analysen im RAGS-Projekt (Reference Architecture for Generation Systems) zurück (Cahill 1998). Einige Beispiele bei der Konstruktion von linguistisch motivierten Strukturen aus Repräsentationen von Leistungssystemen, deren Ergebnisse oder Verhalten mit einem Generierungssystem sprachlich beschrieben werden soll, findet man in Reiter und Dale (2000). Darüber hinaus sind einige Arten von Umformungen linguistisch und taxonomisch motiviert und führen zur Verbesserung der Inhaltsstrukturierung – die **Meaning Text Theorie**, die weiter unten ausführlicher behandelt wird, stellt hierzu ein variantenreiches Instrumentarium zur Verfügung, wobei allerdings der Aufwand zur Erstellung des lexikalischen Materials und der Berechnungsaufwand zur Umstrukturierung der Spezifikationen extrem hoch sind.

Um der Annahme von Diskursplanungsverfahren in Bezug auf Eingabespezifikationen gerecht zu werden, erfolgt oft vor der Diskursplanung die Lexikalisierung im eigentlichen Sinn, während nach der Diskursplanung nur noch die Auswahl von Lexemen vorgenommen wird. Diese umfasst die Wahl zwischen Synonymen, Alternativen von Ausdrücken mit unterschiedlichen Perspektiven (wie etwa *kau-
fen* vs. *verkaufen*) und eventuellen Nominalisierungen (wie etwa *der Kauf*) oder ähnliche Variationen, die den gewünschten Informationsgehalt in vergleichbaren Strukturen ausdrücken. Diese Aufteilung der Lexikalisierung ist systematisch, und sie ermöglicht eine einfachere Textplanung. Sie legt jedoch viel Verantwortung auf die Lexikalisierung im eigentlichen Sinne, so dass die erzeugten Propositionen im Zusammenhang oft sprachlich verbessерungsbedürftig sind, was sich dann in Texten äußert, die als schematisch und hölzern empfunden werden. Wir werden anhand von Beispielen von Wegbeschreibungen sehen, welche Verbesserungen durch Umstrukturierung der Information in Varianten linguistisch motivierter Repräsentationen und sprachliche Umformulierung erzielt werden können, wobei auch klar werden sollte, dass das Bilden der entsprechenden Propositionen ohne den Kontext der Diskursplanung nicht motiviert erfolgen kann.

Die Ergebnisse der Diskursplanung, die sich in einer strukturellen Zusammensetzung von satzartigen Propositionen, meist einer rhetorischen Baumstruktur, manifestieren, sind in mehrerer Hinsicht zu unspezifisch, um eine typische Oberflächenrealisierungskomponente anzusteuern. Zu den dazu erforderlichen Maßnahmen gehört in jedem Fall die Wahl von Lexemen für die in den Propositionen vorkommenden Prädikate.

Beispiel 3.8.2

In der Wegauskunft (3.195) etwa wird die (hypothetische) Fortbewegung des Auskunftsuchenden einmal durch *gehen* in (b) und einmal durch *vorbeikommen* in (c) ausgedrückt, in diesem Fall in Abhängigkeit von der relativen Position der Landmarken. □

Eine weitere Aufgabe besteht in der Erzeugung von Referenzausdrücken für die in den Propositionen durch Variablen repräsentierten Objekte. Die erzeugten Phrasen können Pronominalisierungen sein, einfache Referenzen mittels Objekt-kategorien oder umfangreichere Beschreibungen. Schließlich erfolgt die Bildung von expliziten Satzstrukturen aus den diesbezüglich neutralen Propositionen, mit Festlegungen für Haupt- und Nebensatz, sowie Koordination von Strukturen mit partiell übereinstimmenden Teilen.

Beispiel 3.8.3

In (3.195) wird in (c) *einem großen Haus, dem Rathaus*, also eine umfangreichere Beschreibung als Referenzausdruck erzeugt. Die Beschreibung von (b) wird aus ursprünglich zwei Propositionen gebildet, wobei wegen Subjektsgleichheit das Vorkommen von *sie* im zweiten Hauptsatz ausgelassen wird. □

Das Ergebnis der Satzplanung kann von unterschiedlicher Art sein, je nach Anforderungen der verwendeten Oberflächenrealisierungskomponente. Mögliche Varianten sind abstrakte syntaktische Beschreibungen, lexikalierte Kasusrahmen

sowie Mischformen, auch mit festgelegten Textteilen an einzelnen Stellen dieser Strukturen.

Betrachten wir nun die Aufgabe einer Wegbeschreibung aus einer etwas formaleren Sicht. Dafür könnten die kommunikative Absicht c und die Wissensbasis k (hier ist nur jener Teil angegeben, der für den vorliegenden Fall relevant ist) in einer nicht-linguistischen Form wie folgt aussehen:

$$\begin{aligned} c &= \text{Describe}(\text{Route}, A, B) \\ k &= \{ \text{at}(\text{User}, A), \text{Path}(A, C), \text{Path}(C, D), \text{Path}(D, B), \text{Left}(\text{Path}(A, C), \\ &\quad \text{Path}(C, D)), \text{Straight}(\text{Path}(C, D), \text{Path}(D, B)), \text{Type}(C, \text{church}), \\ &\quad \text{Type}(D, \text{town-hall}), \text{Property}(D, \text{big}), \dots \} \end{aligned}$$

Die Spezifikation der kommunikativen Absicht besagt in diesem Beispiel, dass eine Beschreibung des Wegs von A nach B zu produzieren ist. Die Wissensbasis enthält unter anderem den momentanen Standort des Adressaten der Beschreibung ($\text{at}(\text{User}, A)$), Teilstrecken ($\text{Path}(A, C)$), Richtungsbezüge zwischen angrenzenden Teilstrecken ($\text{Left}(\text{Path}(A, C), \text{Path}(C, D))$) und Eigenschaften von Landmarken ($\text{Type}(D, \text{town-hall})$, $\text{Property}(D, \text{big})$).

Für Zwecke der Diskursgenerierung müssen die Spezifikationen der Wissensbasis zunächst in linguistisch motivierte Prädikate überführt werden, die an Stelle der abstrakten Pfadeigenschaften für den Kommunikationszweck geeignete Prädikationen ausdrücken, wie (hypothetische) Fortbewegungen einer Person (hier: des Auskunfts suchenden). Weitergehende Umstrukturierungen der Prädikationen der Wissensbasis sind in diesem Fall zum Unterschied von anderen Domänen und Anwendungssystemen nicht nötig. Dieser Schritt könnte als Lexikalisierung im eigentlichen Sinne aufgefasst werden, und das Resultat würde etwa wie folgt aussehen:

$$\{ \text{be}(\text{User}, \text{at}(A)), \text{go}(\text{User}, \text{from}(A), \text{to}(C)), \text{go}(\text{User}, \text{from}(C), \text{to}(D)), \\ \text{go}(\text{User}, \text{from}(D), \text{to}(B)), \text{turn}(\text{User}, \text{at}(C), \text{direction(left)}), \\ \text{turn}(\text{User}, \text{at}(D), \text{direction(none)}), \text{church}(C), \text{town-hall}(D), \text{big}(D) \dots \}$$

In der weiteren Verarbeitung wird sich allerdings zeigen, dass Umstrukturierungen innerhalb dieser linguistisch motivierten Prädikationen (etwa Koordination mehrerer Prädikationen, Anreichern mit Konnektiven, diversifizierte Wortwahl) unerlässlich sind, um die Spezifikationen des Textes kommunikativ adäquater zu gestalten, etwa zur Vermeidung stereotyper Wiederholungen.

Wir wollen davon ausgehen, dass diese Interpretation der Elemente der Wissensbasis im Zusammenhang mit der Diskursplanung vorgenommen wird, inklusive der Umstrukturierung zur Verbesserung der Informationsaufteilung. Während die Eingangsspezifikationen typischerweise ungeordnet sind oder zu anderen Zwecken strukturierte Informationen beinhalten (etwa zur Pfadsuche für die Wegauskunft), manifestiert sich das Ergebnis der Diskursplanung in einer baumartigen Textstruktur, in der die Blattknoten die einzelnen Aussagen des zu erzeugenden Textes enthalten und die Zwischenknoten Diskursrelationen (siehe Abschnitt 3.7.1) darstellen, die die Bezüge dieser Aussagen beschreiben. Eine Textstruktur für den in der Einführung angegebenen Text könnte dann wie in Abbildung 3.55 angegeben aussehen.

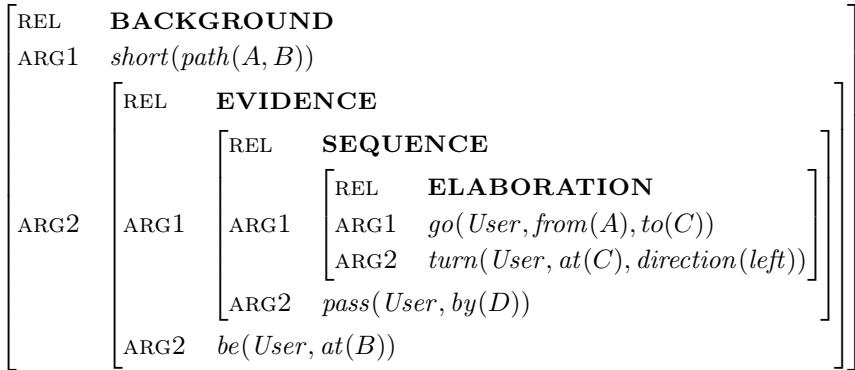


Abbildung 3.55: Textstruktur für die Wegbeschreibung (3.195)

Um diese Struktur zu erzeugen, werden zunächst die Elemente des Pfades von Ausgangsort *A* nach Zielort *B* als Sequenz von Fortbewegungen des Auskunfts suchenden ausgedrückt – darin manifestiert sich der Aspekt der Textorganisation. Zusätzlich werden kommunikativ motivierte Operationen (opportunistisch) angewandt, die diese Struktur umformen. Um dem Hörer das Verständnis der Wegbeschreibung zu erleichtern, wird eine kurze Charakterisierung des Gesamt wegs als **BACKGROUND** der gesamten Wegbeschreibung vorangestellt, und das Erreichen des Zielorts wird explizit angegeben (*be(User, at(B))*) wird mit einer **EVIDENCE** Relation versehen und eingefügt). Diese Informationsanreicherung kann als ein Teil der Inhaltsauswahl angesehen werden. Die am Ende der Strecke von *A* nach *C* nötige Richtungsänderung ergänzt mittels einer **ELABORATION** Relation die Beschreibung dieser Strecke. Die Beschreibung der folgenden zwei Teilstrecken, zwischen denen keine Richtungsänderung stattfindet, (*go(User, from(C), to(D))* und *go(User, from(D), to(B))*) wird zu einer Strecke zusammengefasst und durch die Beschreibung der dabei passierten Landmarke ausgedrückt (*pass(User, by(D))*). Die Rechtfertigung dieser Ausdruckswahl liegt darin, dass aus dem Vorbeigehen an einem Ort (hier: *D*) auch geschlossen werden kann, dass dazu zunächst der Weg von dem vorher genannten Ort zurückgelegt werden muss. Dies ist eine kontextuell motivierte Inferenz, die dem Hörer zugetraut wird, und solche Situationen kommen in natürlichen Dialogen häufig vor. Ähnliches gilt für den Weg zu dem danach genannten Ort. Diese Umformungsschritte können eher als ein Beispiel für Lexikalisierung im eigentlichen Sinn aufgefasst werden denn als Aggregation – obwohl der Aggregationsbegriff von manchen Autoren sehr weit gefasst wird –, da eine Änderung des Prädikats (*pass* statt *go*) vorgenommen wird und sich auch eine modifizierte Argument struktur ergibt (nur die unterwegs passierte Landmarke anstelle von Start- und Zielpunkt).

Um aus dieser Struktur eine für Oberflächengenerierung geeignete Zwischenrepräsentation zu erzeugen, müssen die Aussagen an den Blattknoten, insbesondere die darin vorkommenden Prädikate, lexikalisch ausgedrückt und mit einer linguistischen Umsetzung der Diskursrelationen (Satzbildung und Reihenfolge) kombiniert werden, und es müssen Referenzen zu den Objekten *User*, *A*, *B*, *C* und *D* gebildet werden. Die lexikalische Umsetzung erfolgt weitgehend direkt:

<i>go(from, to)</i>	\rightsquigarrow	<i>weitergehen von ... bis</i>
<i>turn(at, direction)</i>	\rightsquigarrow	<i>abbiegen</i>
<i>pass(by)</i>	\rightsquigarrow	<i>vorbeikommen an</i>

Die einzelnen Aussagen werden dann einfach gemäß ihrer chronologischen Reihenfolge als Hauptsätze aneinandergereiht, wobei aus der **SEQUENCE** Relation ein zeitliches *dann* als explizite Markierung eingefügt wird. Die Herausstellung der **BACKGROUND** Information als eigener Satz ist rhetorisch gut motivierbar. Bei den folgenden Aussagen wird eine Koordination jeweils zweier Hauptsätze vorgenommen, mit Auslassung des Subjekts im jeweils zweiten Teil – dies ist in gewissem Sinn opportunistisch, es gibt aber auch ähnlich gute Ausdrucksmöglichkeiten. Bei der Bildung von Referenzausdrücken wird der Bezug auf den Benutzer (*Sie*), den Standort (*hier*) und auf die zweite Referenz auf *C* (*dort*) mit Pronomen vorgenommen. Für das erste Vorkommen der Landmarken C und D werden Nominalphrasen erzeugt. Diese bestehen aus einer Kategorie (z. B. *die Kirche*) bzw. einer komplexeren Beschreibung (*großes Haus, das Rathaus*), was durch die Annahme motiviert ist, dass die Beschreibung *Kirche* ausreichend zur Identifikation ist, während dies bei *Rathaus* nicht sicher zutrifft.

Die Ergebnisse der Satzplanung können in verschiedenartigen Repräsentationen ausgedrückt werden, wie etwa abstrakte kasusrahmenartige Repräsentationen, lexikalierte Kasusrahmen, oder abstrakte syntaktische Teilbeschreibungen. Die Entscheidung für eine dieser Beschreibungsarten hängt von der Art des Zusammenspiels zwischen Textplanung und Lexikalisierung ab. Die Realisierung dieses Zusammenspiels ist theoretisch noch unzureichend geklärt; sie erfolgt meist opportunistisch durch die Gegebenheiten der Anwendung motiviert (siehe auch die Diskussion der Alternativen in Reiter und Dale 2000). In den in Abbildung 3.56 angegebenen Strukturen wurden abstrakte syntaktische Beschreibungen gewählt.

In der Folge wollen wir Verfahren näher betrachten, mit denen der Aufbau einer Textstruktur ermöglicht oder zumindest wesentlich gesteuert wird, sowie Verfahren zu verschiedenen Aspekten der Satzplanung und Oberflächenrealisierung.

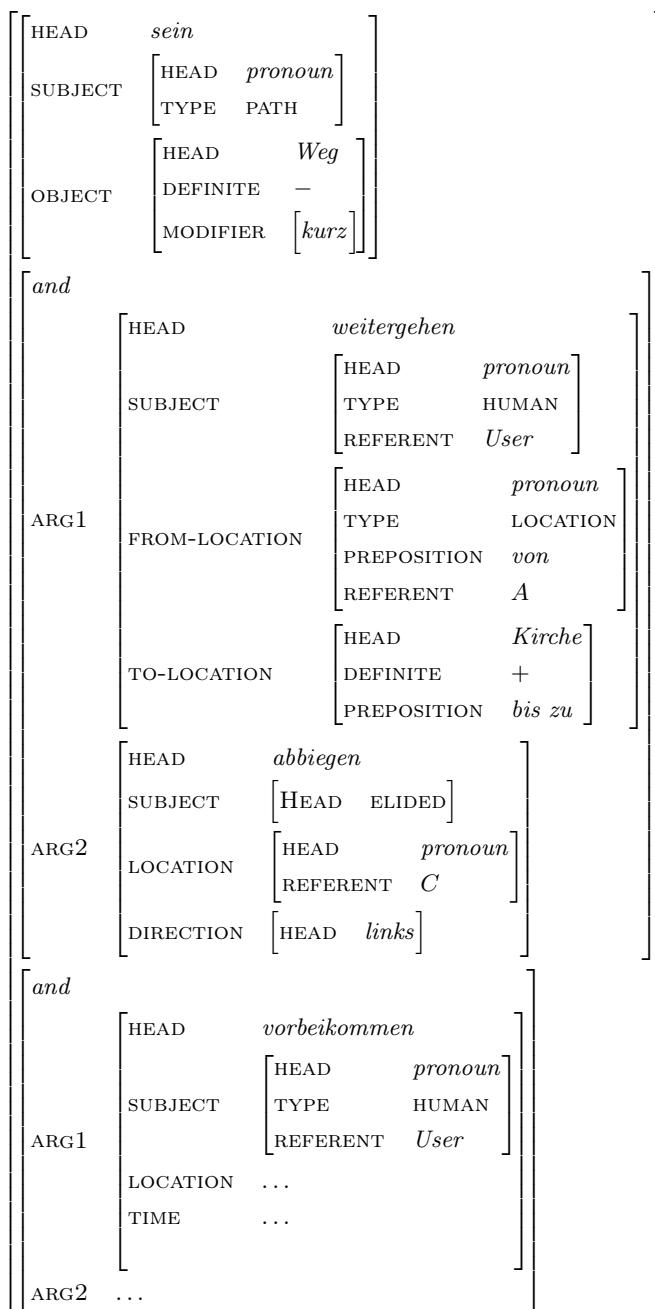


Abbildung 3.56: Ausschnitt aus einer abstrakten syntaktischen Beschreibung der Wegbeschreibung (3.195)

3.8.3 Methoden zur Diskursplanung

Zur Umsetzung der Aufgaben der Diskursplanung sind zwei grundlegende Methoden entwickelt worden:

- **Textschemata**, eingeführt in McKeown (1985a) und McKeown (1985b).
- Schrittweise Planung mit Diskursrelationen, zuerst operationalisiert für Relationen der **Rhetorical Structure Theory** (RST, siehe Abschnitt 3.7.1; Mann und Thompson 1988) von Hovy (1993).

In der Folge beschreiben wir zunächst die Funktionalität jeder dieser Methoden und vergleichen dann ihre Vor- und Nachteile. Die Anwendungsbeschränkungen und einige Ansätze zur Überwindung dieser Beschränkungen behandeln wir in einem separaten Abschnitt.

Textschemata

Schemata sind vordefinierte Repräsentationen von stereotypen Strukturen, die Texten zugrunde liegen, die etwa der Größe eines Absatzes entsprechen. Ein Schema ist als ein Muster definiert, das Bedingungen für den Inhalt und die Reihenfolge von Haupt- und Nebensätzen innerhalb eines Absatzes definiert. Durch die geeignete Spezifikation solcher Schemata – McKeown hat ihre Versionen auf der Basis einer Reihe empirischer Analysen erstellt – kann die erforderliche Kohärenz unterstützt werden. Die Elemente eines Schemas sind rhetorische Prädikate, die die Art von Information charakterisieren, die an der entsprechenden Stelle im Schema angeführt werden kann, und die mit der Semantik der Elemente der Wissensbasis abgeglichen werden müssen. Eine gewisse Variabilität in der Abfolge von Sätzen wird erreicht durch Optionalität einzelner Elemente, Alternativen von rhetorischen Prädikaten sowie Wiederholungen. Zusätzlich können Schemata auch rekursive Definitionen enthalten, wobei anstelle von Referenzen zu rhetorischen Prädikaten Referenzen auf andere Schemata stehen.

Als erste Anwendung in dem System TEXT hat McKeown durch Analyse gemeinsamer Muster verschiedener Korpora vier Diskursstrategien identifiziert und Schemata dafür ausgearbeitet: *constituency*, *identification*, *attributive* und *compare and contrast*. Diese Strategien wurden zur Beantwortung von Fragen über die Struktur einer Datenbank ausgearbeitet, wobei jedes Schema für eine oder mehrere Klassen von Anfragen anwendbar ist. Drei Klassen von Anfragen waren vorgesehen:

1. Anfragen zur Definition eines Konzepts,
2. Anfragen zum Vergleich zweier Objekte und
3. Anfragen zur Beschreibung verfügbarer Informationen.

Das *constituency* Schema etwa, das für Definitionen und Beschreibungen von Objekten anwendbar ist, besteht aus vier Schritten:

1. Identifizierte das zu definierende Konzept als einer generischen Klasse zugehörig oder präsentierte *attributive* Information darüber
2. Präsentiere die Bestandteile (*Konstituenten*) dieses Konzepts
3. Präsentiere *charakteristische Information* über seine Konstituenten
4. Präsentiere *attributive* oder *analoge* Information über das Konzept

In dieser informellen Darstellung beziehen sich die kursiv geschriebenen Phrasen auf die Semantik der rhetorischen Prädikate (wie *Identification*, *Constituency*, *Attributive* und *Analogy*).

Die Instanziierung eines Schemas erfolgt mittels Funktionen, die mit den rhetorischen Prädikaten assoziiert sind. Diese Funktionen wählen Information aus der Wissensbasis aus, die zu dem rhetorischen Prädikat passt, und bildet daraus eine Proposition. Die mit dem Prädikat *Identification* verbundene Funktion etwa liefert für ein zu definierendes Objekt eine Proposition, die das übergeordnete Konzept dieses Objekt, seine definierenden Attribute sowie dieses Objekt selbst enthält. Diese Instanziierung der rhetorischen Prädikate erfolgt schrittweise für das gesamte Schema, gemäß der Folge von rhetorischen Prädikaten in dem Schema. Die Auswahl zwischen alternativen Fortsetzungen, wenn solche an bestimmten Stellen in einem Schema möglich sind, erfolgt auf der Basis von Kriterien über den **Aufmerksamkeitsfokus**, der sich auf jenes Objekt in einem Satz bezieht, auf das der Äußernde besondere Aufmerksamkeit legt. Zur Umsetzung wurden Bedingungen, wann sich der Aufmerksamkeitsfokus von einem Satz zum nächsten ändern kann, durch Prioritäten ausgedrückt. Auf Grund dieser Prioritäten wird versucht, falls möglich, den Aufmerksamkeitsfokus auf ein eben in den Diskurs eingeführtes Objekt zu verschieben, andernfalls wird der aktuelle Aufmerksamkeitsfokus beibehalten und nur, falls dies auch nicht möglich ist, auf einen früheren Fokus zurückgegriffen. Daraufhin wird der Aufmerksamkeitsfokus auch herangezogen, um zwischen Alternativen bei der Realisierung der Propositionen auszuwählen.

In unserer Anwendung, Wegbeschreibung, könnten die zwei Textschemata **Gesamtweg** und **Wegabschnitt** in Abbildung 3.57 einfache Formen von Domänenexten beschreiben. Dabei drückt + mögliche Wiederholung und | eine Alternative aus.

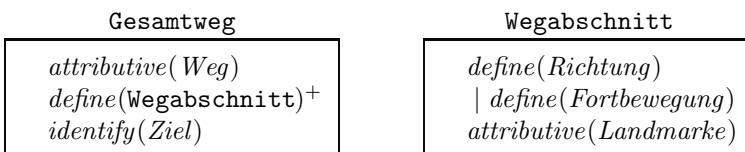


Abbildung 3.57: Textschemata zur Wegbeschreibung (3.195)

Das Schema **Gesamtweg** definiert eine Wegbeschreibung als eine Folge von Wegabschnittsbeschreibungen, der optional ein charakteristisches Attribut des Ge-

samtwegs vorangeht, und die mit der Identifikation des Zielorts abgeschlossen wird. Die Wegabschnitte selbst, wie in dem Schema **Wegabschnitt** definiert, können durch eine Richtungsänderung oder eine Fortbewegung beschrieben werden, optional ergänzt durch eine Landmarke. Diese Aufteilung erweist sich als pragmatisch leicht handhabbar, weicht aber leicht von der rhetorisch organisierten Textstruktur (Abbildung 3.55) ab.

Pragmatische Faktoren wie Erfordernisse an Ausführlichkeit und Explizitheit bestimmen die Auswahl bei Optionen in einem Schema bei dieser Anwendung in viel stärkerem Maße, als bei den Beschreibungen von Datenbankkonzepten. Die geeignete Verwendung des Schemas **Gesamtweg** etwa gestaltet sich unterschiedlich in Abhängigkeit von verschiedenen Eigenschaften, wie Weglänge und Einfachheit. Bei kurzen oder weitgehend geraden Wegen kann man sich auf Wegabschnittbeschreibungen beschränken, während bei längeren Wegen eine kurze Wegcharakterisierung vorweg meist für den Auskunftssuchenden hilfreich ist. Bei unübersichtlichem Ausgangsort oder schwer identifizierbarem Zielort erweisen sich Beschreibungen dieser Orte als erforderlich. Gleichermaßen gilt für die Beschreibung von Landmarken im Verlauf des Wegs bei einer Wegabschnittbeschreibung. Es zeigt sich demnach in diesem Anwendungsbereich, dass die Auswahl zwischen Alternativen weitgehend durch Kontext und Konventionen bestimmt sind, während rhetorische Kriterien vorrangig die Komposition der einzelnen Aussagen regeln.

Bei der bereits oben eingeführten Spezifikation von kommunikativer Absicht c und Wissensbasis k könnten diese beiden Schemata in einer Situation d – Frage nach dem Weg von A nach B – zur Erzeugung eines Diskursplans führen, der in späteren Phasen der Generierung zu dem eingangs angegebenen Text (3.195) umgesetzt werden kann.

Für unser Beispiel wollen wir annehmen, dass der Adressat eine vollständige Beschreibung bei ihm unbekannter Umgebung wünscht. Deshalb erfolgt die Instanziierung der optionalen Wegcharakterisierung (*attributive(Weg)*), das erste rhetorische Prädikat im Schema **Gesamtweg**) durch *short(path(A, B))*. Das folgende rhetorische Prädikat, *describe(Wegabschnitt)*, ist die Referenz zu einem Schema, zu dem die Wissensbasis Material für drei Instanziierungen enthält (die Wegabschnitte A nach C , C nach D und D nach A). Die Anordnung der daraus erzeugten Propositionen erfolgt am besten chronologisch nach domänen spezifischen Kriterien – dies entspricht auch allgemeinen Fokuskriterien – bei Erreichen des Ziels eines Abschnitts wird ein neuer Fokus eingeführt, der im folgenden Abschnitt durch den gleichen Start beibehalten wird. Die einzelnen Abschnitte können dann unterschiedlich behandelt werden, gemäß den Alternativen der Definition des Schemas **Wegabschnitt**. Der erste Abschnitt (von A nach C) wird durch eine Fortbewegung beschrieben, zusätzlich mit einer Landmarke versehen (die Kirche). Die nächsten beiden Abschnitte werden zusammengefasst – es findet bei D keine Richtungsänderung statt – und durch die Richtungsänderung bei C sowie eine darauffolgende Beschreibung der Landmarke D realisiert. Diese Zusammenfassung erfolgt opportunistisch während der Schemainstanzierung und verbessert wesentlich die Qualität der Textspezifikation durch inhaltliche

Kompaktheit und Variation des Ausdrucks. Schließlich wird die Option zum Ausdrücken des letzten Prädikats im Schema, Zielcharakterisierung, wahrgenommen. Die Instanziierungen der Schemata sehen dann wie in Abbildung 3.58 aus.

Rhetorische Elemente	Instanziertes Schema
$attributive(Weg)$ <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> $define(Fortbewegung)$ $attributive(Landmarke)$ </div> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> $define(Richtung)$ $attributive(Landmarke)$ </div> $identify(Ziel)$	$short(path(A, B))$ $go(User, from(A), to(church))$ $-$ $turn(User, at(church), direction(left))$ $pass(User, by(big \& building \& town-hall))$ $be(User, at(B))$

Abbildung 3.58: Instanziertes Schema

Eine geeignete sprachliche Umsetzung der Folge von Prädikationen in dem instanzierten Schema könnte dann zur Erzeugung des eingangs angeführten Beispieltextrs (3.195) führen. Dabei ist zu beachten, dass die Prädikationen selbst nur eine Folge von Aussagen darstellen, während sich die Relationen zwischen ihnen, die die Gestaltung des Textes mitbestimmen, nur in den Schemadefinitionen manifestieren.

Schrittweise Planung mit Diskursrelationen

Eine alternative Möglichkeit zur Beschreibung des gesamten Aufbaus kohärenter Diskurse, wie sie durch Schemata definiert wird, besteht in der formalen Beschreibung ihrer elementaren Teile, der Diskursrelationen (siehe Abschnitt 3.7.1). Diskursrelationen verbinden zwei oder mehrere Textsegmente zu einer Einheit, die wiederum mittels einer Diskursrelation in Bezug gesetzt werden kann zu anderen elementaren oder zusammengesetzten Textsegmenten. Eine in dieser Weise zusammengesetzte Struktur bildet dann einen Diskursstrukturbau, wobei die inneren Knoten und insbesondere die Wurzel Diskursrelationen repräsentieren und die Endknoten elementare Aussagen (Sprechakte). Um zu gewährleisten, dass ein solcher Baum kohärent aufgebaut ist, müssen an die Diskursrelationen Bedingungen geknüpft werden, die die Zusammensetzung von Textteilen regeln, in Abhängigkeit von der Art der Relation (wie etwa in der RST). So wird z. B. gefordert, dass die einzelnen Textteile, die mittels einer Relation **SEQUENCE** zu einer zeitlichen Sequenz zusammengebaut werden sollen, in der ihrer Position in der Sequenz entsprechenden chronologischen Ordnung stehen.

Der Aufbau eines Diskursstrukturbauks für eine Wegbeschreibung erfolgt nun durch Anwendung von Operatoren, die dieses Diskursziel, die Beschreibung des Gesamtwegs, solange zerlegen, bis die einzelnen Teilziele durch elementare Aussagen über Wegabschnitte erfüllt werden können. Dies erfolgt in gewissem Sinn

hierarchisch, wobei zunächst eine Grobstruktur der Wegbeschreibung gebildet und später in ihren Teilen verfeinert wird. Dazu können Teile der Beschreibungen in einzelnen Abschnitten, gebildet mittels mehrfacher Anwendung des **SEQUENCE**-Operators, ergänzt werden, etwa mittels einer **ELABORATION** durch Bezug auf Richtungsänderungen oder Landmarken. Zusätzlich können Erläuterungen über den Weg als Gesamtes zum besseren Verständnis integriert werden, wie **BACKGROUND** Information oder eine Anmerkung über das Erreichen des Ziels als **EVIDENCE**.

Konkret wird zunächst das ursprüngliche Ziel (*Beschreibe den Weg von A nach B*) näher untersucht. Wenn es sich hierbei nicht um einen genügend kurzen Abschnitt handelt (ein anwendungsspezifisch auswertbares Kriterium), der einfach beschrieben werden kann, wird dieses Ziel in eine Sequenz zerlegt, und zwar in die Teilziele *Beschreibe den Weg von A nach C* und *Beschreibe den Weg von C nach B*. Da beide Abschnitte selbst keine Richtungsänderung beinhalten, wollen wir davon ausgehen, dass sie nicht weiter zerlegt werden. Die Beschreibungen der Teilabschnitte können nun, durch Eigenschaften der Wegabschnitte bestimmt, als Fortbewegung (*Sie gehen von hier bis zur Kirche* für $\text{path}(A, C)$ und *Dann kommen Sie an einem großen Haus, dem Rathaus, vorbei* für $\text{path}(C, B)$) verbalisiert werden. Der erste Abschnitt sollte, wegen der anschließenden Richtungsänderung, durch diese ergänzt werden, realisiert durch eine **ELABORATION**. Dadurch wird der Diskursstrukturbau an einem seiner Blattknoten erweitert. Schließlich kann durch Einfügen von **EVIDENCE** (Erreichen des Ziels) und **BACKGROUND** (charakteristische Eigenschaft des Gesamtwegs) Information in Bezug auf die gesamte Wegstrecke hinzugefügt werden. Zum Unterschied von der vorigen Erweiterung erfolgen diese Einfügungen an inneren Knoten des Diskursstrukturbaums, im speziellen Fall sogar an der Wurzel. Der somit entstandene Diskursstrukturbau sieht dann wie in Abbildung 3.59 aus. Diese Struktur ist eine notationelle Variante des als Merkmalsstruktur angegebenen Textplans in Abbildung 3.55, wobei hier allerdings an den Blattknoten Teilzielspezifikationen anstelle von Satzteilspezifikationen stehen.

Die elementaren Teilpläne an den Blattknoten werden, wie an einzelnen Fällen oben beschrieben, durch Satzteile (Haupt- oder Nebensätze) ausgedrückt. Zur kohärenten Verbindung dieser Teile werden die rhetorischen Relationen herangezogen, sowohl zur Entscheidung zwischen Haupt- und Nebensatz, als auch zur Wahl von Konnektiven. Die Relation **SEQUENCE** wird beispielsweise durch Verbindung von zwei Hauptsätzen mit *und* oder durch zwei getrennte Hauptsätze realisiert, wobei der zweite mit *dann* eingeleitet wird.

Darüber hinaus gibt es einige rhetorisch motivierte Kriterien der kontextuell geeigneten Realisierung der einzelnen Textsegmente. Diese Kriterien enthalten generelle und eigentlich offensichtliche Regularien, wie Nähe der durch eine rhetorische Relation verbundenen Satzteile, bevorzugte Aggregation von Propositionen je größer die Anzahl gemeinsamer Konstituenten ist, oder Vermeidung

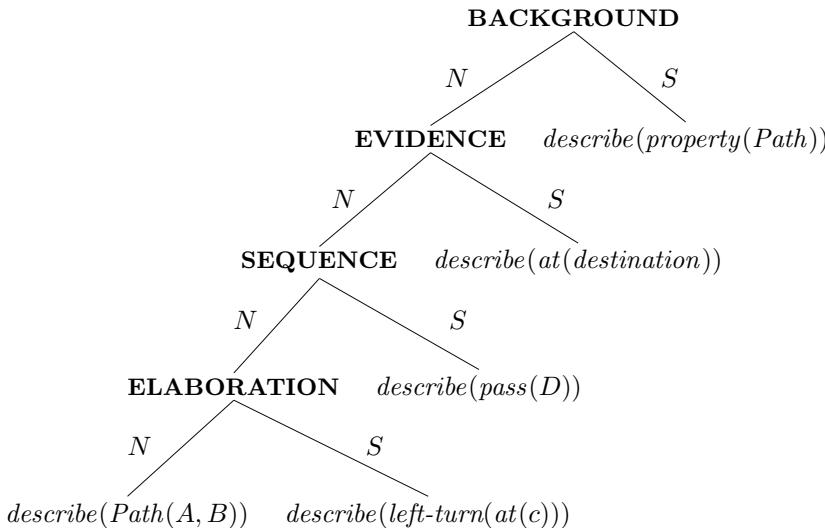


Abbildung 3.59: Textstrukturbbaum mit rhetorischen Relationen für die Wegbeschreibung (3.195)

von Ambiguitäten, deren Einhaltung allerdings nicht immer leicht in einem formalen Generierungssystem zu erkennen ist. Andere Kriterien sind leichter formal zu überprüfen, etwa, dass Einbettungen auf **ELABORATION** Relationen beschränkt bleiben sollen (wobei jedoch der Satellit, der eingebettet wird, wiederum keine **LIST** Relation sein sollte). Wie bereits an früherer Stelle angedeutet, machen diese Kriterien deutlich, dass sie zwar notwendig sind in dem Sinne, dass sie einige offensichtlich unerwünschte Realisierungen ausschließen, aber nicht hinreichend, um wirklich qualitativ gute Ausdrucksvarianten zu ermöglichen.

Die Techniken der Planung mit Diskursrelationen sind schließlich auch auf Multimediaumgebungen erweitert worden. Dabei können die mit den Blattknoten assoziierten kommunikativen Teilziele entweder durch Textteile oder durch graphische Elemente realisiert werden, je nach Eignung der Medien. In der Anwendung zur Beschreibung der Funktionalität einer Espressomaschine etwa wird **BACKGROUND** Information zur Lokalisierung von Teilen der Maschine durch einen Bildausschnitt angegeben, während ein Hinweis zur Wassertemperatur sprachlich ausgedrückt wird. Die Koordination der Medien erweist sich dabei als eine wichtige Aufgabe, die mit cross-medialen Referenzausdrücken unterstützt wird.

RST und darauf basierende Planungsverfahren treffen einige vereinfachende Annahmen, die die Ausdrucksmöglichkeiten beschränken und für stereotype Erscheinungsformen automatisch erzeugter Texte mitverantwortlich sind.

- Textsegmente sind immer Satzteile (Haupt- oder Nebensatz), aber keine Nominal- oder Präpositionalphrasen, eine rein pragmatische, nicht linguistisch motivierte Einschränkung.

- Es besteht die Forderung nach einer einzigen, prominenten Relation zwischen je zwei Propositionen. In manchen Fällen bestehen sowohl informationelle Relationen (d.h. semantische), wie **SEQUENCE** und **CAUSE**, zusätzlich zu einer intentionalen wie **MOTIVATION**.
- Alle Spezifikationen werden entsprechend der vorliegenden Textstruktur explizit ausgedrückt. Dies würde manchmal zu unnatürlichen Satzfolgen, etwa bei Aufzählungen, führen, bzw. zu als redundant empfundenen Beschreibungen, speziell in inferenzreichen Diskursen, wie etwa Erklärungen.

Praktische Systeme versuchen diese Probleme weitgehend durch geschickte, oft anwendungsspezifisch motivierte Inhaltsauswahl und flexible Lexikalisierung zu umgehen.

Eignung der Verfahren

Im Gebiet der Sprachgenerierung wurden Textschemata und das Planen mit Diskursrelationen einige Zeit als alternative Verfahren in einer gewissen Konkurrenzsituation angesehen. Mittlerweile hat sich die Ansicht durchgesetzt, dass Schemata in gewissem Sinne auskompilierte Zusammensetzungen von Diskursrelationen sind.

In Bezug auf die Aufgaben der Textplanung leisten Schemata und Planung mit Diskursrelationen vorrangig die Organisation von zu präsentierenden Inhalten, durch die Reihenfolge der im Laufe der Schemainstanziierungen gebildeten Propositionen. Beziiglich der Inhaltsauswahl sind beide Verfahren in gewissem Sinne neutral. Die Alternativen, Optionalitäten und Wiederholungen in den Schemadeinitionen bzw. die partielle Reihenfolge der Aussagen in den Diskursstrukturbäumen, lassen im Prinzip verschiedene Teilmengen der maximal auf diese Definitionen passenden Inhalte der Wissensbasis zu. Die implizite Defaultstrategie bei der Schemainstanziierung ist jedoch die Auswahl aller zu den Schemaspezifikationen passenden Elemente der Wissensbasis. Ähnliches gilt für das Planen mit Diskursrelationen. Eine echte Auswahl kann demnach entweder durch eine Vorselektion von Teilen der Wissensbasis oder durch zusätzliche, eventuell kontextabhängige Selektionsmechanismen bei der Schemainstanziierung erfolgen.

In noch stärkerem Maß als bei den Schemata gibt es eine große Zahl von Möglichkeiten, wie eine bestehende Diskursstruktur erweitert werden kann. Die in einer Anwendung gewählten Varianten in den Schemata bzw. in zur Erweiterung einer partiellen Diskursstruktur oder zum Weglassen von Information sind oft mehr durch Adressateneigenschaften und Domänenkonventionen bestimmt als durch generelle rhetorische Kriterien. Die Reihenfolge bei Schemainstanziierungen und Textplanexpansionen erfolgt am besten opportunistisch, da größere Strukturen mehr Information über Eignungskriterien enthalten. Praktische Textplanungsverfahren arbeiten jedoch meist strikt top-down und chronologisch bei Schemata.

Generell erwiesen sich jedoch Schemata durch die explizite Festlegung der möglichen Alternativen und des Gesamtaufbaus für manche Verwendungen als

zu unflexibel, da sie es nicht erlauben, die ihrem Aufbau und entsprechenden Instanziierungen zu Grunde liegenden Absichten darzustellen und daraus Schlüsse zu ziehen. Die generell, allgemein akzeptierte Einschätzung der beiden Verfahren ist, dass inkrementelles Planen mit Diskurs flexibler ist als der Einsatz von Schemata, aber auch schwieriger zu kontrollieren. In der Praxis kann man manchmal kombinierte Ansätze finden, wobei Schemata für weniger variantenreiche Textteile eingesetzt werden und inkrementelle Planung für die variantenreicheren. Ein typisches Beispiel für eine solche Mischform sind standardisierte Geschäftsbriefe.

3.8.4 Satzplanungsverfahren

In diesem Abschnitt werden kurz die wesentlichsten Verfahren zur Bewältigung von Aufgaben der Satzplanung vorgestellt. Die gewählte Reihenfolge impliziert dabei keineswegs einen präferierten Ablauf in einem umfassenden Satzplanungsprozess.

Lexikalisierung

Für die in den Propositionen des Textplans aufscheinenden Prädikate müssen lexikalische Elemente der Zielsprache (Lexeme, grammatische Funktionen und Merkmale) gewählt werden, die den Informationsgehalt dieser Prädikate ausdrücken. Dieser Vorgang, die Lexikalisierung im engeren Sinn, muss in einer kompositionalen Weise erfolgen, so dass die einzelnen lexikalischen Elemente eine korrekte Struktur in der Zielsprache bilden. Außerdem müssen die lexikalischen Ausdrücke für den pragmatischen Kontext angemessen sein. Dies ist in Bezug auf Kompositionalität oft nicht einfach, da die Abbildung zwischen Prädikaten im Textplan und lexikalischen Elementen nicht immer eine 1:1-Abbildung ist. Weiter müssen auch einsprachspezifische Besonderheiten beim Zusammentreffen von Wörtern mit kontextabhängiger Bedeutung berücksichtigt werden (sogenannte Kollokationen). Z.B. wird im Deutschen ein Mensch, der viele Zigaretten konsumiert, als *starker Raucher* bezeichnet; die Entsprechungen des Adjektivs differieren jedoch sowohl im Englischen *heavy* (wörtlich: *schwer*) *smoker* als auch im Französischen *grand* (wörtlich: *groß*) *fumeur*.

Je nach Reichhaltigkeit der Entsprechungen zwischen Prädikaten im Textplan und lexikalischen Elementen stehen oft mehrere Alternativen bei der Lexikalisierung zur Verfügung. Die Auswahl zwischen diesen wird durch verschiedene Einflussfaktoren gesteuert:

- Konnotationen von Wörtern müssen berücksichtigt werden. So ist etwa das Bevorzugen von *Köter* gegenüber dem neutralen *Hund* nur dann gerechtfertigt, wenn eine negative Einstellung des Sprechers gegenüber dem Tier ausgedrückt werden soll. Ähnliches gilt für die bevorzugte Verwendung von Basiskategorien – *Pudel* wird nur dann statt *Hund* verwendet, wenn die Information über die Rasse in der Situation wesentlich ist.
- Die *Perspektive* auf Sachverhalte kann durch die Wortwahl verändert werden. Mit der Wahl von *kaufen* bzw. *Kauf* wird die Perspektive des Erwerbs auf den Käufer gelegt, mit *verkaufen* bzw. *Verkauf* auf den Verkäufer.

- Ein kompensatorischer Effekt besteht zwischen Wiederholungen und knappen Ausdrucksformen. Während die erste Variante auf verbesserte Verständlichkeit abzielt, ermöglicht die zweite kürzere Texte. Manche Formen von Wiederholungen können im Laufe weiterer Verarbeitung kompakter ausgedrückt werden (durch Aggregierung, siehe nächsten Abschnitt).
- Ein generelles Kriterium für die Auswahl zwischen Lexemen bilden die Eigenschaften der **kontextuellen Einbettung**, die den Adressaten der Äußerung und die Stellung im Diskurs umfassen. Dazu gehören die Vermeidung von Wörtern, die dem Adressaten nicht bekannt sind, und Konsistenz mit textuellen Eigenschaften des vorangegangenen Diskurses.

Verfahren zur Lexikalisierung sind meist musterbasierte Abbildungen zwischen zwei Repräsentationsebenen: den konzeptuellen Beschreibungen als Ausgangsspezifikationen (hier: dem Textplan) und den lexikalischen Strukturen als Ziel. Dabei sind viele Entsprechungen elementweise definiert, manchmal wird eine Zusammensetzung von konzeptuellen Elementen auf ein Lexem abgebildet (etwa bei semantisch komplexen Wörtern) oder ein konzeptuelles Element auf eine sprachliche Struktur (etwa bei idiomatischen Ausdrücken). Die Kompositionalität ist auf den jeweiligen Repräsentationsebenen definiert; zusammenhängende Strukturen auf der konzeptuellen Ebene werden ohne Überlappungen und Lücken zerlegt, so dass zur Abbildung jedes dabei entstehenden Teils mindestens ein Muster definiert ist. Die Bilder dieser Muster auf der Seite der Zielsprache müssen dann gemäß semantischen und grammatischen Erfordernissen zusammensetzbare sein; das heißt unter anderem, Selektionsrestriktionen und Typbedingungen von Phrasenköpfen müssen durch die jeweils abhängigen Strukturen erfüllt sein. Falls dies an einer Stelle unzutreffend ist, muss eine andere Kombination von Abbildungen auf lexikalische Elemente genommen werden.

Die Regelung der Zusammensetzbarkeit ist gerade bei komplexeren Abbildungen eine Herausforderung. Horacek (1996) zeigt, wie metonymische Relationen je nach Zielsprache implizit oder explizit ausgedrückt werden können. Stede (1996) ist in der Lage, recht unterschiedliche Aufteilungen des Informationsgehalts auf die gewählten Lexeme vorzunehmen. Dadurch kann etwa aus derselben konzeptuellen Repräsentation das im Englischen bessere *he swam across the river* (dt. *er schwamm über den Fluss*) und das im Französischen bessere *il a traverse la riviere a la nage* (dt. *er hat den Fluss schwimmend überquert*) erzeugt werden.

Bei dem Beispiel zur Wegbeschreibung liegen solch komplizierte Fälle nicht vor. Die konzeptuellen Prädikate werden im Wesentlichen auf geeignete Lexeme 1:1 abgebildet. Eine Ausnahme ist das Prädikat *left-turn*, das durch das Verb *abbiegen* zusammen mit der Ergänzung *links* ausgedrückt wird.

Aggregation

Unter Aggregation versteht man im Rahmen der Sprachgenerierung das Erzeugen von Satzrepräsentationen aus satzteilartigen (engl. *clauses*) semantischen Repräsentationen. Dabei werden insbesondere Koordinationsphänomene berücksichtigt, so dass Redundanzen, die sich durch mehrere semantische Repräsentan-

tionen gemeinsamer Teile ergeben, kompakt ausgedrückt werden können. Relevante Phänomene sind die Koordination von Konstituenten und verschiedene Arten von Konstituententilgungen. Koordination von Konstituenten kann erfolgen, wenn sich zwei Aussagen nur in einem der beteiligten Objekte unterscheiden, wie in der Erzeugung des Satzes *Hans mag Maria und Anna* aus den Aussagen *Hans mag Maria* und *Hans mag Anna*. Dabei ist jedoch eine mögliche Interferenz mit lexikalischer Semantik bei bestimmten Verben zu berücksichtigen, da in Sätzen wie *Hans und Maria gehen auf eine Party* die kollektive Lesart präferiert wird, die unter Umständen nicht den getrennten Aussagen der Ausgangspräsentation entspricht. Bei Konstituententilgungen werden nach van Oirschot (1987) einige Regeln für Tilgungen bei Gleichheit unterschieden, wie Gapping in *Hans isst Fisch und Willi _ Reis*, Rechtsknotentilgung in *Hans fing _ und Willi aß den Fisch*, Verbphrasentilgung in *Hans isst Fisch und Willi _ auch* und Konstituentenreduktion in *Hans isst Fisch zu Mittag und _ Reis am Abend*. Ein automatisches Verfahren muss somit nicht nur Aussagen geeignet zusammenfügen, sondern beim Durchführen der Tilgungen diese sprachlichen Phänomene berücksichtigen, was existierende Systeme nur zum Teil tun.

Wir wollen in diesem Zusammenhang das Verfahren von Shaw (1998a) bzw. Shaw (1998b) betrachten, das die meisten der oben genannten Phänomene berücksichtigt. Das Verfahren setzt sich aus vier Phasen zusammen:

1. Zunächst werden die Propositionen, die die semantischen Repräsentationen beinhalten, sortiert, wobei allerdings pragmatische und kontextuelle Einschränkungen, wie rhetorische Relationen zwischen einzelnen Aussagen, berücksichtigt werden. Die Sortierung erfolgt nach Graden der Ähnlichkeit der Propositionen, beginnend mit jenem Argument, das die kleinste Anzahl von Werten in allen Propositionen hat. Dadurch wird eine Sortierung erreicht, bei der sich zwei unmittelbar aufeinander folgende Propositionen möglichst wenig unterscheiden.
2. Als nächstes werden sequentiell jeweils zwei aufeinanderfolgende Propositionen verglichen, die identischen Elemente zunächst nur markiert und daraus eine gemeinsame Aussage gebildet. Dadurch ergibt sich gegenüber anderen Ansätzen, die gleichzeitig Tilgungsoperationen einführen, eine erhöhte Flexibilität.
3. Diese Markierung erfolgt gemäß parameterisierbarer Komplexitätsgrenzen; so werden zwar mehrere Aussagen, die sich nur in den Werten eines Arguments unterscheiden, zu einer koordinierten Aussage zusammengefasst, jedoch nur maximal zwei Aussagen, die sich in den Werten mehrerer Argumente unterscheiden. Dieses Vorgehen führt zur Festlegung von Satzgrenzen.
4. Schließlich erfolgt die Tilgung redundanter Elemente. Dazu werden syntaktisches Wissen und Information über die Reihenfolge von Konstituenten herangezogen. Dadurch können Fälle von Vorwärtstilgung, wie in *On Monday, Al re-stocked coffee and [on Monday,] [Al] removed wrotten milk*

unterschiedlich zu *Al re-stocked coffee [on Monday,] and [Al] removed wrotten milk on Monday* behandelt werden (Beispiele aus Shaw 1998a).

In unserer Wegbeschreibung spielen solche Aggregationsoperationen keine große Rolle. Die Festlegung der Reihenfolge der Sätze und der Satzgrenzen ist von Besonderheiten der Domäne beeinflusst. Die Hintergrundinformation wird zuerst als eigener Satz ausgedrückt. Darauf werden die beiden Aussagen über die gerade Fortbewegung und die Richtungsänderung zusammengefasst, wobei im zweiten Teil das Subjekt getilgt werden kann. Ebenso wird mit der Aussage über die Landmarke und der Evidenz über das Erreichen des Ziels verfahren. Es zeigt sich hier, dass diese Zusammenfassungen eher opportunistisch durch Aggregationsmöglichkeiten und vom Nutzen der Information – Kenntnis der Weglänge – motiviert sind als von der darunterliegenden rhetorischen Struktur.

Referenzausdrücke

Die Propositionen im Textplan enthalten Referenzen auf Objekte, die mindestens einen von zwei grundlegenden Zwecken erfüllen sollen:

- Identifikation des Objekts (durch eine referentielle Beschreibung)
- Information über ein Objekt (durch eine attributive Beschreibung)

Für die Realisierung dieser Aufgaben stehen mehrere sprachliche Ausdrucksmitel zur Verfügung, mit unterschiedlicher Eignung je nach Kontext. Dazu gehören Pronomen, Eigennamen, indefinite und definite Beschreibungen. Letztere sind auch in reduzierter Form möglich. Bei attributiven Referenzen werden spezifizierte Informationen in sprachliche Ausdrücke überführt und im Zusammenhang mit dem zu beschreibenden Objekt genannt. Dies erfolgt opportunistisch im Kontext des gesamten Textes und ist somit eher eine Aufgabe für die Lexikalisierung und, bei umfangreicher Information, auch für die Textplanung. Die Generierung referentieller Beschreibungen hingegen erfordert die Erzeugung eines sprachlichen Ausdrucks, von dem plausibel angenommen werden kann, dass der Adressat den Referenten dadurch eindeutig identifizieren kann. Da dieses Objekt intern einfach durch eine zugeordnete Variable identifiziert ist, erfordert das Finden einer geeigneten referentiellen Beschreibung somit in der Bestimmung zusätzlicher Angaben.

Die einfachste Referenz auf ein Objekt ist jene mit einem Pronomen. Die Entscheidung allerdings, ob eine solche Referenz im Einzelfall adäquat ist oder nicht, kann je nach Kontext sehr heikel und subtil sein. Daher ist für automatische Verfahren ein konservatives Vorgehen wie in Reiter und Dale (2000) zu empfehlen, durch das Pronomen möglichst selten in inadäquaten Situationen erzeugt werden, dafür öfters kein Pronomen gewählt wird, obwohl es in den entsprechenden Situationen gut wäre. Das einfache Kriterium besagt, dass ein Pronomen nur dann zu wählen ist, wenn seine grammatischen Eigenschaften (Person, Numerus und Geschlecht) nur auf den intendierten Referenten und auf kein anderes Objekt im selben oder im unmittelbar vorangegangenen Satz passen.

Für nicht-pronominale referentielle Beschreibungen sind mehrere ähnliche Verfahren angegeben worden, wobei die neueren meist Verfeinerungen und Erweiterungen der grundlegenden Methode in Dale und Reiter (1995) sind. Die Auswahl der Beschreibungselemente erfolgt nach drei komplementären Anforderungen:

- Der Referenzausdruck muss adäquat sein in dem Sinne, dass er genügend Informationen zur Identifikation des intendierten Referenten enthält.
- Der Referenzausdruck muss effizient sein, also keine für die Aufgabe unnötige Information enthalten.
- Der Referenzausdruck muss sensitiv in Bezug auf die Kenntnisse und Fähigkeiten des Adressaten sein.

Speziell die geeignete Interpretation des Effizienzkriteriums hat die Forscher einige Jahre beschäftigt. Zunächst wurde versucht, möglichst minimale Beschreibungen zu erzeugen, was sich jedoch als sehr rechenaufwändig herausstellte. Motiviert durch die empirische Erkenntnis, dass Menschen selbst oft redundante Deskriptoren in Referenzausdrücken verwenden, wurden einfachere Verfahren vorgeschlagen, die psychologischen Erkenntnissen besser Rechnung tragen. Das grundlegende Vorgehen ist dabei wie folgt: Es werden die Mengen aller potentiell mit dem intendierten Referenten verwechselbaren Objekte im aktuellen Diskursabschnitt bestimmt. Des Weiteren werden alle Deskriptoren des intendierten Referenten gesammelt und nach domänenspezifischen Präferenzen in eine Liste sortiert. Dann wird die zu erzeugende Beschreibung aus Elementen dieser Liste aufgebaut. Dabei wird für jeden Deskriptor inkrementell geprüft, ob er auf mindestens eines der potentiell mit dem intendierten Referenten verwechselbaren Objekte nicht zutrifft, auf das die bisher erzeugte Beschreibung hingegen passt. Ein solcher Deskriptor schließt somit die Verwechslung mit mindestens einem anderen Objekt aus und wird daher zu der Beschreibung hinzugefügt. Dieses Verfahren wird solange wiederholt, bis alle potentiell mit dem intendierten Referenten verwechselbaren Objekte ausgeschlossen werden können (der Erfolgsfall) oder keine Deskriptoren mehr zur Verfügung stehen (dann ist mit diesen Mitteln auch keine identifizierende Beschreibung möglich).

Die Erzeugung von Referenzausdrücken ist aktuell ein intensiv bearbeiteter Themenbereich in der Sprachgenerierung. Vorrangig geht es dabei um die Orientierung an menschlichen Präferenzen, bei einer größeren Auswahl von Beschreibungsattributen, und in Bezug auf die Berücksichtigung von logisch redundanten, aber sehr auffälligen Attributten; diese Maßnahmen sind im Basisalgorithmus im wesentlichen unspezifiziert. Darüber hinaus wurde auch die Ausdrucksmächtigkeit der Referenzgenerierung verbessert, in Bezug auf Referenzen auf Mengen von Objekten, durch koordinierte Ausdrücke, Berücksichtigung hierarchischer Kontexte und Vagheiten.

Beispiel 3.8.4

Im Wegbeschreibungstext (3.195) kommen mehrere pronominale Referenzen und zwei referenzielle Beschreibungen vor. Dabei sind die pronominalen Referenzen auf die auskunfts suchende Person mit *Sie* mit den oben skizzierten, einfachen

Verfahren zur Erzeugung pronominaler Referenzen erklärbar wegen der Eindeutigkeit der grammatischen Person. Die Erzeugung der lokalen Referenzen, *hier* und *dort*, bedarf spezieller Algorithmen mit Wissen über räumliches Schließen. Um die Anwendung des oben skizzierten Verfahrens zur Erzeugung referenzieller Beschreibungen für die beiden Landmarken zu erklären, die sich in den Ausdrücken *Kirche* bzw. *ein großes Haus, das Rathaus* manifestiert, sind einige Annahmen zugrunde zu legen. Als Attribute stehen in der Deskriptorenliste die Kategorien der Landmarken, *Kirche* bzw. *Rathaus* an vorderster Stelle, gefolgt von Größe als einer der auffälligsten Eigenschaften von Gebäuden. Zur Identifikation der Kirche kann angenommen werden, dass das Nennen der Kategorie zur Identifikation ausreicht – wenn sie die Gestalt einer typischen Kirche hat und sich keine weitere Kirche in der Nähe befindet. Der Fall des Rathauses ist problematischer. Da nicht einfach die Kategorie des Gebäudes genannt wird, ist davon auszugehen, dass der Auskunftssuchende dieses Rathaus nicht kennt. Also wird nur die allgemeinere Kategorie, Haus, genannt, und das nächstauffällige Attribut, die Größe, angeführt. Das weitere Anführen der Bezeichnung *Rathaus* lässt sich dann nur durch die mit der quantitativen, vagen Beschreibung *groß* verbundenen Unsicherheit erklären, in der Hoffnung, dass dem Auskunftssuchenden die Identifikation durch Heranziehen von stereotypem Wissen über Rathäuser und hypothetischem Schließen in der lokalen Situation erleichtert wird. □

Diese Überlegungen reichen jedoch weit über die konzeptuelle Abdeckung aller existierender Referenzgenerierungsalgorithmen hinaus. Es zeigt sich somit, dass die stark abstrahierenden Verfahren für Realweltsituationen mit komplexen Objekten unzureichend sind, vor allem wegen der Nichtmodellierung des Interpretationspielraums von Attributen und den daraus folgenden Unsicherheiten.

Die Erzeugung von Referenzausdrücken ist derzeit ein intensiv bearbeiteter Themenbereich in der Textgenerierung. Vorrangig geht es dabei um die Orientierung an menschlichen Präferenzen bei größerer Auswahl von Beschreibungsattributen und bei der Berücksichtigung von logisch redundanten, aber auffälligen Attributen zur leichten Objekterkennung. Diese Kriterien sind in den Basisalgorithmen im wesentlichen nicht spezifiziert. Darüberhinaus wurde auch die Ausdrucksstärke der Referenzen verbessert, in Bezug auf Referenzen auf Mengen von Objekten durch koordinierte Ausdrücke, Berücksichtigung von hierarchisch organisierten Strukturen, sowie Vagheiten.

3.8.5 Verfahren zur Oberflächenrealisierung

Sobald die Verfahren der Satzplanung angewendet wurden, ist die durch die Textplanung erzeugte Baumstruktur in eine logische Form überführt worden. Dann sind Verfahren zur Oberflächenrealisierung anwendbar, die eine in logischer Form vorliegende Satzsemantik in eine Sequenz von flektierten Wörtern überführt. Es gibt zwei prinzipielle Verfahren für diese Aufgabe: **strukturgesteuerte** und **lexemgesteuerte**. Wir betrachten sie der Reihe nach.

Strukturgesteuerte Oberflächenrealisierung

In strukturgesteuerten Verfahren erfolgt die Realisierung mittels einer deklarativen Grammatik, die im Idealfall unabhängig von der Interpretationsrichtung ist. Wenn die Grammatik allerdings im Stil parsing-orientierter Verarbeitungstechniken einfach top-down oder bottom-up eingesetzt wird, gibt es für die Generierung zwei fundamentale Probleme (vgl. Unterkapitel 3.5):

- Linksrekursion kann zu Endlosschleifen führen.
- Das Zusammenspiel zwischen Interpretation von Grammatikregeln und Zugriff auf lexikalische Information kann ohne ausgeklügelte Kontrolltechniken sehr ineffizient werden.

Endlosschleifen treten bei top-down Verarbeitung auf im Zusammenhang mit wechselseitiger Rekursion, oder bei bestimmten Konstruktionen, wie Abspaltung eines Verbpartikels aus einer VP. Eine solche Regel (Abspaltung eines Verbpartikels) ist linguistisch sinnvoll, sie könnte jedoch mehrmals hintereinander angewendet werden, was zu einer nicht intendierten Übergenerierung führt. Das grundlegende Problem besteht darin, dass die Wohlfundiertheit des Generierungsprozesses auf lexikalischer Information beruht, die für top-down Verfahren nicht zugänglich ist.

Im Gegensatz zu der top-down Verarbeitung haben bottom-up Verfahren kein Problem mit Endlosschleifen. Sie leiden jedoch bei der typischen Abarbeitung der rechten Seite der Grammatikregeln, von links nach rechts, an Ineffizienz, da diese Strategie an der bei der Analyse vorhandenen Information, der Eingabezeichenkette, orientiert ist, und nicht an der logischen Form, der Ausgangsrepräsentation bei der Generierung. Daraus ergeben sich eine Vielzahl von Indeterminismen, z. B. bei der Generierung einer Nominalphrase in Bezug auf Kasus und Numerus, da diese erst vom nachfolgenden Verb festgelegt werden. Deshalb wurden Verarbeitungsstrategien entwickelt, die sich an der semantischen Struktur orientieren, die sogenannten **kopfgesteuerten** Generierungstechniken (Shieber, Pereira, van Noord und Moore 1990). Sie basieren auf einer systematischen Gemeinsamkeit der logischen Form zwischen Mutterknoten und einem der Tochterknoten, dem **Pivot**, und setzen semantisch monotone Grammatiken voraus. Im Wesentlichen werden zunächst top-down Pivot Knoten expandiert, solange bis ein Zugriff auf lexikalische Information erfolgen kann. Dann werden sie, soweit erforderlich, bottom-up mit Mutterknoten verbunden. Schließlich werden mit der nun verfügbaren lexikalischen Information top-down alle nicht-pivot Knoten expandiert. Dieses Verfahren löst zwar das Effizienzproblem, trifft dafür aber starke Annahmen über die Grammatik.

Lexikalisch gesteuerte Oberflächenrealisierung

Im Gegensatz zu den strukturgesteuerten Verfahren ist die Eingabe für die lexikalisch gesteuerten keine voll strukturierte logische Form, sondern eine Sammlung lexikalischer Elemente, wobei die semantischen Relationen durch geeignet instanzierte Variablen ausgedrückt sind. Dies entspricht inhaltlich weitgehend

einer logischen Form, nur die Skopusbeziehungen sind wegabstrahiert. Verschiedene Algorithmen sind zur Erzeugung von Sätzen aus solchen Strukturen vorgeschlagen worden, mit unterschiedlichem Erfolg in Bezug auf Effizienz. Sie reichen von **shake-and-bake** Generierung (Whitelock 1992) über **Chart-Generierung** (Kay 1996), verfeinert durch ein **semilexikalisches** Verfahren (Carroll et al. 1999), und durch Berücksichtigung von Präferenzen (White 2004).

„Shake-and-bake“ Generierung kombiniert die lexikalischen Elemente der Eingabe in jeder möglichen Reihenfolge und testet die grammatische Wohlgeformtheit der jeweiligen Ergebnisse mit einem shift-reduce Parser. Da dieses Verfahren keinerlei Information über die Eingabesemantik ausnutzt, ist sein exponentielles Verhalten nicht verwunderlich.

Chart-Generierung adaptiert ein aus dem Parsing (siehe Unterkapitel 3.5) als effizient bewährtes Verfahren und vermeidet die wiederholte Berechnung von Kombinationen lexikalischer Elemente. Zunächst werden alle Lexikoneinträge bestimmt, die von der Eingabe subsumiert werden, und auf die Agenda gesetzt, von wo sie nacheinander in die Chart eingetragen werden. Dabei werden Interaktionen zwischen Einträgen der Chart berücksichtigt, das heißt, es werden Zusammensetzungen von Charteneinträgen gebildet, die einer Grammatikregel entsprechen, und sie werden in die Chart eingefügt. Ein erfolgreicher Abschluss des Verfahrens ist dann sichergestellt, wenn eine Zielstruktur, meist ein Satz, gebildet wurde, und dabei alle Elemente der Eingabe genau einmal verwendet wurden. Chart-Generierung erzielt signifikante Verbesserungen gegenüber shake-and-bake Generierung, hat aber Effizienzprobleme mit Kombinationen von intersektiven Modifikatoren. Auch die von Kay vorgeschlagene Verbesserung, dass nur maximale Strukturen mit anderen kombiniert werden, hilft nur teilweise, da dadurch das Bilden aller Teilmengen von Modifikatoren nicht vermieden werden kann.

Das Problem der Chart-Generierung mit intersektiven Modifikatoren führte zu der Verbesserung in dem semilexikalischen Verfahren. Dieses ist ein zweistufiges Verfahren, in dem zunächst Chart-Generierung ohne intersektive Modifikatoren durchgeführt wird und diese anschließend mit Adjunktion sukzessive eingefügt werden. Die Korrektheit dieses Vorgehens ergibt sich daraus, dass Adjunktion von intersektiven Modifikatoren höchstens zu einer Spezialisierung, aber nie zu einer Änderung syntaktischer Kategorien führt und somit durch eine solche Adjunktion niemals zusätzliche Grammatikregeln anwendbar werden. Bei der Adjunktion von Modifikatoren werden Bedingung über ihre Reihenfolge berücksichtigt und Wahlmöglichkeiten bezüglich der Reihenfolge werden kompakt durch Zeigerstrukturen ausgedrückt, so dass Teilstrukturen nicht mehrfach erzeugt werden müssen.

Das semilexikalische Verfahren wurde für umfangreiche Grammatiken des Englischen getestet und zeigte signifikante Laufzeitverbesserungen gegenüber reiner Chart-Generierung zwischen einem Faktor drei für normale Sätze bis zu Faktoren größer als zehn für Sätze mit untypischer Anzahl intersektiver Modifikatoren. Der Algorithmus ist sehr geeignet für die Verarbeitung lexikalisierter Grammatiken wie HPSG (siehe Unterkapitel 3.5) und macht nur geringe Annahmen über die Grammatik, die weit weniger restriktiv sind als die Forderung nach einem semantischen Kopf in strukturgesteuerten Verfahren.

Die von White eingeführte Verfeinerung besteht im wesentlichen in der Ausnutzung von empirisch motivierten Präferenzen zur Verbesserung der Effizienz. Dabei werden frühzeitig Kanten in der Chart eliminiert, deren assoziierte Phrasen jenen von alternative Kanten gemäß einem sprach- und eventuell domänenspezifischen n-gram Modell unterlegen sind. Mit diesem Verfahren kann auch die Entscheidung über Alternativen in der Lexikalisierung in die Oberflächengenerierung verlagert werden.

3.8.6 Linguistische Theorien zur Generierung

Einige vornehmlich funktional orientierte linguistische Theorien eignen sich besonders zur Generierung. Dazu gehören unter anderem Baumadjunktionsgrammatiken, die in Unterkapitel 2.2 beschrieben werden. Sie werden wegen ihrer Flexibilität für oberflächennahe Aufgaben in der Generierung vielfach eingesetzt. Zwei speziell für die Generierung geeignete Theorien, die **Meaning Text Theorie** und die **Systemisch Funktionale Grammatik**, werden in der Folge kurz vorgestellt.

Meaning Text Theorie

Die Meaning Text Theorie (MTT) ist ein extrem stratifiziertes, d. h. in Schichten eingeteiltes Sprachmodell, wobei die semantische, die tiefe syntaktische und die oberflächennahe syntaktische Repräsentationsebene in Sprachen wie Englisch die wichtigsten Strata (d. h. Schichten bzw. Ebenen) sind. Zentral für MTT ist die Rolle des Lexikons mit einer speziellen Eigenschaft, den **lexikalischen Funktionen** (Melcuk 1982). Das sind Verweise von einem Lexikoneintrag zu anderen Wörtern, die zu diesem Wort semantisch oder kollokational in Beziehung stehen.

Die theoretischen Konzepte von MTT sind in dem „Explanatory Combinatorial Dictionary“ (Melcuk und Polguere 1987) implementiert, dessen Einträge in semantische, syntaktische, und lexikalische Zonen untergliedert sind. Die semantische Zone eines Lexikoneintrags spezifiziert ein semantisches Netzwerk, das die mit diesem Eintrag assoziierte Bedeutung in Form von nächst einfacheren Wortbedeutungselementen ausdrückt. Auf der Basis dieser Definitionen können große semantische Netzwerke inkrementell reduziert werden, indem einzelne Knoten, die komplexe Bedeutungen repräsentieren, für Teilnetze substituiert werden, die die Bedeutung eines solchen Knotens mit einfacheren Bedeutungselementen repräsentieren. Die syntaktische Zone spezifiziert die syntaktische Kategorie eines Lexikoneintrags, seine syntaktischen Eigenschaften sowie die Art der Abbildung der semantischen Rollen des Eintrags auf tiefen und oberflächennahe syntaktische Repräsentationen. Die lexikalische Zone spezifiziert Bezüge zu semantisch verwandten Lexemen als Werte von etwa 60 lexikalischen Funktionen. Dazu gehören paradigmatische Funktionen, die Verbindungen zu verwandten Wörtern derselben syntaktischen Kategorie ausdrücken, wie etwa Synonyme, Generalisierungen und Konverse sowie syntagmatische Funktionen, die den Wert von Kollokationen für Wörter berechnen, die im selben Satz in einem spezifischen syntaktischen Bezug zu dem Lexikoneintrag stehen.

Die Generierung eines in Form eines semantischen Netzwerks spezifizierten Textes erfolgt dann in drei Phasen, wobei das Netzwerk natürlich ontologisch zu den Lexikoneinträgen passen muss:

1. Netzwerkreduktionen und -vereinfachungen
2. Übergang vom semantischen zum tiefen syntaktischen Stratum
3. Übergang vom tiefen syntaktischen zum oberflächennahen syntaktischen Stratum

Durch Netzwerkreduktionen können umfangreiche Definitionen durch den damit assoziierten Term ausgedrückt werden, wie etwa *more than one program of a type such that someone compiles something with these programs* durch *compilers*. Damit ist die Behandlung einiger Fälle von Lexikalisierung im eigentlichen Sinne (gemäß Cahill) möglich, was eine besondere Fähigkeit von MTT im Vergleich zu anderen linguistischen Theorien darstellt. Bei Anwendbarkeit mehrerer Äquivalenzdefinitionen können unter Umständen ganz unterschiedliche Aufteilungen des Informationsgehalts erfolgen, so dass *Fred überquerte den Fluss schwimmend* und *Fred schwamm quer über den Fluss* (siehe obigen Abschnitt über Lexikalisierung) aus dem gleichen zugrunde liegenden Netzwerk erzeugt werden können.

In der zweiten Phase bestehen oft Alternativen für das verbale Ausdrücken einzelner Knoten. So kann etwa ein mit *name* bezeichneter Knoten entweder durch *Name* oder *heißen* ausgedrückt werden. Die Auswahl zwischen diesen Alternativen und Zusammensetzungen wird gemäß der Thema/Rhema-Struktur getroffen.

In der dritten Phase schließlich bestehen durch auf lexikalischen Funktionen definierte Regeln weitere Möglichkeiten zur Paraphrasierung, etwa durch Bildung von Funktionsverbkonstruktionen. Beispielsweise liefert eine bestimmte lexikalische Funktion zu *entscheiden* die nominalisierte Form *Entscheidung* und eine andere lexikalische Funktion das dazu passende Funktionsverb *treffen*. Dabei sorgt die Definiertheit oder Nicht-Definiertheit einer lexikalischen Funktion für einzelne Lexeme für die Vermeidung der Bildung inkorrektener Phrasen.

Beim Einsatz von Meaning Text Theorie zur Generierung muss die Eingabe zunächst in satzgrößenartige semantische Netzwerke zerlegt werden, die jeweils mit Markierungen zur Thema/Rhema-Struktur angereichert sind. Meaning Text Theorie ist ein ehrgeiziger und wohlfundierter Ansatz zur Repräsentation lexikalischen Wissens und geht dabei über die Fähigkeiten anderer Theorien hinaus, z. B. durch die expliziten Netzwerkdefinitionen. Die Anwendung der Theorie setzt aber die Ausarbeitung solcher umfangreicher Lexika voraus, was einen enormen Entwicklungsaufwand erfordert. Dariüber hinaus sind die auf diesen Definitionen arbeitenden Transformationsoperationen zur Exploration der alternativen Optionen mit hohem Rechenaufwand verbunden. Meaning Text Theorie ist somit ein typisches Beispiel einer Theorie, die über eine erhebliche Ausdrucksmächtigkeit und linguistische Abdeckung verfügt, diese Vorteile allerdings mit hohem Effizienzverlust erkauft.

Systemisch Funktionale Grammatik

Die Systemisch Funktionale Grammatik ist eine Theorie natürlicher Sprache, für die der Begriff der Funktion zentral ist (Halliday 1994). Sie ist ähnlich wie die Meaning Text Theorie ein stratifiziertes Modell, und sie enthält neben den üblichen linguistischen Repräsentationsebenen auch außer-linguistische Aspekte, wie etwa soziale Kontexte. Neben der Organisation in Form der Strata, die auf Abstraktion basieren, gibt es eine inhaltliche Organisation in Form von **Metafunktionen**, die auf Diversifikation basieren, also unterschiedliche Perspektiven liefern. Es gibt drei Metafunktionen: **ideationale**, **interpersonale** und **textuelle**. Die ideationale Sicht bezieht sich auf inhaltliche, semantische Aspekte, also den reinen Informationsgehalt einer Äußerung. Die interpersonale Sicht betrifft den sozialen Bezug zum Hörer, besagt also wie der Informationsgehalt gesehen wird (hypothetisch, befragt, etc.). Die textuelle Sicht schließlich behandelt die kontextuelle Einbettung der Information, ist somit z. B. für die Thema/Rhema-Struktur zuständig. Die Erscheinungsform einer generierten Äußerung kann durch Spezifikationen beeinflusst werden, die jeweils unterschiedlichen Metafunktionen zuzuordnen sind.

Elemente des operationalen Teils einer Systemisch-funktionalen Grammatik sind **Systemnetzwerke**. Dies sind Vererbungsnetzwerke, die eine Menge zusammenhängender Auswahlmöglichkeiten darstellen. Jedes System in diesem Netzwerk verfügt über eine Eingangsbedingung und über zwei oder mehr paradigmatische Merkmale, die grammatische Alternativen darstellen, von denen bei Ansteuern eines Systems eine auszuwählen ist. So besteht etwa bei dem System *Satzmodus* die Wahl zwischen den Merkmalen *imperativ* und *indikativ*; letzteres entspricht wiederum der Eingangsbedingung für das System *Indikativ Typ* mit den Wahlmöglichkeiten *deklarativ* und *interrogativ*. Beim Traversieren des Netzwerks werden die den gewählten Varianten entsprechenden paradigmatischen Merkmale gesammelt. Das Traversieren des Netzwerks erfolgt durch iterative Ausführung eines der Systeme, deren Eingangsbedingung erfüllt ist (oder mehrerer parallel). Der Effekt dieser Ausführung kann zur Erfüllung der Eingangsbedingung weiterer Systeme beitragen.

Die Grammatik selbst ist mit der syntagmatischen und mit der semantischen Ebene verbunden. Die syntagmatische Organisation basiert auf mit paradigmatischen Merkmalen verbundenen Realisierungsfestlegungen, die etwa das Vorhandensein eines Merkmals oder Reihenfolgebedingungen ausdrücken. So führt etwa das paradigmatische Merkmal *wh-* zur Einführung einer wh-Funktion und legt fest, dass in der Ausgabe die wh-Konstituente vor dem finiten Verb kommt. Die Verbindung zum semantischen Stratum erfolgt mittels sogenannter **Chooser**, die mit Systemen verbunden sind, um grammatische Entscheidungen semantisch zu motivieren. Die Chooser sind intern als Entscheidungsbäume organisiert, bei deren Verzweigungen eine Reihe unterschiedlicher Faktoren zur Auswertung herangezogen werden können: textuelle Relationen, Domänenwissen, ein Benutzерmodell und kommunikative Ziele.

Zwei der führenden auf Systemisch-funktionaler Grammatik basierende Generierungssysteme sind GENESYS (Fawcett und Tucker 1990) und Penman (Mann

und Matthiessen 1985). GENESYS ist ein sehr reichhaltiges Satzgenerierungssystem, das von Benutzerspezifikationen gesteuert wird. Unter anderem kann es semantisch motivierte Spezifikationen für Intonation erzeugen. Penman ist in der Lage Sätze aus semantischer Eingabe zu erzeugen. Einige Weiterentwicklungen arbeiten an multilingualer Generierung mit mehreren, teilweise überlappenden Grammatiken für die Zielsprachen.

3.8.7 Ausblick

Ein genereller Trend in der Textgenerierung besteht in der verstärkten Orientierung von Systemen an menschlichen Ausdruckspräferenzen und existierenden Korpora. Dies führte zum Einsatz statistischer Methoden, sei es um symbolische Verfahren mit feineren, teils anwendungsspezifischen Kriterien zu steuern, sei es in Form von stark oberflächenorientierten Verfahren, die Texte in weitgehend direkter Weise erzeugen, ohne die zahlreichen Zwischenrepräsentationen der hier vorgestellten symbolischen Verfahren.

Während symbolische Verfahren Texte im wesentlichen durch eine Folge deterministischer Entscheidungen sukzessive immer präzisere Beschreibungen generieren, arbeiten statistische Verfahren mit einer systematischen Auswahl, gesteuert durch n-gram Modelle, aus einer in der Regel sehr großen Menge dynamisch erzeugter Alternativen. Hauptprobleme bei diesem Ansatz sind unter anderem die Tendenz zur Bevorzugung kürzerer Varianten, die Kontrolle über Fernabhängigkeiten, sowie generell der erhebliche Rechenaufwand bei größerem Umfang an Wortschatz und Satzstrukturen. Speziell an der Verbesserung der Effizienz wird intensiv gearbeitet, vor allem durch kompakte Repräsentationen aller Alternativen. Eine wesentliche Attraktivität besteht wie bei der statistisch-basierten maschinellen Übersetzung in der Einfachheit und universellen Anwendbarkeit der Methodik, vorausgesetzt es sind genügend große Korpora mit Bezug zu den zu verbalisierenden Daten vorhanden.

Eine ganz aktuelle Entwicklung sind Wettbewerbe für die Erzeugung von Texten gemäß einem „golden Standard“, wie dies in anderen Bereichen der Computerlinguistik seit längerem üblich ist. Die Bewerbe beziehen sich auf die Erzeugung von Referenzausdrücken, die mit in kontrollierten Experimenten beobachteten Ausdrücke verglichen werden. Zunächst beschränkten sich die Bewerbe auf die Attributauswahl, später wurden auch Oberflächenform und Kontext, unter anderem wiederholte Referenzen auf ein Objekt, mit einbezogen.

Für viele Teilaufgaben des Generierungsprozesses gibt es ein oder mehrere etablierte Verfahren zur Lösung der jeweiligen Teilaufgabe. Typischerweise sind diese Verfahren an einer lokal funktionalen Sicht orientiert, ohne Perspektive auf den gesamten Generierungsprozeß. Des Weiteren sind die linguistischen, rhetorischen und auch psychologischen Kriterien zur Wahl zwischen alternativen Ausdrucksmöglichkeiten an verschiedenen Stellen der Verarbeitung oft zu allgemein um in einer konkreten Situation eine angemessene Entscheidung zu begründen. Sie müssen ergänzt werden durch spezifische, operationale Kriterien, gemäß der gewünschten Funktionalität des jeweils betrachteten Systems. Bekannte Verfahren leisten vorrangig die kohärente Organisation von satzartigen Propositionen,

sowie verschiedene Teilbereiche der Satzplanung, wie etwa Erzeugung referentieller Beschreibungen, das heißt identifizierender Referenzausdrücke. Am weitesten fortgeschritten sind Oberflächenrealisierungsverfahren, manche mit recht umfangreicher Abdeckung, jedoch meist nur für das Englische. Wenig methodische Unterstützung erhält man als Systemdesigner neben den ausreichend spezifischen Auswahlkriterien bei der Modellierung von Inferenzen und bei der Behandlung von Annahmen über den Hörer, speziell dann wenn diese Annahmen mit Unsicherheit behaftet sind und Entscheidungen über den zu generierenden Text dadurch zum Teil strategisch (vorsichtig oder spekulativ) motiviert werden.

Insgesamt erweist sich die Textgenerierung, speziell in den verschiedenen Planungsaspekten, als ziemlich schwierig, da man relativ viele zu Teil sehr unterschiedliche Verfahren kennen muss, um die Problematik des Gebiets zu verstehen und Anforderungen für die Funktionalität eines zu entwickelnden Systems einzuschätzen zu können.

3.8.8 Literaturhinweise

Eine ausführliche Diskussion zu Architekturen von Generierungssystemen findet man in De Smedt et al. (1995). Oberflächenrealisierungskomponenten mit umfangreicher Abdeckung sind SURGE (Elhadad und Robin 1999) und KPML (Bateman 1997). Einige der bekanntesten Generierungssysteme sind FOG (Goldberg et al. 1994), das Wettervorhersagen in Englisch und Französisch produziert, IDAS (Reiter et al. 1995), das online technische Dokumentationen aus einer Wissensbasis erzeugt, und MAGIC (McKeown et al. 1997), das multimediale Kurzbeschreibungen zur Gesundheitsvorsorge generiert. Ein neueres System ist GEA (Carenini und Moore 2006), das evaluierende Argumente generiert und eines der wenigen Generierungsprogramme ist, für das eine effektive Anwendbarkeit statistisch signifikant nachgewiesen wurde.

3.9 Programmiersprachen in der Computerlinguistik

André Hagenbruch

Betrachtet man das Spektrum zwischen computerlinguistischer Forschung und industrieller Sprachtechnologie, eröffnet sich ein weites Feld unterschiedlichster Problemstellungen. Begibt man sich an die Modellierung einer konkreten Aufgabe, steht man zunächst vor der Entscheidung, in welchem Umfang diese implementiert werden kann: Reicht ein einfacher Prototyp zur Demonstration eines Phänomens, soll ein in sich geschlossenes, kommerziell verwertbares System konstruiert werden oder ist sowohl die Software als auch der Quellcode für eigene Entwicklungen oder Projekte anderer wiederverwertbar? Daraus ergeben sich wiederum neben Fragen der adäquaten linguistischen Modellierung auch solche, nach der optimalen softwaretechnischen Realisierung. Gibt es ein Programmierparadigma oder eine Programmiersprache, die anhand ihrer Architektur zur Modellierung des gegebenen Problems gegenüber anderen Kandidaten besonders geeignet erscheint? Existieren bereits Software-Bibliotheken, Frameworks oder gar von Programmiersprachen unabhängige Entwurfsmuster, die bei der Implementierung nachgenutzt werden können? Liegt der Fokus der Entwicklung auf möglichst effizienter Verarbeitung, auf einfacher Implementierbarkeit, Flexibilität oder Skalierbarkeit? Müssen Faktoren wie Multilingualität (und möglicherweise damit einhergehend: unterschiedliche Schreibsysteme) oder Multimodalität berücksichtigt werden? Zu all diesen Fragen existieren in der Literatur kaum Orientierungspunkte (vgl. Leidner 2003): Gaben Gazdar und Mellish 1989 Antworten auf diese Fragen in Form dreier Monographien mit den Titeln *Natural Language Processing in Lisp/Prolog/Pop11*, findet sich in aktuellen Standardwerken wie Jurafsky und Martin (2009) oder Manning und Schütze (2003) nur Pseudocode zu ausgewählten Beispielen. Das vorliegende Kapitel versucht gerade Einsteigern einen Einblick in die Vor- und Nachteile einiger aktueller Programmiersprachen zu geben und anhand eines historisch strukturierten Überblicks Entwicklungen in diesem Bereich nachzuzeichnen und Trends aufzuzeigen.

3.9.1 Die Anfänge: Hochsprachen und symbolische Sprachverarbeitung

Die Anfänge computerlinguistischer Implementierungen lassen sich in den 50er- und 60er-Jahren verorten: In Projekten zur maschinellen Übersetzung setzte man Programmiersprachen wie COMIT, Snobol oder Trac ein. An der Schwelle von der Programmierung in reinem Maschinencode zu abstrakten Hochsprachen waren dies die ersten Sprachen, mit denen Zeichenketten verarbeitet wurden. Diese waren ausreichend, um die damals noch naiven Vorstellungen einer „mechanischen“ Übersetzung anhand von Wörterbüchern und einfachen morphologischen und syntaktischen Ersetzungen realisieren zu können. Ambitionierter waren Experimente wie **SHRDLU** oder **ELIZA**, die im Rahmen der KI-Forschung für Furore sorgten (vgl. Carstensen 2009b).

Das zwischen 1968 und 1970 von Terry Winograd entwickelte SHRDLU erlaubte es seinem Benutzer, Objekte in einer einfachen Klötzchenwelt mittels natürlichsprachlicher Anweisungen zu bewegen oder Aussagen über Relationen der Objekte untereinander abzufragen. Dazu bediente sich das Programm eines stark auf dieses Szenario eingeschränkten Lexikons und einer einfachen Grammatik; zur Bewältigung der Bauanleitungen besaß es ein eigenes System zur Problemlösung. Darüber hinaus war es mit SHRDLU möglich, komplexe Objekte aus den vorhandenen Grundbausteinen zu erzeugen. Vermittels eines Speichers, in dem die Historie der ausgeführten Aktionen protokolliert wurde, war es nicht nur möglich, den Verlauf der Aktionen nachzuvollziehen, sondern auch Weltwissen, z. B. darüber, welche Objekte miteinander kombiniert werden konnten, zu simulieren.

Implementiert wurde diese Anwendung in einem Dialekt der in ihrer ursprünglichen Form 1958 von John McCarthy spezifizierten Sprache **Lisp**. Wie ihr ausgeschriebener Name *List Processing Language* bereits andeutet, besteht der Quellcode eines Lisp-Programms aus in runden Klammern gekleideten Listen, sogenannten S-Ausdrücken. Notiert werden diese Ausdrücke in Präfix-Schreibweise, d.h. Operatoren, Funktions- und Makroaufrufe stehen vor ihren Argumenten: (+ 3 4). Listen selbst bestehen aus sogenannten *cons cells*, deren erstes Element als **car** und deren zweites Element als **cdr** bezeichnet wird:

```
(car (cons 1 2)) ; AUSGABE: 1
(cdr (cons 1 2)) ; AUSGABE: 2
```

Da der **car**-Teil einer solchen Zelle ein beliebiges Objekt enthalten darf, lassen sich auf diese Weise durch Verkettung von *cons cells* neben einfach verlinkten Listen auch Datenstrukturen wie Bäume, Mengen, assoziative Listen (engl. *associative lists*, Alists) und Eigenschaftenlisten (engl. *property lists*, Plists) realisieren. Zur Illustration der Struktur von Lisp-Programmen mag ein kurzes Beispiel aus Gazdar und Mellish (1989a, S. 122ff) dienen, das anhand einer kleinen Grammatik Parsebäume englischer Sätze generiert:

```
1 (defvar rules)
2 (setf rules
3   '(( (S) (NP) (VP))
4     ((VP) (V))
5     ((VP) (V) (NP))
6     ((V) died)
7     ((V) employed)
8     ((NP) nurses)
9     ((NP) patients)
10    ((NP) Medicenter)
11    ((NP) Dr Chan)))
13 (defun generate (description)
14   (if (atom description)
15     description
16     (let ((rs (matching_rules description)))
17       (if (null rs)
```

```

18      (error "Cannot_generate_from_description~S"
19          description)
20      (cons
21          (category description)
22          (generate_all (cdr (oneof rs))))))
23
24 (defun generate_all (body)
25     (if (null body)
26         '()
27         (cons
28             (generate (car body))
29             (generate_all (cdr body))))))
30
31 (defun category (description)
32     (car description))
33
34 (defun matching_rules (description)
35     (let ((results ()))
36         (dolist (rule rules results)
37             (if (unify description (car rule))
38                 (setq results (cons rule results))))))
39
40 (defun unify (x y)
41     (equal x y))
42
43 (defun oneof (list)
44     (nth (random (length list)) list))

```

Zunächst werden die Grammatikregeln und ein kleines Lexikon als Listeneinträge definiert, die in der globalen Variablen `rules` gespeichert werden; zur besseren Unterscheidbarkeit von Kategorien und Wörtern werden erstere in eigene Listen gesetzt. Die Funktion `generate` nimmt als Argument die zu generierende Phrase und wird solange rekursiv aufgerufen, bis das `cdr` der Liste ein Wort ist. Dazu gibt die Funktion `matching_rules` aus der Liste der Phrasen eine Kandidatenliste zurück, aus der die Funktion `oneof` eine beliebige auswählt. Rückgabewert der Funktion `generate` ist ein einzelner Parsebaum, während `generate_all` eine Liste aller Parsebäume zurückgibt.

Ein weiteres wichtiges Merkmal von Lisp ist die Zugehörigkeit zu den **funktionalen Programmiersprachen**: alle verwendeten Funktionen operieren bei der Berechnung ihrer Ergebnisse allein auf den Werten der ihnen übergebenen Argumente, sodass keine Seiteneffekte durch vorherige Verarbeitungsschritte auftreten können. Ein weiteres dieses Paradigma kennzeichnendes Merkmal ist die Verwendung von Funktionen höherer Ordnung, d.h. Funktionen, die ihrerseits Funktionen als Argumente übernehmen oder Funktionen als Werte zurückgeben; dadurch lässt sich sehr eleganter und konziser Quellcode erstellen. All diese Eigenschaften machten Lisp in den 80er-Jahren zu einer der beiden Sprachen der Wahl, wenn es um die Implementierung von Parsern oder Anwendungen zur Generierung natürlicher Sprache, der Modellierung von Grammatiken oder se-

mantischer und pragmatischer Phänomene, wie sie z. B. in Gazdar und Mellish (1989a) beschrieben werden. Parallel zu diesem Buch erschien mit Gazdar und Mellish (1989b) auch eine Darstellung dieser Domäne in Prolog. Zusammen mit einer dritten Ausgabe, welche die Programmierung in Pop11 beschrieb, stellte dieser Titel das erste Standardwerk zur Einführung in die computerlinguistische Programmierung dar.

Prolog (*Programmation en Logique*) wurde Anfang der 70er Jahre von Alain Colmerauer entwickelt, und ähnlich wie Lisp gab es bis zu ihrer Standardisierung 1995 als ISO/IEC 13211-1 verschiedene Dialekte. Strukturell besteht ein Prolog-Programm aus geordneten Listen prädikatenlogischer Terme (s. Unterkapitel 2.1), sogenannten **Klauseln**. Alle Klauseln zusammengenommen bilden eine Wissensbasis, auf der man Anfragen ausführen kann, deren Antwort entweder **yes** oder **no** ist. Darüber hinaus werden Implikationen durch mit **:-** verbundenen Klauseln notiert. Dies nennt man dann eine **Regel**. Dazu ein Beispiel aus Blackburn et al. (2006):

```
woman(mia).

listensToMusic(mia).
playsAirGuitar(mia) :-  
    listensToMusic(mia).
```

Fragt man nun z. B. mit `?- woman(mia)`, ob Mia eine Frau ist, gibt der Prolog-Interpreter **yes** zurück. Ebenso bei der Query `?- playsAirGuitar(mia)`. Obwohl nicht explizit in der Wissensbasis notiert, folgt der Fakt, dass Mia Luftgitarre spielt durch Anwendung des modus ponens (s. Unterkapitel 2.1) aus obenstehender Regel. `?- playsAirGuitar(butch)`. hingegen wird mit **no** beantwortet, da über Butch keine Informationen in der Wissensbasis vorhanden sind. Weitere wichtige Merkmale von Prolog sind einerseits das Vorhandensein der Unifikations-Operation (s. Unterkapitel 2.3) als fester Bestandteil der Sprache, andererseits die Möglichkeit, die **Definite Clause Grammar** (DCG) in den meisten Prolog-Systemen über Präprozessordirektiven einzubinden. Durch letzteres lassen sich auf einfache Art kontextfreie Grammatiken direkt in Prolog definieren.

Obwohl beide Sprachen in den letzten Jahren an Bedeutung für die computerlinguistische Programmierung eingebüßt haben (vgl. z. B. <http://xkcd.com/224/>), finden sich immer noch aktuelle Anwendungen wie z. B. der in Lisp geschriebene *RegexCoach* (<http://www.weitz.de/regex-coach/>) oder die Möglichkeit, vermittels Prolog Wissensbasen im Bereich des *Semantic Web* (siehe auch Unterkapitel 4.7) abzufragen. Neben den oben erwähnten Titeln empfiehlt sich Seibel (2004) als Einführung in *Common Lisp* und Blackburn et al. (2006) zur Einführung in Prolog.

3.9.2 C/C++

Ebenfalls aus den frühen 70er-Jahren stammt die von Dennis Ritchie entwickelte Programmiersprache **C** (Brian Kernighan, der ebenfalls mit ihr assoziiert wird,

war an der Spezifikation nicht beteiligt); ihre Entwicklung ist eng an jene des Betriebssystems *Unix* gekoppelt (zur Möglichkeit, computerlinguistische Analysen rein mit GNU/Unix-Werkzeugen zu betreiben, siehe Brew und Moens 2000). Mit seiner Verbreitung nahm auch die Verwendung von C zu, sodass binnen weniger Jahre unterschiedlichste Varianten von C entstanden. Dies ist dem Umstand geschuldet, dass man in dieser Sprache einerseits sehr systemnah programmieren kann, es andererseits zunächst kaum Bibliotheken gab, die beispielsweise Ein-/Ausgabefunktionalitäten auf einheitliche Weise realisiert hätten. Erst in den 80er-Jahren kam ein Standardisierungsprozess in Gang, der in drei Etappen 1990, 1995 und zuletzt 1999 eine ISO-Spezifikation der eigentlichen Sprache und der sogenannten *ISO Standard C Library* hervorbrachte. Da aber auch diese sehr minimalistisch gehalten ist (was der Portierbarkeit auf verschiedene Plattformen zugute kommt), sind C-Programme häufig umständlicher zu realisieren als analoge Implementierungen in anderen Sprachen, dafür aber oftmals auch performanter. Dies ergibt sich aus der Möglichkeit, Speicherbereiche selbst verwälten und durch Variablen auf bestimmte Speicheradressen verweisen zu können (diese Variablen heißen *Zeiger*, engl. *Pointer*). Diese Freiheit bringt aber auch für den Programmierer die Pflicht mit sich, eigenständig nicht mehr verwendete Speicherbereiche freizugeben bzw. bereits verwendeten Speicher nur dann zu überschreiben, wenn dieser nicht mehr gebraucht wird. Da dies eine häufige Fehlerquelle darstellt, besitzen andere Sprachen eine sogenannte **Garbage Collection**, die nicht mehr referenzierten Speicher automatisch freigibt.

Strukturell betrachtet besteht C-Quellcode zunächst aus Präprozessordirektiven, die Funktionen aus der Standardbibliothek oder anderen Bibliotheken zur Verwendung in den eigenen Code einbinden. Das eigentliche Programm steht in einer Funktion namens `main`, in der einfache Anweisungen oder vordefinierte Funktionen ausgeführt werden. Abgeschlossen wird `main` mit einer `return`-Anweisung, die den Ausführungsstatus wiedergibt. Um ein C-Programm auszuführen, muss es zunächst mit einem **Compiler** in eine Objektdatei übersetzt werden, die dann wiederum durch einen **Linker** gebunden und in eine ausführbare Datei überführt werden kann.

Eine Sprache, die direkt aus C hervorgegangen ist, und meist im gleichen Atemzug mit ihr genannt wird, ist **C++**. Diese wurde in Erweiterung des Sprachumfangs von C insbesondere in Richtung auf das Paradigma der **objektorientierten Programmierung** hin in den 80er-Jahren von Bjarne Stroustrup entwickelt. Als ISO/IEC 14882 ist auch sie standardisiert, wobei es 1998 und 2003 Spezifikationen gab und eine neue für 2009 erwartet wird. Neben der eigentlichen *Standard C++ Library* gibt es noch die *Standard Template Library* und die *Boost-Bibliothek*, die für einen höheren Abstraktionsgrad bei der Programmierung sorgen, als dies bei C der Fall ist.

Betrachten wir als Beispiel ein C++-Programm, das Wörter aus einer Datei ausliest und diese zählt (Stephens et al. 2006, S. 183ff):

```

1 #include <iostream>
2 #include <fstream>
3 #include <map>
4 #include <string>
```

```

6 typedef std::map<std::string , int> StrIntMap;

8 void countWords ( std::istream& in , StrIntMap& words) {
9     std::string s;
10    while (in >> s) {
11        ++words[ s];
12    }
13 }

15 int main (int argc , char** argv) {
16     if (argc < 2)
17         return (EXIT_FAILURE);

19     std::ifstream in (argv [1]);

21     if (!in)
22         exit (EXIT_FAILURE);

24     StrIntMap w;
25     countWords (in , w);

27     for (StrIntMap::iterator p = w.begin();
28          p != w.end (); ++p) {
29         std::cout << p->first << "=>" 
30                     << p->second << "\n";
31     }
32 }
```

In den Zeilen 1–4 werden zunächst einige Bibliotheken eingebunden, darunter die Datenstruktur `map`, die einen Container aus Schlüssel-/Wertpaaren – in diesem Fall Zeichenketten und ganze Zahlen (Zeile 6) – realisiert (in anderen Programmiersprachen wird eine solche Datenstruktur auch als assoziative Liste, *Hash* oder *Dictionary* bezeichnet). Standardmäßig sind diese Paare gemäß ihres Schlüssels sortiert; wünscht man beispielsweise für ein Histogramm eine Sortierung nach den Werten, muss dafür eine eigene Sortierfunktion implementiert werden.

In der Funktion `countWords` werden durch Leerzeichen voneinander getrennte Wörter aus dem im Funktionsaufruf (Zeile 25) übergebenen Datei-Datenstrom gelesen; in der `map` `words` werden die so segmentierten Wörter als Schlüssel angelegt und der Zähler für ihre Häufigkeit durch den sogenannten *Autoinkrement*-Operator `++` erhöht, wenn das jeweilige Wort erneut gefunden wird. In der in Zeile 27 beginnenden `for`-Schleife wird über die Einträge der `map` iteriert, wobei die Attribute `first` und `second` aus dem Template `pair` stammen. Hieran sieht man, wie die Struktur aus Schlüssel-/Wertpaaren intern realisiert wird.

C und C++ gehören nach wie vor zu den am weitesten verbreiteten Programmiersprachen und bilden die Grundlage nicht nur für eine Vielzahl verschiedenster Anwendungen, sondern auch für Frameworks und sogar andere Programmiersprachen. In der Computerlinguistik eignen sie sich durch die Mög-

lichkeit der hardwarenahen Programmierung vor allem für Aufgaben der Verarbeitung multimodaler Daten, oder zum Einsatz in performanzkritischen Anwendungen. Eine allgemeine Einführung in C++ bietet Kirch-Prinz und Prinz (2007), während Stephens et al. (2006) eine Sammlung verschiedenster Beispiel-Anwendungen darstellt.

3.9.3 Programmierarchitekturen: Java und .Net

Die immer weitere Verbreitung preiswerter und dennoch leistungsfähiger Computer seit den frühen 90er-Jahren hat das Verhältnis der durch eine Programmiersprache zu erzielenden Performanz zur Produktivität und Erlernbarkeit einer Sprache zugunsten dieser letzteren Faktoren verändert. In diese Lücke stießen einerseits dynamische Sprachen (die im nächsten Abschnitt besprochen werden), andererseits Frameworks wie Suns Java und Microsofts .Net.

Java wird seit 1991 unter der Federführung von James Gosling entwickelt. Bei Java handelt es sich neben der eigentlichen Programmiersprache um eine Laufzeitumgebung, der (in C und C++ geschriebenen) *Java Virtual Machine* (JVM), die für eine Reihe unterschiedlichster Betriebssysteme verfügbar ist. Im Entwicklungszyklus wird Quellcode zunächst in Bytecode übersetzt, der dann in einer beliebigen JVM ausgeführt werden kann. Lange Zeit beschränkte man sich dabei auf Java als Sprache, doch entwickelt sich seit einigen Jahren ein Trend, dynamische mit statisch typisierten Sprachen zu kombinieren: Während es mit dem *Simplified Wrapper and Interface Generator* (SWIG) (<http://www.swig.org>) möglich ist, andere Sprache wie z. B. Common Lisp, Java oder auch Skriptsprachen wie Perl oder Python in C/C++-Programme einzubinden, lassen sich Sprachen wie das eigens für die JVM entwickelte Groovy oder auf Java portierte Varianten eigener Sprachen wie z. B. Jython, JRuby oder Clojure (eine Lisp-Variante) direkt in der virtuellen Maschine ausführen. Daraus ergibt sich, dass man einerseits aus diesen Sprachen heraus Java-Klassen nutzen kann, andererseits Programmteile, die in einer dieser Sprachen implementiert sind, in Java-Programme einbetten kann. Darüber hinaus ermöglicht das *Java Native Interface* (JNI) die Verwendung plattformspezifischer in C/C++ geschriebener Bibliotheken (z. B. von *dlls* unter Windows).

Diesen Ansatz einer weitestgehend sprachagnostischen Programmierarchitektur verfolgt Microsoft mit seiner **.Net**-Plattform. Diese basiert auf der als ECMA-335 standardisierten *Common Language Infrastructure* (CLI), welche die Entwicklung mit unterschiedlichsten Programmiersprachen unter verschiedenen Betriebssystemen unterstützen soll. Damit diese Programme auch in solch heterogenen Umgebungen ausgeführt werden können, benötigen sie eine spezielle Laufzeitumgebung, die *Common Language Runtime* (CLR). Eine solche stellt Microsoft (neben entsprechenden Programmzbibliotheken) für seine Windows-Betriebssysteme zur Verfügung. Dass ein solcher Ansatz tatsächlich greift, beweist das mittlerweile von Novell betriebene *Mono*-Projekt, das die CLI und die CLR (einschließlich einiger .Net-eigenen Bibliotheken) unter Unix-artigen Betriebssystemen zur Verfügung stellt. Auch hier finden sich neben der eigens für .Net/Mono entwickelten Programmiersprache **C#** (als ECMA-334 standar-

disiert) in der Liste der verfügbaren Sprachen beispielsweise IronPython, IronRuby, Perl#, oder J# (einer Java-Variante).

Das oben beschriebene Beispiel zum Wörterzählen sieht in Java so aus:

```

1 import java.io.*;
2 import java.util.Map;
3 import java.util.TreeMap;

5 public class WordCount {
6     private static Map countWords (BufferedReader in) throws
7         IOException {
8         String [] words = null;
9         Map<String, Integer> counts = new TreeMap<String, Integer>
10            ();
11         String line;
12         while ((line = in.readLine ()) != null) {
13             words = line.split (" ");
14             for (String word : words) {
15                 Integer count = counts.get (word);
16                 count = (count == null) ? 1 : count + 1;
17                 counts.put (word, count);
18             }
19         }
20         return counts;
21     }
22
23     public static void main (String [] args) {
24         Reader fh = null;
25         Map<String, Integer> frequency = new TreeMap<String,
26             Integer> ();
27         try {
28             fh = new FileReader (args [0]);
29             BufferedReader in = new BufferedReader (fh);
30             frequency = countWords (in);
31             fh.close ();
32         } catch (IOException e) {
33             System.err.println (e);
34         }
35         for (Map.Entry<String, Integer> entry : frequency.entrySet
36            ()) {
37             System.out.printf ("%s => %d\n", entry.getKey (), entry.
38                 getValue ());
39         }
40     }
41 }
```

Wie schon das C++-Programm besteht auch dieses Beispiel aus einer `countWords`- und einer `main`-Methode. Und genauso lesen wir auch hier an Leerzeichen segmentierte Wörter in einen Hash-Container aus Zeichenketten und ganzen Zahlen ein. Da der `TreeMap`-Container allerdings nicht automatisch beim ersten

Vorkommen eines Worts mit 1 belegt wird, weicht unsere Strategie hier vom vorherigen Beispiel ab: Zunächst holen wir uns den unter dem Schlüssel `word` in der `TreeMap` `counts` gespeicherten Wert (Zeile 13). Ist dieser noch nicht vorhanden, steht dort an seiner statt die Null-Referenz `null`. Dies prüfen wir in Zeile 14 mithilfe des ternären Operators: Ist der Wert `null`, wird `count` 1 zugewiesen, ansonsten wird `count` um 1 erhöht. Ebenso wie die `map` in C++ ist auch Javas `TreeMap` nach den Schlüsseln sortiert. Um aber in dieser Struktur auf die Werte zugreifen zu können, benötigt man nicht das Wissen über das implizit zugrunde liegende Template `pairs`, sondern greift vermittels der Methoden `getKey` bzw. `getValue` auf die Attribute und Werte zu.

Beschränkte sich die Fehlerbehandlung im C++-Beispiel mit `EXIT_FAILURE` auf ein Makro, das einen Fehler zur Laufzeit anzeigt, wird in der `countWords`-Methode bereits angezeigt, dass sie einen Fehler der Klasse `IOException` (die in der ersten Zeile durch den Wildcard-Import eingebunden wird) aufwirft. Im Hauptprogramm wird in den Zeilen 24 bis 29 versucht, aus einer Datei zu lesen und die Wörter zu zählen. Schlägt dies fehl, wird der `IOException`-Fehler in Zeile 29 ff. dadurch behandelt, dass er ausgegeben wird.

Die große Stärke von Java besteht in seiner breiten Nutzerbasis; dementsprechend findet sich auch im Bereich der Computerlinguistik eine Vielzahl von Anwendungen. Darüber hinaus existieren aber auch komplett Frameworks, von denen hier zwei vorgestellt werden sollen. Das ursprünglich von IBM seit 2004 und mittlerweile von der *Apache Software Foundation* entwickelte *Unstructured Information Management Applications*-Framework (UIMA; <http://incubator.apache.org/uima/>) versteht sich als Architektur zur Erstellung multimodaler Analysewerkzeuge, die in Suchtechnologien integriert werden können. Neben dem eigentlichen Java-Framework existiert auch ein C++-Framework, über das auch Annotatoren in Perl, Python und Tcl implementiert werden können. Zu den Werkzeugen gehören Komponenten wie z. B. Tokenizer, Parser, Textzusammenfassung und Kategorisierung, aber auch eine semantische Suchmaschine. Neben einer ausführlichen Dokumentation gibt es eine Benutzer- und Entwickler-Mailingliste. Ein weiteres prominentes Framework, das auch in UIMA eingebunden werden kann, ist das seit 1995 federführend von Bob Carpenter entwickelte *LingPipe* (<http://alias-i.com/lingpipe/>). Auch hier finden sich Werkzeuge zur flachen Satzverarbeitung, der *named entity recognition*, Auflösung von Ko-Referenzen, Klassifikation, Clustering, korpusabhängigen Rechtschreibvorschlägen u.ä. Neben einer Mailingliste existiert ein Blog, in dem regelmäßig über Neuentwicklungen oder interessante Probleme berichtet wird. Beispiele für computerlinguistische Frameworks in .Net sind *SharpNLP* (<http://www.codeplex.com/sharpnlp>) und *Antelope* (<http://www.proxem.com/Default.aspx?tabid=55>).

Einführungen in Java bieten Ullenboom (2009) und Sierra und Bates (2008), während Hammond (2002) und Mason (2000) die computerlinguistische Programmierung mit Java beleuchten. Stellman et al. (2008) gibt eine Einführung in C# und Foord und Muirhead (2009) in IronPython.

3.9.4 Dynamische Sprachen: Perl und Python

Eine der prägnantesten Eigenschaften der in den letzten beiden Abschnitten betrachteten Sprachen war ihre **statische Typisierung**: Jeder Variable muss bei ihrer Deklaration explizit derjenige Datentyp vorangestellt werden, den sie enthalten soll. Zwar können dadurch Fehler bereits zur Übersetzungszeit (und nicht erst zur Laufzeit) vom Compiler erkannt werden und auch schon in dieser Phase Performanzoptimierungen realisiert werden, für die Programmierung ist dieses Vorgehen allerdings nicht sonderlich flexibel, beispielsweise wenn ein Variablen-
typ erst zur Laufzeit bekannt sein kann oder man unterschiedliche Datentypen in einer Container-Datenstruktur ablegen will. Den dazu konträren Ansatz, dass der Typ von Variablen nicht implizit angegeben werden muss, sondern zur Laufzeit ermittelt wird, wird durch die **dynamischen Programmiersprachen** (früher oft auch **Skriptsprachen** genannt) vertreten. Unter ihnen nahm lange Zeit **Perl** eine zentrale Stellung ein.

Perl wird seit 1987 vom Linguisten Larry Wall entwickelt (zum Verhältnis von Perl zu natürlicher Sprache siehe <http://www.wall.org/~larry/natural.html>) und stellt eine Mischung aus C- und Awk/Sed-Konstrukten sowie anderen Unix-Programmen zur einfachen Verarbeitung textueller Daten dar (*Perl makes easy things easy and hard things possible*). Herzstück dieser Sprache ist die Integration regulärer Ausdrücke – eine als *Perl Compatible Regular Expressions* (PCRE) bekannte C-Bibliothek bildet die Grundlage vieler Implementierungen regulärer Ausdrücke in anderen Programmiersprachen und Anwendungen. Durch die Vielzahl ihrer Einsatzgebiete bezeichnet man Perl auch als *Swiss Army Chainsaw*; das zentrale Motto dieser Sprache lautet *There is more than one way to do it* (TMTOWTDI-Prinzip), das eine Vielzahl syntaktischer Varianten zur Beschreibung eines Problems, ähnlich dem Gebrauch natürlicher Sprache, erlaubt. Daraüber hinaus unterstützt Perl das *Rapid Prototyping* dadurch, dass Quellcode nicht wie in C/C++ oder Java zur Ausführung jeweils manuell neu übersetzt werden muss; zur Laufzeit wird automatisch vom Perl-Interpreter Bytocode erzeugt, der anschließend sofort ausgeführt wird.

Obschon bei einer Variablen-deklaration nicht explizit angeben werden muss, ob es sich dabei um eine Zeichenkette, eine ganze Zahl oder ein Objekt handelt, unterscheidet Perl durchaus zwischen unterschiedlichen Variablen-typen, die durch ein Präfix gekennzeichnet werden. Diese sind: \$ für einen skalaren Wert, @ für ein Array (ein Container für Listen), % für Hashes, & für Funktionen und * für sogenannte *Typeglobs*, einer Struktur, in der die gesamte Symboltabelle eines Programms abgelegt ist. Betrachten wir zur Illustration wiederum unser Beispiel:

```

1 sub count_words {
2     my $fh = shift;
3     my $freq_ref;
4     while (<$fh>) {
5         chomp;
6         my @words = split ();
7         foreach my $word (@words) {

```

```

8     $freq_ref ->{$word}++;
9   }
10  }
11  return $freq_ref;
12 }

14 open (IN, $ARGV [0]) or die $!;
15 my %frequency = %{count_words (*IN)};
16 foreach my $freq (sort {$frequency {$b} <=> $frequency {$a}})
17   keys %frequency)
18 {
19   print "$freq -> $frequency {$freq}\n";

```

Analog zu den vorherigen Beispielen gibt es beim Wörterzählen in Perl eine `count_words`-Funktion und ein Hauptprogramm. Dieses wird im Unterschied zu den beiden anderen Programmen allerdings (für gewöhnlich) nicht extra gekennzeichnet. Schon im Aufruf der Funktion `count_words` (Zeile 15) macht Perl seinem Ruf, vorwiegend aus kryptischen Zeichenkombinationen zu bestehen, alle Ehre: Die Argumentliste besteht aus einer Referenz auf eine Dateizugriffskennung. Da der Rückgabewert der Funktion eine Referenz auf einen Hash ist, wird diese durch `%{...}` wieder aufgelöst, sodass der resultierende Hash direkt der Hash-Variablen `%frequency` zugewiesen werden kann.

Eine weitere Eigenart von Perl findet sich in der Funktion selbst: Besaßen die Funktionssignaturen in den vorherigen Beispielen jeweils eine explizite Argumentliste, findet sich diese nun implizit im Array `@_`. Da dieses das Standardargument einer Funktion ist, lassen sich die Werte auf dieser Liste durch Zugriff auf ihre Indizes oder durch einfaches Löschen und gleichzeitiges Zuweisen des ersten Werts mittels `shift` innerhalb der Funktion verwenden. Des weiteren muss man darauf achten, dass der Typ einer Referenz in Perl immer ein Skalar ist, wie man z. B. beim Aufbau des Hashs in Zeile 8 sieht: Dass wir hier tatsächlich eine Zeichenkette und die zu inkrementierende Häufigkeit in einem Hash speichern, lässt sich allein an dem Paar geschweifter Klammern ablesen.

Im Gegensatz zu den vorherigen Beispielen lässt sich hier die Sortierung des Histogramms in absteigender Reihenfolge der Werte ohne eine eigene Sortefunktion realisieren. In Zeile 16 iterieren wir über den Hash, indem wir die `sort`-Funktion einen Vergleichsausdruck evaluieren lassen, in dem zwei Werte aus dem Hash `%frequency` mittels des sogenannten *Raumschiffoperators* `<=>` und der eingebauten Variablen `$a` und `$b` numerisch miteinander verglichen werden.

Wenn man heute von Perl spricht, meint man meist das 1994 veröffentlichte Perl 5. Obwohl es seitdem mehrere Versionen durchlaufen hat (aktuell Perl 5.10), blieb seine Syntax über die Zeit stabil. Im Jahr 2000 hat Larry Wall mit Perl 6 eine vollständig objektorientierte und um Methoden aus der funktionalen Programmierung erweiterte Version der Sprache angekündigt, die dann in einer virtuellen Maschine namens *Parrot* ausgeführt werden soll. Obwohl dies als Langzeitprojekt angekündigt wurde, ist noch nicht abzusehen, wann diese Version veröffentlicht wird. Zwar sind schon große Teile der Spezifikation imple-

mentiert, doch hat die Perl-Community diesen Fortschritt noch nicht so recht kommunizieren können. Inwieweit sich dies auf gegenwärtige und zukünftige Benutzerzahlen auswirkt, bleibt abzuwarten.

Schwartz et al. (2008) ist seit Jahren die Einführung in Perl, während Conway (1999) objektorientierte und Dominus (2005) funktionale Programmierung in Perl beleuchten. Eine computerlinguistische Einführung stellt Hammond (2003) dar.

Einen weiteren bedeutenden Vertreter aus der Familie der dynamischen Programmiersprachen findet man in **Python**, das seit 1989 hauptsächlich von Guido van Rossum entwickelt wird. Ein Hauptaugenmerk liegt dabei auf der klaren Strukturierung der Sprache und ihrer einfachen Erlernbarkeit. So ermöglicht Python beispielsweise die parallele Verwendung verschiedener Programmierparadigmen, wie objektorientierter, strukturierter oder funktionaler Programmierung, ohne dass sich ein Bruch in der Sprache ergäbe. Ein wichtiges Merkmal von Python stellt die konsequente Verwendung von Leerraum zur Unterscheidung von Blöcken dar. Während andere Sprachen diese meist durch geschweifte Klammern gliedern, ist allein die konsistente Verwendung von Leerzeichen oder Tabulator-Eintrückungen zur Kennzeichnung von Blöcken vorgesehen. Anders als Perl verwendet Python zwar keine Präfixe zur Unterscheidung von Datentypen, doch findet man auch hier gewöhnungsbedürftige Notationen in Form von doppelten an- und abführenden Unterstrichen, die für interne Methoden, beispielsweise `__str__()` zur Zeichenkettenrepräsentation eines Objekts, verwendet werden. Widmen wir uns zur Veranschaulichung dieser Eigenschaften wiederum unserer Beispieldarstellung:

```

1 from sys import argv
2 from collections import defaultdict

4 def count_words (text):
5     frequency = defaultdict (int)
6     for line in text:
7         words = line.rstrip ().split ()
8         for word in words:
9             frequency [word] += 1
10    return frequency

12 def main ():
13     text = open (argv [1], 'r')
14     frequency = count_words (text)
15     print sorted (frequency.items (), key = lambda (k,v): (v,k),
16                   reverse = True)

17 if __name__ == '__main__':
18     main ()
```

Obwohl das *Zen of Python* (<http://www.python.org/dev/peps/pep-0020/>) mit dem Leitspruch *There should be one – and preferably only one – obvious way to do it* explizit gegen Perls TIMTOWDTI-Prinzip antritt, stellt das Wörterzählen in einen Hash (der in Python *Dictionary* heißt) ein gutes Gegenbeispiel

dar. Die traditionelle Lösung sieht so aus, dass man anhand der `get`-Methode zuerst schaut, ob das entsprechende Wort schon vorhanden ist; ist dies nicht der Fall, wird der Schlüssel zunächst mit 0 initialisiert, um dann sofort auf 1 gesetzt zu werden: `frequency [word] = frequency.get (word, 0) + 1` (eine syntaktische Variante dazu ist `frequency [word] = frequency.setdefault (word, 0) + 1`). Die Klasse `defaultdict` bekommt als erstes Argument eine `default_factory`, die in diesem Fall die Funktion `int ()` aufruft, wenn der Schlüssel noch nicht im Dictionary vorhanden ist. Da `int ()` standardmäßig 0 zurückgibt, wird das neue Wort mit diesem Wert im Dictionary initialisiert und dann auf 1 gesetzt.

Ein Beispiel für die Möglichkeit der funktionalen Programmierung in Python sehen wir in Zeile 16, in der die Schüssel-/Wertpaare nach der absteigenden Reihenfolge ihrer Häufigkeit sortiert ausgegeben werden. Die Signatur der `sorted`-Funktion besteht aus einem Objekt, das iteriert werden kann, einer optionalen Vergleichsfunktion, einem optionalen Argument `key`, dem eine Funktion übergeben wird, die einen Vergleichsschlüssel für das jeweilige Listenelement zurückgibt, sowie dem optionalen Argument `reverse`, das die Möglichkeit bietet, die Sortierreihenfolge umzukehren. In unserem Fall liefert die Methode `items ()` die Möglichkeit, über das Dictionary mit den Worthäufigkeiten zu iterieren. Das eigentliche Element funktionaler Programmierung findet sich nun in der Schüssel-Funktion: `lambda` ist ein Ausdruck, der aus Lisp übernommen wurde, und eine anonyme Funktion erzeugt, in der nur *ein* Ausdruck ausgewertet werden kann – in unserem Fall die Vertauschung von Schüssel und Wert zur Sortierung. `reverse = True` sorgt dann für die absteigende Reihenfolge.

Mit dem **Natural Language Toolkit** (NLTK; <http://www.nltk.org>) existiert auch für Python ein computerlinguistisches Framework, das seit 2001 hauptsächlich von Steven Bird, Ewan Klein und Edward Loper zur Einführung in die Sprachverarbeitung und in Python selbst entwickelt wird. Hauptbestandteile sind neben dem eigentlichen Code eine ausführliche Dokumentation, die ein Buch als Einführung in die computerlinguistische Programmierung umfasst, und Daten in Form von Korpora, Grammatiken und statistischen Modellen. Der Code gliedert sich in Klassen zur flachen Satzverarbeitung, Lemmatisierung, Parsing, semantischen Modellierung, Anbindung an die mitgelieferten Korpora und an WordNet, Clustering, Klassifikation, allgemeine symbolische und statistische Verfahren wie z. B. Unifikation oder *likelihood*-Methoden. Zur Kommunikation stehen Benutzer- und Entwickler-Mailinglisten zur Verfügung.

Die Einführung einer abwärtsinkompatiblen Version der Sprache, die Perl noch bevorsteht, wurde mit Python 3.0 im Dezember 2008 vollzogen: Während die kurz zuvor erschienene Version 2.6 bereits die Verwendung neuer Sprachelemente aus der Version 3.0 ermöglicht, dürfte es noch einige Zeit dauern, bis einerseits die vorhandenen Anwendungen und Frameworks portiert sind und sich andererseits die neuen Strukturen etabliert haben. Einführungen, die dieser Situation Rechnung tragen, bieten Lutz (2008) und Ernesti und Kaiser (2008), während Bird et al. (2009) in die computerlinguistische Programmierung mit NLTK einführt.

3.9.5 Von der Desktop- zur Web-Applikation

Eine weitere weitverbreitete Anwendung der in den beiden vorhergehenden Abschnitten vorgestellten Programmiersprachen ist der Popularisierung von Internetdiensten seit den 90er-Jahren v.a. in Form des World Wide Web geschuldet: Schon recht bald nach der Einführung der ersten Browser manifestierte sich der Wunsch, neben statischen auch dynamische HTML-Seiten programmatisch auf dem Server erzeugen zu können, bzw. eigene Programme im Webbrowser ausführen zu lassen. Serverseitig etablierte sich dazu eine **Common Gateway Interface** (CGI) genannte Technik, die es ermöglicht, bei einer Anfrage durch einen Client Argumente an eine dafür spezifizierte Anwendung weiterzureichen und die Ergebnisse der Verarbeitung an den Client zurückzugeben. Zur Beschleunigung dieser Anwendungen können die Interpreter der verwendeten Sprachen (vor allem Perl, PHP und Python) als Module in den Webserver integriert werden. Dort laufen sie permanent als eigenständiger Prozess und müssen nicht bei jeder Anfrage neu aufgerufen werden.

Bereits in der ersten, 1995 erschienenen, Java-Version konnten Programme als sogenannte *Applets* in einem eigenen von Sun *HotJava* genannten Browser bzw. als JVM-Plugins in anderen Browsern (v.a. im Netscape Navigator) ausgeführt werden. Diese Technik hat allerdings im Laufe der Jahre zugunsten von *JavaScript* (auch als *ECMA-Script* bekannt) an Bedeutung verloren. Um serverseitig dynamische HTML-Inhalte zu erzeugen, verwendet man sogenannte *Servlets*, Java-Klassen, die in webserverartigen Containern wie *Tomcat* oder *Jetty* ausgeführt werden. In .Net heißt die entsprechende Technologie *Active Server Pages* (ASP.Net) und besitzt wie das gesamte .Net-Framework vor allem den Vorteil, alle in der CLR ausführbaren Sprachen auch für die Webprogrammierung verfügbar zu machen.

Einen Schritt über die reine Einbindung dynamischer Inhalte hinaus gehen neuere Ansätze, die unter dem Begriff *Rich Internet Application* (RIA) zusammengefasst werden. Durch die Popularisierung von Datenformaten wie XML und JSON einerseits und der immer weiter gehenden Verbreitung von Datenbankmanagementsystemen wie z. B. *MySQL* oder *PostgreSQL*, und der Möglichkeit, hochperformante Suchmaschinen beispielsweise aus Frameworks wie *Apache Lucene*, mit einem überschaubaren Einsatz von Ressourcen an die eigenen Erfordernisse anpassen zu können andererseits, verlagert sich die Entwicklung zum mindest prototypischer Anwendungen immer stärker ins Web. Dazu tragen v.a. auch Webframeworks wie *Ruby on Rails* oder das in Python implementierte *Django* bei, die Entwicklern durch Bereitstellung einfach miteinander zu kombinierender Funktionskomponenten schnelle und einfache Implementierungen ermöglichen. Darüber hinaus sorgen Technologien wie *Asynchronous Javascript and XML* (Ajax) dafür, dass sich Oberflächengestaltungen – und ein entsprechendes Handling durch den Benutzer – realisieren lassen, die der gewohnten graphischen Benutzeroberfläche von Desktopanwendungen gleichwertig sind.

4 Ressourcen

Kapitelherausgeber: Kai-Uwe Carstensen

Ohne die Existenz bzw. die Entwicklung computerlinguistischer Ressourcen ist der theoretische und praktische Fortschritt in der Computerlinguistik heute nicht mehr vorstellbar. Die Verfügbarkeit entsprechender großer Datenmengen (z. B. **Textkorpora** und **Sprachdatenbanken**) ist mittlerweile meist Voraussetzung für computerlinguistische Tätigkeiten, sowohl im Rahmen empirisch abgesicherter theoretischer Untersuchungen als auch im Rahmen der Entwicklung praktischer Verfahren für die Sprachtechnologie. Natürlichsprachliche Systeme verfügen heute in der Regel über umfangreiche datenintensive Komponenten, und zwar sowohl sprachliche (z. B. **Lexika**) als auch nicht-sprachliche (zur Repräsentation **nicht-sprachlichen Wissens**). Auch das Testen einzelner Komponenten oder die Evaluierung eines gesamten Systems geschieht in zunehmendem Maße mithilfe ausgedehnter Testsets (Datenbanken aufbereiteter und unaufbereiteter sprachlicher Daten), um eine qualitativ hochstehende objektive Bewertung zu gewährleisten.

Die Quellen dieser Ressourcen sind heterogen. Nicht selten ist es immer noch die Kompetenz des theoretischen Linguisten, auf die zurückgegriffen wird (vgl. den Beitrag über **lexikalisch-semantische Ressourcen**). Oder es ist die sprachliche Performanz idealerweise ausgewählter und nach verschiedenen Kriterien kategorisierter Probanden, die —z. B. beim Aufbau von Sprachdatenbanken— benötigt wird. Am häufigsten wird jedoch versucht, bereits vorliegende sprachliche Daten automatisch zu erschließen und nach der Anwendung computerlinguistischer Analysemethoden in entsprechend annotierter Form abzulegen (**Baumbanken**). Die bedeutendste Quelle hierfür ist mittlerweile sicherlich das **World-Wide-Web** bzw. ausgewählte Teile davon (z. B. **Wikipedia**).

Neben dem Sammeln computerlinguistisch relevanter Information ist deren interne Darstellung (Datenmodelle, Repräsentationsformate, Markup-Sprachen) unter dem Aspekt der *Wiederverwertbarkeit* ein entscheidender Faktor. Während in den Anfängen der Computerlinguistik oft systemspezifische Ressourcen erstellt wurden, die außerhalb ihrer Umgebung unbrauchbar waren, wird heutzutage versucht, sich an allgemein akzeptierten Standards (z.B. XML) zu orientieren und die Entwicklung der Repräsentationssprachen daran auszurichten.

Das Sammeln, Aufbereiten, Verwalten und Zur-Verfügung-stellen computerlinguistischer Daten ist heute ein eigenständiger und zunehmend wichtiger Teilbereich der Computerlinguistik, der in diesem Kapitel vorgestellt werden soll.

4.1 Korpora

Heike Zinsmeister

Im vorangehenden Kapitel zu den computerlinguistischen Methoden wurden an mehreren Stellen linguistische Korpora erwähnt, die als empirische Datengrundlage dienen und zum Trainieren von (statistischen) Programmen oder allgemein zum Testen eingesetzt werden (siehe z. B. Unterkapitel 3.4 und Abschnitt 3.7.2). Korpora können zudem als Zeugnisse für die Möglichkeiten computerlinguistischer Verarbeitung betrachtet werden, da sie oftmals bis zu einem gewissen Grad auf automatischer oder semi-automatischer Vorverarbeitung und Annotation basieren.

Unabhängig von der Verwendung in der Computerlinguistik kann ein linguistisches **Korpus** (neutr. *das Korpus*) definiert werden als Sammlung gesprochener oder schriftlicher Äußerungen, die digital erfasst, also auf Rechnern gespeichert und maschinenlesbar sind, und für eine linguistische oder computerlinguistische Aufgabe aufbereitet wurden. Von den eigentlichen Korpora unterscheiden sich **Textarchive**, die ebenfalls digitalisierte Sprachdaten enthalten, welche aber nicht primär für linguistische Zwecke bereitgestellt werden. Ein bekanntes Textarchiv ist das internationale **Gutenberg Project** (Lebert 2008) mit Texten, deren Urheberrecht abgelaufen ist bzw. deren Autoren die Texte zur Nutzung freigegeben haben. Reine **Belegsammlungen** unterscheiden sich ebenfalls von Korpora, indem sie nur einzelne Sätze oder Paragraphen aufführen und nicht ganze Texte oder zumindest substanzelle Ausschnitte aus diesen. Belegsammlungen enthalten mitunter auch konstruierte und bewusst ungrammatische Beispielsätze – Korpora hingegen **authentische Sprachdaten**, die in einer linguistisch unreflektierten Kommunikationssituation produziert wurden. Belegsammlungen bieten Evidenz für bestimmte linguistische Phänomene. Die lexikalisch orientierte Sammlung CoDII (*Collection of distributionally idiosyncratic items*, Trawinski et al. 2008) z. B. beinhaltet korpusbasierte Belege für Lizenzierungsbedingungen von negativ polaren Ausdrücken (wie *sich scheren um*), die nur zusammen mit einer Negation oder in anderen spezifischen Kontexten auftreten.

Neben der oben genannten Korpusdefinition wird in der Praxis auch ein anwendungsbezogener Korpusbegriff verwendet. Nach diesem wird jeder Text als Korpus bezeichnet, wenn er für linguistische oder computerlinguistische Aufgaben genutzt wird. Dies schließt auch Texte mit ein, die nicht speziell für die Linguistik/Computerlinguistik aufbereitet wurden, sondern in einer unstrukturierten Rohfassung vorliegen wie z. B. Texte, die aus dem HTML-Code von Internetseiten extrahiert oder aus Textarchiven entnommen wurden. Unter diesen weiter gefassten Korpusbegriff fallen auch Sprachdaten, die nicht digital erfasst sind, sondern z. B. nur als magnetische Tonbandaufnahmen oder in gedruckter Form vorliegen (z. B. Ruoff 1984).

Im Folgenden geht es vorwiegend um digitale, aufbereitete Korpora. In Abschnitt 4.1.1 wird die generelle Architektur eines einzelnen Korpus thematisiert,

in Abschnitt 4.1.2 eine umfassende Typologie, die dabei hilft, Korpora anhand verschiedener Eigenschaften zu systematisieren. Die Typologie soll auch dazu dienen, einige bekannte Korpusressourcen vorzustellen. Wozu Korpora in der Computerlinguistik verwendet werden, wird in Abschnitt 4.1.3 erklärt. Abschnitt 4.1.4 bietet schließlich Hinweise auf weiterführende Arbeiten und Ressourcen.

4.1.1 Aufbau eines Korpus

Ein aufbereitetes Korpus besteht aus drei Schichten: den Sprachdaten, den analysierenden Annotationen und den beschreibenden Metadaten. Um die Daten verschiedener Projekte austauschbar und vergleichbar zu halten, haben internationale Initiativen Standardisierungsempfehlungen für Annotationen, Metadaten und die allgemeine Datenstruktur und Enkodierung erarbeitet, auf die am Ende dieses Abschnitts kurz eingegangen wird.

Sprachdaten

Den Kern eines Korpus bilden die **Sprachdaten**, die aus Texten, Sprachaufnahmen oder deren Verschriftlichungen bestehen und die in digitalisierter Form abgespeichert sind. Sie können auf sprachlichen **Primärdaten** basieren, die z. B. als Tonaufnahme oder Textveröffentlichung unabhängig vom Korpus existieren. Je nach Art der Primärdaten unterscheidet man **Textkorpora** von **Korpora der gesprochenen Sprache**. Liegen in einem Korpus der gesprochenen Sprache die Primärdaten selbst vor – die physikalisch messbaren Sprachsignale – und nicht nur eine **schriftliche Transkription**, kann das Korpus in einer **Sprachdatenbank** verwaltet werden (siehe Unterkapitel 4.5). Textuelle Primärdaten können bereits in digitaler Form vorliegen, müssen es aber nicht. Sie können auch nur als gedruckte Texte, Handschriften oder ähnliches zur Verfügung stehen. Existieren die Primärdaten als konkrete Veröffentlichungen, besitzen sie nicht nur einen Wortlaut, sondern auch eine äußere Form: die Verteilung des Texts auf einer oder mehreren Seiten, die Größe, die Farbe und der Font der Buchstaben usw. Primärtextliche Eigenschaften dieser Art sind selten in den Sprachdaten kodiert, die im Korpus als Grundlage für die weitere Analyse genutzt werden.

Annotation

Als zweite Schicht lagern sich verschiedene **Annotationsebenen** um die Sprachdaten. Die erste Analyseebene besteht aus der **Segmentierung** (auch Segmentation), der Zerlegung des Sprachsignals oder der Zeichenkette in linguistisch definierte Einheiten wie Phoneme, Wörter oder Sätze (siehe auch Unterkapitel 3.4). In Sprachdatenbanken bilden **Transkriptionen** die nächste Analyseebene, wobei der Wortlaut einer Äußerung orthographisch als Text wiedergegeben werden kann oder ihre Lautung in phonetischen oder phonemischen Symbolen. Gegebenenfalls werden auch non-verbale Geräusche notiert wie Räuspern oder Lachen. In Textkorpora können bei der Segmentierung auch **textstrukturelle Einheiten** wie Paragraphen, Kapitel, Überschriften oder Fußnoten abgegrenzt

werden. Die Segmentierung kann indirekt kodiert sein z. B. durch die Konvention, dass Tokengrenzen durch Leerzeichen markiert sind, Satzgrenzen durch Zeilenumbrüche und Paragraphengrenzen durch Leerzeilen. Besser ist eine Annotation, welche die Sprachdaten von der Analyse explizit trennt, indem die textstrukturellen Einheiten durch Annotationslabel benannt werden und den Sprachdaten mit Hilfe einer **Auszeichnungssprache** wie **XML** (eXtensible Markup Language) zugeordnet werden (vergleiche auch Unterkapitel 2.5).

Auf der Basis der Segmentierung können weitere **linguistische und außerlinguistische Annotationsebenen** vorliegen. Sehr oft enthalten Korpora eine Annotation der Wortart (englisch *part of speech*, POS) und der Basiswortform (englisch *lemma*). Allgemeine Empfehlungen zur POS-Annotation wurden von der Text Encoding Initiative formuliert (TEI AI1W2 1991). Für das Deutsche wurden davon die STTS-Guidelines (Stuttgart-Tübingen-Tagset) abgeleitet (Schiller et al. 1999). **Syntaktische Annotationen** findet man zur Konstituentenstruktur und zu grammatischen Funktionen (Marcus et al. 1993; Marcus et al. 1994; Brants et al. 2002), Dependenz (Hajičová et al. 1999, Foth 2006) und für das Deutsche auch zu topologischen Feldern (Telljohann et al. 2006). **Semantische Annotationen** beinhalten Lesarten (*word senses*; Fellbaum 1998), semantische Rollen und semantische Frames (Palmer et al. 2005; Meyers et al. 2004; Burchardt et al. 2006) sowie Tempus und Aspekt (Pustejovsky et al. 2003). Zu den **diskursbezogenen Annotationen** gehören Koreferenzphänomene (Poesio 2000; Naumann 2006), Informationsstatus (Nissim et al. 2004; Riester 2008), Informationsstruktur (Calhoun et al. 2005; Götze et al. 2007), Diskursrelationen (Mann und Thompson 1988; Miltakaki et al. 2004) und Dialogakte (Anderson et al. 1991; Carletta et al. 1997; Alexandersson et al. 1998). In Lernerkorpora zum Erst- oder Fremdspracherwerb werden **Fehler** annotiert (MacWhinney 1995; Granger 2002; Lüdeling 2008), ebenso in Korpora zu gestörter Sprache. Über die rein linguistische Analyse hinaus gehen z. B. die Annotationen von **Emotionen** und **Meinungen** (Wiebe et al. 2004), ebenso die Analyse von **Mimik** (Foster 2007) und **Gestik** (Martell 2002; Kipp et al. 2007).

Um eine konsistente Annotation zu erreichen und auch für jede spätere Nutzung ist es sehr wichtig, dass die Annotationen ausführlich dokumentiert sind. Die Bedeutung der **Annotationslabel** (*tags*) werden in einem **Tagset** eindeutig definiert und die **Annotationskriterien** in Richtlinien (*guidelines*) mit Beispielen belegt. Für die Dokumentation der **Annotationsqualität** wird die Übereinstimmung unter den Annotatoren festgehalten (*inter-annotator agreement*; Artstein und Poesio 2008).

Das Korpus in den Abbildungen 4.1 und 4.2 verwendet für die Wortartenannotation das **STTS-Tagset**. Hier steht das Label *ART* für *Artikel* und *NN* für *Normales Nomen*, welches als Appellativum definiert ist und sich von Eigennamen abgrenzt. Die STTS-Richtlinien geben neben der Definition auch eine Reihe von Beispielen an und verweisen auf jeweils bekannte Grenzfälle zu anderen Labels.

```
<sentence editor="shartung" date="2004083117:26:19" origin="T990430.196">
  <word form="Der" pos="ART"/> <word form="Scheibenwischer" pos="NN"/>
  <word form="quietscht" pos="VFIN"/> <word form"." pos="$."/>
</sentence>
```

Abbildung 4.1: Ausschnitte aus der XML-Inline-Annotation von Satz 20209 der TüBa-D/Z *Der Scheibenwischer quietscht.*

```
<body>Der Scheibenwischer quietscht.</body>

<mark id="tok_1" xlink:href="#xpointer(string-range{//body,'',1,3))"/>
<mark id="tok_2" xlink:href="#xpointer(string-range{//body,'',5,15))"/>
<mark id="tok_3" xlink:href="#xpointer(string-range{//body,'',21,9))"/>
<mark id="tok_4" xlink:href="#xpointer(string-range{//body,'',30,1))"/>

<feat xlink:href="#tok_1" value="sts.type_pos.xml#ART"/>
<feat xlink:href="#tok_2" value="sts.type_pos.xml#NN"/>
<feat xlink:href="#tok_3" value="sts.type_pos.xml#VFIN"/>
<feat xlink:href="#tok_4" value="sts.type_pos.xml#DOLLAR_PERIOD"/>

<mark id="s_20209" xlink:href="#xpointer(id('tok_1')/range-to(id('tok_4')))">
```

Abbildung 4.2: Ausschnitte aus der XML-Standoff-Annotation von Satz 20209 der TüBa-D/Z im PAULA-Format

In Abbildung 4.1 ist jedes Wort und auch der Satzpunkt in ein XML-Element *word* als Wert des Attributs *form* eingebettet. Informationen über Leerzeichen gehen in dieser Kodierung verloren. Abbildung 4.2 stellt ein alternatives Format (vereinfacht) dar. Hier werden die Token durch Bezugnahme auf Buchstabenpositionen in der Zeichenkette definiert. Das erste Token *tok_1* (*Der*) beginnt an Position 1 und ist drei Zeichen lang. Das zweite Token *tok_2* (*Schweibenwischer*) beginnt an Position 5 und ist fünfzehn Zeichen lang. Die Wortarten-Label werden über Links den Token zugeordnet. Ebenso ist die Satzausdehnung in Relation zu den Token definiert. Beim ersten Beispiel sind verschiedene Annotationsebenen in einem **Inline-Format** gekoppelt. Die Wortform und das Wortarten-Label sind Attribute eines gemeinsamen Elements *word* und die syntaktische Hierarchie (hier vereinfacht nur die Satz- und Wortebene) ist durch die Einbettung der XML-Elemente (*sentence*, *word*) nachgebildet. Anders im zweiten Beispiel. Dort werden in einem **Standoff-Format** alle Informationsebenen (Text, Token, Wortart, Satz) getrennt aufgeführt. Sie sind lediglich über *Pointer* und *Links* indirekt miteinander verbunden.

Metadaten

Metadaten werden auch als *Daten über Daten* bezeichnet. In ihnen werden die Primärdaten, die im Korpus enthaltenen Sprachdaten und die Annotations beschrieben. Sie erfassen z. B., welchen Textgattungen die Daten zugehören, wie groß der Datenumfang ist und wie die Sprachdaten kodiert sind. Außerdem

werden kontextuelle Aspekte der Entstehung des Korpus dokumentiert, z. B. die Entstehungs- und Publikationszeiten der Primärdaten, der Publikationsort, beteiligte Personen, die Entstehungszeit der Annotation und die Namen der Annotatoren. Zusätzlich findet man Verweise auf externe Quellen wie die Definitionen der Annotationslabel (der Tagsets), Annotationsrichtlinien und Publikationen, die das Korpus beschreiben. Eine nicht unerhebliche Information ist die Angabe von **urheberrechtlichen Eigenschaften** des Korpus und seiner Primärdaten. Außer diesen Angaben über die Daten und die Annotationen findet man auch Informationen über die Metadaten selbst, z. B., ob die Metadaten manuell oder automatisch erstellt wurden und ob sie einem bekannten **Standard** folgen.

Standardisierung

Immer kürzere Zyklen in der Hard- und Softwareentwicklung gefährden die nachhaltige Nutzbarkeit von Korpora. Deshalb sind projektübergreifende, einheitliche Beschreibungen und Formate wichtig, die es erleichtern, Korpusdaten auch über die jeweilige Projektlaufzeit hinaus nutzbar zu halten (Schmidt et al. 2006; Zinsmeister et al. 2008). Ressourcen, die sich am selben Standard orientieren, können zudem besser miteinander verglichen und kombiniert werden.

Von der **Dublin Core Metadata Initiative** (DC) wurde Mitte der 1990er Jahre erstmals eine Kermenge von Metadaten für die Beschreibung elektronischer Ressourcen definiert. Im Standard der **Open Language Archive Community** (OLAC) wurde der Dublin Core für mehrsprachige und multimodale Ressourcen, die Text-, Bild- und Audiomaterial verbinden, erweitert. Alternativ hat die **ISLE Meta Data Initiative** (IMDI) ebenfalls einen Metadatenstandard auf der Basis des Dublin Core vorgeschlagen. Die seit mehr als zwanzig Jahren bestehende **Text Encoding Initiative** (TEI) definiert im TEI-Header einen eigenen Satz von Metadaten, mit dem Ziel, speziell Textdokumente und Korpora zu archivieren. Da, wo das DC-Metadatenset z. B. nur ein unspezifisches Element *source* vorsieht, bietet das TEI-Set spezielle Elemente für bibliographische Angaben, so dass deren Bestandteile wie *editor* und *edition* in spezifischen Feldern abgelegt werden können.

Neben den Metadaten schlug die TEI auch ein Standardformat für die Korpusannotation vor. Darauf aufbauend hat ein internationales Gremium zur Standardisierung von sprachtechnologischen Ressourcen, die Expert Advisory Group on Language Engineering Standards, **EAGLES**, Empfehlungen erarbeitet (z. B. Leech et al. 1996 für POS-Annotation), die im **Corpus Encoding Standard** (CES bzw. dem XML-basierter Nachfolger XCES) umgesetzt wurden und die TEI-Kategorien um sprachtechnologisch relevante Kategorien erweiterten.

4.1.2 Typologie

Korpora lassen sich anhand einer Reihe von Kriterien klassifizieren. In Anlehnung an die Korputypologie in Lemnitzer und Zinsmeister (2006, Kap. 5) werden in den folgenden Abschnitten eine Reihe von Ressourcen vorgestellt.

Weil die Erstellung von Korpora relativ zeit- und kostenintensiv ist, besteht der Anspruch, dass ein Korpus möglichst **wiederverwendbar** und **multifunktional** einsetzbar sein sollte. Der **ursprüngliche Verwendungszweck** eines Korpus legt zwar dessen weitere Nutzung nicht fest, kann aber bestimmte Eigenschaften des Korpus erklären. Im Projekt **Verbmobil** z. B. wurden für die Entwicklung eines Übersetzungssystems für Spontansprache mehrsprachige Korpora erstellt und annotiert (Burger et al. 2000; Jekat und v. Hahn 2000). Um das Vorhaben handhabbar zu halten, wurde die sprachliche Domäne auf Terminverhandlungen zwischen Geschäftspartnern, Reiseplanungen und Hotelreservierungen beschränkt. Hierfür wurden spontane Dialoge auf Deutsch, Englisch und Japanisch aufgenommen, von denen Teilkorpora für die Entwicklung und das Testen einer integrierten Grammatikkomponente mit syntaktischer Information annotiert wurden (TüBa-D/S, TüBa-E/S und TüBa-J/S; Hinrichs et al. 2000). Obwohl die daraus resultierenden Baumbanken nur die genannten Domänen abdecken, können sie unabhängig vom Verbmobil-Projekt für andere Forschungsfragen zur Syntax bei gesprochener Sprache und Dialogen eingesetzt werden. Das Brown University Standard Corpus of Present-Day American English (kurz: **Brown Corpus**) wurde anders als die Verbmobil-Korpora von vornherein als **repräsentatives Korpus** geplant (Francis und Kučera 1979). Es sollte die Gesamtheit des schriftlich veröffentlichten amerikanischen Englisch des Jahres 1961 repräsentieren und umfassende Analysen und computerbasierte Auswertungen erlauben. Dafür wurden nach systematischen Kriterien Exzerpte von bis zu 2000 Wörtern aus 155 Texten unterschiedlicher Textgenres entnommen. Das Brown-Korpus etablierte sich als Standard und wurde in vielen computerlinguistischen Arbeiten genutzt. Für das britische Englisch wurde nach den selben Kriterien das Lancaster-Oslo/Bergen (LOB) Corpus erstellt und für das Deutsche das LIMAS-Korpus.

Die **Sprachenauswahl** bezieht sich auf die Sprache der Primärdaten. **Mono-linguale Korpora** enthalten nur eine Sprache, **bi- und multilinguale Korpora** zwei oder mehrere Sprachen. Handelt es sich um Quelltexte einer Sprache und deren Übersetzungen in eine oder mehrere andere Sprachen, spricht man von **Parallelkorpora**. Mehrsprachige Sammlungen zu vergleichbaren Diskursbereichen, bei denen die Texte keine unmittelbaren Übersetzungen von einander sind, werden als **Vergleichskorpora** bezeichnet. In den Übersetzungswissenschaften wird der Begriff *Vergleichskorpus* etwas anders verwendet. Er beschreibt dort monolinguale Korpora, welche sowohl Originaltexte als auch übersetzte Texte in derselben Sprache enthalten. Für computerlinguistische Anwendungen sind besonders solche bi- und multilingualen Parallelkorpora relevant, bei denen die parallelen Texte auf Paragraphen-, Satz- oder Wortebene **aligniert** vorliegen, so dass die Texteinheiten der Übersetzung den jeweiligen Texteinheiten des Quelltextes zugeordnet werden. Ein häufig zitiertes Korpus ist das European Parliament Proceedings Parallel Corpus (kurz: **Europarl Corpus**, Koehn 2005), das auf Mitschriften und Übersetzungen von Debatten des Europäischen Parlaments beruht. Es umfasst Sprachpaare von elf europäischen Sprachen.

Ein weiteres Kriterium für die Klassifizierung von Korpora ist das **Medium**, in dem die Primärdaten entstanden bzw. erfasst wurden. Man unterscheidet Kor-

pora der **geschriebenen Sprache**, Korpora der **gesprochenen Sprache** und **multimodale Korpora**. Bei multimodalen Korpora werden die Primärdaten mit verschiedenen Medien erfasst, oft werden Audio- mit Videoaufnahmen kombiniert, so dass auch non-verbale Kommunikationsaspekte ausgewertet werden können wie beim **Smartkom-Korpus im Bayerischen Archiv für Sprachsignale** (BAS), bei dem Gestik, Mimik und Augenbewegung mit einbezogen wurden, um verschiedene Interaktionen zwischen Mensch und Maschine zu untersuchen (Schiel et al. 2002). Die Einordnung eines Korpus in geschriebene oder gesprochene Sprache ist im Einzelfall nicht trivial. Ist die Aufnahme einer ausformulierten Ansprache ein Beleg für gesprochene Sprache? Sind E-Mails oder Protokolle aus Chat-Räumen wie im **Dortmunder CHAT-Korpus** Belege für geschriebene Sprache? Um die Daten angemessen beschreiben zu können, bedarf es einer detaillierteren Klassifikation, die nicht nur das Medium der sprachlichen Realisierung berücksichtigt, sondern auch deren konzeptuellen Hintergrund. Die Text Encoding Initiative sieht daher für die Angabe des Mediums in den Metadaten eines Korpus die Werte *spoken to be written* und *written to be spoken* vor. Die Transkription der Aufnahme einer ausformulierten Ansprache wäre demnach *written to be spoken*.

Die **Annotation** ist ein weiteres Unterscheidungskriterium. Eine Reihe von Korpora basieren auf den selben Sprachdaten und unterscheiden sich nur durch ihre Annotationsebenen. Das markanteste Beispiel dafür sind Daten aus dem **Wall Street Journal** (WSJ) Subkorpus, das einen Teil der **Penn Treebank** bildet (welche zusätzlich u. a. die Daten des Brown-Korpus beherbergt). Das WSJ-Subkorpus beinhaltet Annotation der Wortart, eine syntaktische Analyse, die von der Rektions- und Bindungstheorie (Chomsky 1981) inspiriert ist sowie die Angabe von grammatischen Funktionen (siehe auch Unterkapitel 4.2). Die selben Daten wurden im **PropBank-Projekt** und im **NomBank-Projekt** mit Prädikat-Argumentstrukturen für Verben bzw. Nomen versehen. Teile davon sind auch in der **Penn Discourse Bank** und der **TimeBank** enthalten.

Korpora variieren stark in ihrer **Größe**. Neben vielen kleinen Korpora existieren langfristig angelegte Großprojekte. Die erste Generation digitaler Korpora wie das Brown Corpus beinhaltet eine Million Wortformen. Die zweite Generation, zu der das **British National Corpus** (BNC) gehört, umfasst bis zu 100 Millionen Wortformen. Korpora der dritten Generation gehen weit über die bisherigen Größenordnungen hinaus. Die aus dem Web extrahierten und automatisch aufbereiteten Korpora von **WaCky** (*Web as Corpus kool ynitiativ*) beinhalten jeweils mehr als eine Milliarde Token. Die Texte sind automatisch vom HTML-Code und von Duplikaten bereinigt, segmentiert und POS-getaggt. Das deutsche deWaCky-Korpus z. B. hat eine Größe von 1 278 177 539 Token oder 25,9 GB (Baroni et al. 2009). Ein anderes Beispiel ist die ein Terabyte große Sammlung von Google (Brants und Franz 2006), auch wenn sie nach der eingangs genannten Definition kein Korpus im eigentlichen Sinn darstellt, weil sie nur eine Sammlung von Wort-Quintupeln mit Frequenzangaben ist und keine fortlaufenden Texte enthält. Für viele statistische Anwendungen in der Computerlinguistik sind diese Wortketten vollkommen ausreichend. Viele Algorithmen arbeiten sogar nur auf der Basis von Dreierketten (Trigrammen, siehe Unterkapitel 4.2).

pitel 2.4). Andere Megakorpora werden von Wörterbuchverlagen verwaltet. Das englische COBUILD-Korpus, genannt die **Bank of England**, mit über einer halben Milliarde Token, ist ein gemeinsames Produkt der Universität Birmingham und des Verlages Harper-Collins. Der Duden-Verlag pflegt ein deutschsprachiges Megakorpus mit mehr als 1,3 Milliarden Token, welches aber nicht frei verfügbar ist. Die größte deutschsprachige Korpusammlung findet sich am **Institut für Deutsche Sprache** in Mannheim (IDS). Dort stehen Korpora mit insgesamt mehr als zwei Milliarden Token zur Verfügung. Eine Teilmenge davon ist auch online durchsuchbar. Linguistisch nicht aufbereitet, aber frei verfügbar sind die **XML-Dumps** von Wikipedia, in denen einzelsprachliche Versionen der Internet-Enzyklopädie gespeichert werden.

Korpora unterscheiden sich in Bezug auf die **Persistenz** ihrer Daten. Nicht alle Korpora basieren auf einem **statischen Datensatz**. **Monitorkorpora** wie das Mannheimer Morgen-Korpus des Instituts für Deutsche Sprache oder die bereits genannte Bank of England wachsen permanent, weil immer neue Daten eingepflegt werden. Ein Monitorkorpus der anderen Art liegt der Belegsammlung Wortwarte zugrunde, in der seit dem Jahr 2000 Wortneubildungen dokumentiert werden. Das zugrundeliegende Korpus besteht aus dem täglichen Online-Angebot von Zeitschriften und wird aus urheberrechtlichen Gründen nicht gespeichert.

Das nächste Kriterium ist der **Sprachbezug**, der wieder stark mit dem ursprünglich intendierten Anwendungszweck zusammenhängt. Man unterscheidet **Referenzkorpora**, die versuchen, eine Sprache in ihrer Gesamtheit zu vertreten, wie das British National Corpus oder das **Kerncorpus** des Digitalen Wörterbuchs der deutschen Sprache (DWDS), von **Spezialkorpora**, die sich auf Sprachdaten bestimmter eingeschränkter Domänen beschränken. Um ein **ausgewogenes, repräsentatives Korpus** zu erhalten, werden sorgfältige **Designkriterien** (*sampling criteria*) entwickelt. Allerdings können auch Referenzkorpora immer nur eine Annäherung an eine Sprache sein, da man die Grundgesamtheit einer Sprache nicht wirklich erfassen kann. Welche Belege sollte man für ein Korpus der *deutschen Sprache* zusammenstellen? Umgangssprachliche und dialektale Äußerungen, offizielle Statements, Zeitungsartikel, Romane, Gesetzesstücke, E-Mails und Chat-Konversationen, Lyrik, Texte aus dem 18. oder 19. Jahrhundert, die Bibelübersetzung von Martin Luther (als Beginn des Neuhighdeutschen)? Neben Spezial- und Referenzkorpora gibt es **opportunistischen** Sammlungen, bei denen aus pragmatischen Gründen auf Designkriterien verzichtet wird. Ein prominentes Beispiel dafür ist das bereits genannte Wall Street Journal Subkorpus der Penn Treebank.

4.1.3 Anwendungen

Allgemein sind linguistische Korpora eine wertvolle empirische Datenressource. In der (Computer-)Linguistik kommen sie auf verschiedene Arten zum Einsatz, von denen die gängigsten kurz vorgestellt werden.

Die **Qualität** von computerlinguistischen Analyseprogrammen wird oft durch einen automatischen Abgleich mit manuell erstellten Referenzdaten, dem soge-

nannten **Goldstandard**, getestet. In industrienahen Projekten bezeichnet man das Referenzkorpus auch als Benchmark-Korpus. Je nach Anwendung werden unterschiedliche Qualitätsmaße verwendet. Oft handelt es sich um Varianten der ursprünglich aus dem Information Retrieval stammenden Maße **Präzision** (englisch *precision*) und **Abdeckung** (englisch *recall*). Um verschiedene Evaluationsergebnisse besser vergleichen zu können, wird in der Literatur oft der sogenannte **F-Wert** angegeben, der harmonische Mittelwert aus Präzision und Abdeckung.

Bei der **Entwicklung von korpusbasierten Programmen**, insbesondere beim Einsatz von maschinellen Lernverfahren (vgl. Unterkapitel 2.4), wird das Korpus dazu anfänglich geteilt. Ein Teil des Korpus wird als **Trainingskorpus** genutzt, auf dessen Basis z. B. die Regeln des Programms und Wahrscheinlichkeiten (manuell oder automatisch) abgeleitet werden können. Ein zweiter Teil wird als **Entwicklungs korpus** eingesetzt, der zum Testen während der Programmierung und zur Erweiterung bzw. Verbesserung des Programms dient. Ein dritter und letzter Teil des Korpus bildet schließlich das **Testkorpus**. Diese Daten sollten während der Programmierung nicht betrachtet werden, so dass das fertig gestellte Programm auf diesen bis dahin ungesehenen Daten objektiv getestet werden kann (die Testdaten werden auch als **Heldout Data** bezeichnet). Eine mögliche Teilung ist 80% Trainings-, 10% Entwicklungs- und 10% Testkorpus (Jurafsky und Martin 2009, S. 92). Eine alternative Teilungsmöglichkeit besteht in der **k-fachen Kreuzvalidierung** (englisch *k-fold cross validation*). Bei der zehnfachen Kreuzvalidierung (*ten-fold cross validation*) z. B. wird das Korpus in zehn gleich große Teile geteilt. Das Programm wird auf neun der zehn Teile trainiert und auf dem zehnten getestet. Dies wird insgesamt zehnmal durchgeführt, so dass jedes Korpusteil einmal als Testkorpus zum Einsatz kommt. Als Ergebnis wird dann der Mittelwert aller zehn Evaluierungsergebnisse angegeben. Es wird davon abgeraten, bei der Korpusteilung zusammenhängende Textblöcke zu wählen. Man sollte besser eine zufällig gestreute Auswahl treffen und z. B. nur jeden zehnten Satz für das Testkorpus extrahieren.

Neben der Entwicklung und dem Testen von korpusbasierten Programmen, dienen Korpora auch als Testbett zur **empirischen Untermauerung** von (computer)linguistischen Theorien. Generative Grammatiktheorien wie die Head Driven Phrase Structure Grammar (**HSPG**) und die Lexical-Functional Grammar (**LFG**) wurden in Parsern implementiert und an realen Korpusdaten getestet (Oepen et al. 2002; Zinsmeister et al. 2002).

In der **Lexikographie** spielen Korpora seit jeher eine große Rolle, um Lesarten von Wörtern zu identifizieren und Beispiele für den Wortgebrauch zu finden. „You shall know a word by the company it keeps“, fasst Firth (1968, S. 179) den Ansatz des Kontextualismus zusammen, aus dem das für die Lexikographie wichtige Konzept der **Kollokation** stammt: das habituelle gemeinsame Auftreten von zwei oder mehreren Wörtern. Im Deutschen *putzt* man z. B. *seine Nase*, während man sie im Englischen *bläst* (*blow your nose*). Kollokationen können über relative Auftretenshäufigkeiten in Korpora ermittelt werden, vgl. Unterkapitel 5.2.

Im **Sprachunterricht** wurden Korpora traditionell nur indirekt genutzt, z. B. als Datenressource für die Erstellung von Unterrichtsmaterialien. Der unmittelbare Einsatz von Korpora im Unterricht ist eine neue Entwicklung (Mukherjee 2002; Nesselhauf 2004; Bick 2005).

In der Linguistik besteht eine lange Tradition der Verwendung von Korpora z. B. in Teildisziplinen wie der historischen Linguistik und der Spracherwerbsforschung. In der Mitte des zwanzigsten Jahrhunderts grenzten sich theoretisch arbeitende Linguisten von den damals etablierten korpusbasierten Methoden ab und argumentierten, dass **Korpusdaten als empirische Evidenz** für linguistische Erkenntnis ungeeignet wären (Chomsky 1962, nach McEnery und Wilson 2001, S. 10). Zur Zeit erlebt die Verwendung von Korpora jedoch auch in der theoretischen Linguistik eine Renaissance (z. B. Bresnan et al. 2007). Es stehen dafür inzwischen leicht zugängliche Ressourcen zur Verfügung, die mit entsprechenden Such- und Analysewerkzeugen ausgewertet und visualisiert werden können (Baayen 2008; Johnson 2008; Gries 2008; Gries 2009).

4.1.4 Weiterführende Informationen

Weiterführende Informationen zu allen Themen dieses Unterkapitels bietet das Handbuch Corpus Linguistics (Lüdeling und Kytö 2008). Die Corpora Mailing List (<http://gandalf.aksis.uib.no/corpora/>) hilft bei allen Fragen rund um Textkorpora und liefert Informationen zu Konferenzen und Veröffentlichungen. Es empfiehlt sich, das umfangreiche Archiv der Liste zu konsultieren, bevor man eine eigene Frage an die Listengemeinschaft stellt. Eine umfassende Linkssammlung zu Korpora und Tools wird von David Lee gepflegt (<http://devoted.to/corpora>). Besonders an Computerlinguisten wendet sich die Sammlung der Stanford Natural Language Processing Group (<http://www-nlp.stanford.edu/links/statnlp.html>). Weitere Verweise auch auf deutschsprachige Seiten findet man auf der Linkssammlung des Lehrstuhls von Anke Lüdeling (<http://www.linguistik.hu-berlin.de/institut/professuren/korpuslinguistik/links/>). Die europäische Organisation Evaluations and Language Resources Distribution Agency (ELDA) veranstaltet alle zwei Jahre die International Conference on Language Resources and Evaluation (LREC). Die Special Interest Group for Annotation der Association for Computational Linguistics (ACL-SIGANN) führt in unregelmäßigen Abständen ebenfalls Workshops zum Thema durch. Zuletzt sei noch auf das Natural Language Toolkit verwiesen (<http://www.nltk.org>), ein Open Source-Projekt, das computerlinguistisch relevante Python-Module zusammenstellt. In das NLTK-Paket integriert ist eine Sammlung von Korpora mehrerer Sprachen, unter anderem Teile der syntaktisch annotierten englischen Penn Treebank.

4.2 Baumbanken

Sandra Kübler

Baumbanken als spezielle Form von Korpora (vgl. Unterkapitel 4.1) sind ein fester Bestandteil der Computerlinguistik, da sie detaillierte linguistische Information kodieren. Unter einer Baumbank wird eine Sammlung von Einheiten (meist Sätzen) verstanden, deren syntaktische Satzstruktur (vgl. Unterkapitel 3.5) annotiert ist. Die Bezeichnung Baumbank ist abgeleitet vom englischen *treebank*, der von Geoffrey Leech (vgl. Sampson 2003) geprägt wurde. Diese Bezeichnung verweist darauf, dass die Satzstruktur traditionell in Form einer Baumstruktur kodiert ist.

Das vorliegende Unterkapitel beschäftigt sich zunächst mit den zentralen Eigenschaften von Baumbanken (Abschnitt 4.2.1). Anschließend werden die wichtigsten Baumbanken (Abschnitt 4.2.2) und Suchprogramme für Baumbanken (Abschnitt 4.2.3) vorgestellt.

4.2.1 Zentrale Eigenschaften

Verwendung

Eine zentrale Verwendungsmöglichkeit für Baumbanken ist die empirische linguistische Forschung. In einer Zeit, in der sich die Linguistik darauf besinnt, dass Introspektion nur eine von mehreren Möglichkeiten ist, eine Datengrundlage für linguistische Forschung zu schaffen, gewinnen Korpora an Wert. Baumbanken sind vor allem dann wichtig, wenn reiner Text nicht ausreicht, die Suche auf das gewünschte Phänomen einzuschränken. Ein Beispiel wäre die Suche nach Koordinationen. Ist man nur an Koordinationen interessiert, die eine Konjunktion enthalten, so sind Beispiele dafür relativ einfach im Text zu finden. Ist man jedoch an Koordinationen interessiert, deren Konjunkte keine parallele Struktur aufweisen, muss man bei einem reinen Korpus alle Koordinationen durchsehen und die wenigen interessanten heraussuchen. Ein syntaktisch annotiertes Korpus erleichtert die Suche, da man explizit nach Koordinationskonstituenten mit ungleichen Konjunkten suchen kann. Wie einfach oder schwierig die Suche ist, hängt vor allem davon ab, wie gut das Annotationsschema der Baumbank das gesuchte Phänomen abdeckt (siehe Abschnitt 4.2.1).

Ein zweiter wichtiger Anwendungsbereich für Baumbanken findet sich in der Computerlinguistik selbst: Hier werden Baumbanken vor allem zum Training von statistischen Parsern verwendet. Statistische Parser extrahieren zwei unterschiedliche Arten von Information aus den Baumbanken: Zum einen werden die Grammatikregeln aus den Bäumen extrahiert, zum anderen Wahrscheinlichkeiten für diese Regeln. Während die Grammatikregeln auch manuell geschrieben werden können, ist es nahezu unmöglich, Wahrscheinlichkeiten ohne Baumbanken zu ermitteln. Des Weiteren können Baumbanken auch zur Evaluierung von symbolischen wie statistischen Parsern verwendet werden. Bei der Evaluierung von symbolischen Parsern muss jedoch beachtet werden, dass diese oft sehr kom-

plexen Grammatikmodelle verwenden (siehe Unterkapitel 3.5), die stark von der Baumbankannotation abweichen. In solchen Fällen kann es zu einer Benachteiligung bestimmter Parser kommen. Black et al. (1991) berichten, dass beim Vergleich der intendierten Ausgaben von verschiedenen Parsern für einen Satz nur eine einzige Klammerung bei allen Parsern übereinstimmte.

Ein weiterer Anwendungsbereich für Baumbanken hat sich erst in den letzten Jahren entwickelt: Baumbanken werden mit zunehmendem Erfolg im Unterricht eingesetzt, zum einen für praktische Anwendungen im Syntaxunterricht (siehe z. B. Hladka und Kucera 2008), zum anderen im Sprachenunterricht (siehe z. B. Bick 2005).

Erstellung

Die Erstellung einer Baumbank ist sehr zeit- und kostenintensiv. Aus diesem Grund muss zu Beginn eines solchen Annotationsprojekts entschieden werden, welche Herangehensweise sinnvoll und mit den vorhandenen Ressourcen machbar ist. Grundsätzlich ist es möglich, die gesamte Annotation mittels eines Parsers durchzuführen. Allerdings spricht man in diesem Fall eher von geparsten Korpora als von Baumbanken (siehe z. B. das BLLIP Korpus, das circa 30 Millionen Wörter des Wall Street Journals umfasst). Eine Baumbank wird i. A. manuell oder semi-automatisch erstellt und hat folglich einen kleineren Umfang als ein automatisch geparstes Korpus. Allerdings ist die manuelle Annotation zuverlässiger, da Parser eine Fehlerrate von 5%–25% aller Knoten haben, abhängig von Sprache und Annotationsschema. Semi-automatische Annotation bedeutet, dass die Annotation in einem interaktiven Prozess zwischen Annotator und Parser entsteht. Hierbei schlägt der Parser Annotationen vor, und die Annotatoren entscheiden zwischen möglichen Analysen und korrigieren Parserfehler. Beispiele für solche Systeme sind *Annotate* (Brants und Plaehn 2000, das aber nicht mehr gepflegt wird), *sympathy*¹ und *TreeBanker* (Carter 1997). Ein solches semi-automatisches Vorgehen hat nicht nur positive Auswirkungen auf die Annotationsgeschwindigkeit, sondern auch auf die Annotationskonsistenz, da der Parser für ein bestimmtes Phänomen konsistent dieselbe Annotation vorschlägt.

Annotationsschemata

Bei der Auswahl des Annotationsschemas für eine Baumbank müssen mehrere wichtige Entscheidungen getroffen werden: Konstituenz oder Dependenz, theorieneutral oder theoriespezifisch, Baumstrukturen oder Graphen, um nur die wichtigsten zu nennen. Diese Punkte werden im folgenden ausführlicher besprochen.

Bis Mitte der 90er Jahre wurden vor allem konstituenzbasierte Annotationsschemata verwendet. Hier werden die hauptsächlichen syntaktischen Kategorien wie NP oder VP in einer Baumstruktur annotiert. Abbildung 4.3 zeigt einen Baum aus der Penn Treebank (siehe Abschnitt 4.2.2). In diesem Baum, wie in allen folgenden in diesem Unterkapitel, werden drei Ebenen der Annotation

¹<http://www.mpi.nl/tools/sympathy.html>

unterschieden: 1. Tags auf Wortebene, die die Wortart und morphosyntaktische Information angeben (z. B. NNP für Eigenname, oder MD für Modalverb); 2. Knotenlabel für phrasale syntaktische Kategorien (z. B. NP); 3. Kantenlabel für grammatische Funktionen (z. B. SBJ für Subjekt oder TMP für Temporaladverbial).

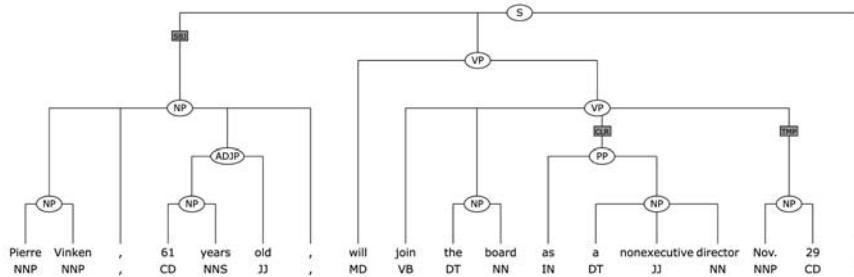


Abbildung 4.3: Der erste Satz der Penn Treebank

In neuerer Zeit dagegen nimmt die Dependenzannotation (Weber 1997) einen wichtigen Platz ein. Hier werden gerichtete Abhangigkeiten zwischen zwei Wortern annotiert. Abbildung 4.4 zeigt eine mogliche Dependenzannotation fur denselben Satz (siehe auch den Abschnitt uber die Prager Dependenzbaumbank in Abschnitt 4.2.2).

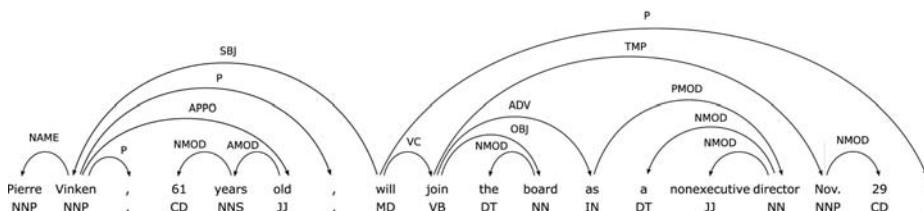


Abbildung 4.4: Dependenzannotation für den Satz aus Abbildung 4.3

Falls die Entscheidung für eine konstituenzorientierte Annotation gefallen ist, ist die nächste Frage, ob die Annotation theorieneutral oder theoriespezifisch, also z. B. basierend auf der Government and Binding-Theorie (Chomsky 1981) oder auf der HPSG (Pollard und Sag 1994), gewählt werden sollte. Eine theorieneutrale Annotation erhöht die Wiederverwertbarkeit einer Baumbank. Allerdings ist eine vollständige Theorieunabhängigkeit unmöglich, d. h. bei bestimmten syntaktischen Phänomenen widersprechen sich die einzelnen Theorien. Dann muss eine der alternativen Sichtweisen gewählt werden. Aus diesem Grund sprechen sich diverse Baumbankersteller (vgl. z. B. Simov und Osenova 2003) dafür aus, eine Theorie als Grundlage zu verwenden, weil damit die Konsistenz in der Annotation erhöht wird.

Eine weitere wichtige Frage bezieht sich auf das Datenmodell, bzw. darauf, wie nicht-lokale Phänomene beschrieben werden: Bei einer konstituenzorientierten Annotation besteht die Entscheidung grundsätzlich darin, ob eine reine Baumstruktur gewählt wird, in der diese Abhängigkeiten nur indirekt annotiert sind, oder ob kreuzende Kanten erlaubt werden, was zu einer allgemeineren Graphenstruktur führt. Während kreuzende Kanten vom linguistischen Standpunkt aus bevorzugt werden, können sie nicht ohne Probleme geparst werden. Diese Entscheidung wurde in den beiden deutschen Baumbanken, die in Abschnitt 4.2.2 beschrieben werden, unterschiedlich getroffen, daher wird hier für Beispiele auf diesen Abschnitt verwiesen.

Qualitätsmerkmale

Die grundlegenden Merkmale bei Baumbanken sind die selben wie bei Korpora (vgl. Unterkapitel 4.1). Zur verwendeten Textgrundlage ist allerdings zu bemerken, dass die meisten Baumbanken aus lizenzirechtlichen Gründen Zeitungstexte verwenden. Darüber hinaus sind für Baumbanken spezielle Qualitätskriterien relevant:

Wiederverwertbarkeit der Annotation. Wie schon erwähnt, ist die Annotation einer Baumbank ein zeit- und kostenintensives Unterfangen. Deswegen sollte das Annotationsschema der Baumbank so gewählt werden, dass sie für verschiedene Anwendungen und für längere Zeit verwendbar ist. Dies betrifft zum einen das Annotationsschema, zum anderen aber auch das Datenformat. Wie im vorhergehenden Abschnitt beschrieben, ist eine theorie neutrale Annotation vorzuziehen, solange gewährleistet werden kann, dass sie konsistent ausgeführt wird. Was das Datenformat anbelangt, gilt dasselbe wie auch für Korpora, dass die Annotation nicht in proprietären Datenformaten sondern als reiner Text dargestellt werden sollte. Eines der flexibelsten Formate für Baumbanken ist das TIGER-XML (Mengel und Lezius 2000), das im Rahmen des TIGER Projekts entwickelt wurde und das sich immer breiterer Anwendung erfreut.

Konsistenz der Annotation. Konsistenz in der Annotation ist für alle Anwendungsbereiche von überragender Bedeutung. Wenn die Baumbank für empirische Syntaxforschung verwendet wird, ist es wichtig, dass alle Vorkommen eines Phänomens gefunden werden. Dies kann nur geschehen, wenn sie alle gleich annotiert wurden. Ist dies nicht der Fall, werden entweder nicht alle Vorkommen gefunden, oder es werden Vorkommen gefunden, die nicht relevant sind. Wird die Baumbank zum Training von statistischen Parsern oder anderen Lernverfahren verwendet, ist Konsistenz ebenso wichtig, weil diese Verfahren Regelmäßigkeiten in den Daten finden. Sind Phänomene nicht konsistent annotiert, werden falsche Wahrscheinlichkeiten gelernt, die bei Anwendung des Parsers zu Parsingfehlern führen können.

Ein Mittel, um Konsistenz zu erreichen, ist ein ausführliches Annotations-Stylebook, in dem alle Entscheidungen vermerkt werden, so dass sie beim nächsten Auftreten des gleichen Phänomens nachgeschlagen werden können (siehe

nächster Absatz). Da die Annotation jedoch zumindest zum Teil manuell ausgeführt wird, ist ein gewisser Prozentsatz an Fehlern nicht zu vermeiden. Eine Möglichkeit, Konsistenz zu gewährleisten, besteht darin, jeden Satz von mindestens zwei Annotatoren unabhängig voneinander bearbeiten zu lassen. Danach werden die verschiedenen Annotations verglichen und Unterschiede bereinigt. Oft ist dieses Vorgehen aus organisatorischen oder finanziellen Gründen nicht möglich. Daher ist es wichtig, regelmäßig Konsistenzüberprüfungen durchzuführen. Diese können in Form von Skripten stattfinden, oder auch mit elaborierteren Methoden, wie sie von Dickinson und Meurers (2005a) entwickelt wurden (siehe auch Dickinson und Meurers 2005b, Boyd et al. 2008).

Dokumentation der Annotation. Die Dokumentation in einem Stylebook ist nicht nur in der Annotationsphase unerlässlich, sondern auch für die Wiederverwendbarkeit der Baumbank allgemein. Für die Suche nach einem bestimmten syntaktischen Phänomen muss erst bekannt sein, wie dieses Phänomen annotiert wurde. Soll z. B. nach Koordinationen ungleicher Konjunkte in der Penn Treebank gesucht werden, muss bekannt sein, dass dieses Phänomen mittels eines Knoten UCP annotiert wurde. Solche Informationen müssen im Stylebook dokumentiert sein. Beispiele für Stylebooks finden sich für die Penn Treebank (Bies et al. 1995) und die deutsche Baumbank TüBa-D/Z (Telljohann et al. 2006).

4.2.2 Die wichtigsten Baumbanken

In den letzten Jahren sind Baumbanken für eine Reihe von Sprachen entwickelt worden. Anstatt alle hier aufzuzählen, sei auf die Wikipedia Webseite für *treebank* verwiesen², dort wird eine aktuelle Liste der bestehenden Baumbanken geführt. In diesem Abschnitt werden daher nur die Penn Treebank als die meist verwendete Baumbank, die Prager Dependenzbaumbank als die einflussreichste dependenzannotierte Baumbank, und die zwei größten deutschen Baumbanken kurz vorgestellt. Für das Deutsche sind insgesamt vier Baumbanken verfügbar, allerdings basieren sie auf zwei Annotationsschemata, daher werden die kleineren Baumbanken nicht weiter beschrieben.

Die Penn Treebank. Die Penn Treebank (Marcus et al. 1994; Marcus et al. 1993) ist die am häufigsten verwendete Baumbank. Sie ist die Grundlage für Training und Testen einer ganzen Generation statistischer Parser, so dass Parsing schon als *Science of the Wall Street Journal*³ bezeichnet wurde (die Baumbank basiert vor allem auf Texten aus dem Wall Street Journal). Ihren Namen verdankt sie ihrem Entstehungsort, der University of Pennsylvania. Die Baumbank umfasst circa 50 000 Sätze, die mit Wortarten und mit syntaktischen Strukturen basierend auf der Theorie *Government and Binding* (Chomsky 1981) annotiert wurden. Die Penn Treebank gilt als Vorreiter für syntaktisch annotierte Korpora. Ihre breite Verwendung verdankt sie vor allem der einfachen (wenn auch

²URL: <http://en.wikipedia.org/wiki/Treebank>

³Stephan Oepen, Vortrag auf dem Workshop Unified Linguistic Annotation 2007, Bergen, Norwegen

nicht kostenlosen) Verfügbarkeit über das Linguistic Data Consortium (LDC).⁴ Wegen des großen Erfolgs der Baumbank wurde das Annotationsschema mit minimalen Änderungen auch für andere Sprachen verwendet, wie für die Penn Arabic Treebank (Bies und Maamouri 2003) oder die Penn Chinese Treebank (Xue et al. 2005).

Die Baumstrukturen wurden als Klammerstrukturen repräsentiert, die zur besseren Lesbarkeit um Einrückungen erweitert wurden (vgl. Abbildung 4.5).

```
( (S
  (SBAR-NOM-SBJ
    (WHNP-97 (WP What) )
    (S
      (NP-SBJ (PRP she) )
      (VP (VBD did)
        (NP (-NONE- *T*-97) ))))
    (VP (VBD was)
      (PP-PRD (IN like)
        (S-NOM
          (NP-SBJ (-NONE- *) )
          (VP (VBG taking)
            (NP (DT the) (NN law) )
            (PP-CLR (IN into)
              (NP (PP$ your) (JJ own) (NNS hands)))))))
      (. .) ))
```

Abbildung 4.5: Klammerstruktur-Darstellung des Satzes *What she did was like taking the law into your own hands.* aus der Penn Treebank

Aus Abbildung 4.5 wird deutlich, dass abgesehen von der reinen Konstituentenstruktur auch grammatische Funktionen (z. B. NP-SBJ für das Subjekt *she*) und Bewegungen annotiert werden. Die Bewegung des Frageworts *what* wird durch eine Spur (engl. *trace*) beschrieben, die die tatsächliche Position (WHNP) mit der gedachten Position (*T*) über eine Index-Nummer (97) koindiziert. Das fehlende Subjekt von *taking the law* wird durch ein leeres Element (*) ausgedrückt.

Die Prager Dependenzbaumbank. Diese Baumbank für Tschechisch wurde an der Karls-Universität in Prag entwickelt (Hajič et al. 2000). Die Annotation der Prager Dependenzbaumbank besteht aus drei Ebenen: der morphologischen, der analytischen und der tektogrammatischen Ebene. Sie hat einen Umfang von circa 2 Millionen Wörtern, von denen ca. 1,5 Mio. syntaktisch annotiert sind. Abbildung 4.6 zeigt die Annotation auf der analytischen Ebene für einen Beispielsatz. Obwohl die Annotation in Abbildung 4.6 stark an eine Baumstruktur erinnert, sind direkte Beziehungen zwischen Wörtern annotiert, wobei das übergeordnete Wort die höhere Position einnimmt, d. h. es besteht eine Dependenzbeziehung

⁴<http://www.ldc.upenn.edu>

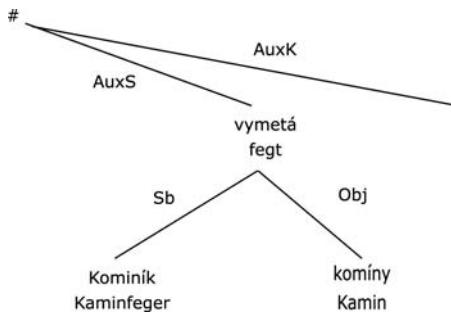


Abbildung 4.6: Der Satz *Kominik vymetá komíny.* aus der Prager Dependenzbaumbank

zwischen dem Verb *vymetá* und dem Nomen *komíny*. Wie in den Konstituentenstrukturen erhält diese Dependenz einen Typ, in diesem Fall *direktes Objekt Obj*. Die rechten und linken Zweige bilden die ursprüngliche Wortstellung im Satz ab, d. h. das Verb hat zwei abhängige Glieder, von denen das Subjekt links und das Objekt rechts vom Verb stehen.

Die analytische Ebene annotiert syntaktische Phänomene der Oberfläche. Tiefsyntaktische Phänomene werden auf der tektogrammatischen Ebene annotiert. So werden Präpositionen nur auf der analytischen Ebene als Knoten annotiert, auf der tektogrammatischen Ebene werden sie durch die grammatischen Relationen, die sie repräsentieren, ersetzt.

Das TIGER Korpus. Die erste deutsche Baumbank war die NeGra Baumbank (Skut et al. 1998), die im SFB-Projekt „Nebenläufige grammatische Verarbeitung“ in Saarbrücken annotiert wurde. Diese Baumbank mit circa 20 000 Sätzen ist der Vorläufer des TIGER Korpus (Brants et al. 2002). Beide Baumbanken basieren auf Texten aus der Zeitung „Frankfurter Rundschau“. TIGER verwendet das NeGra Annotationsschema mit geringen Änderungen weiter. In Version 2.1 umfasst das TIGER Korpus circa 50 000 Sätze. Abbildung 4.7 zeigt einen Beispielsatz aus dem TIGER Korpus.

Die syntaktische Annotation besteht aus drei Ebenen: Wortarten, syntaktische Kategorien und grammatische Funktionen. Für die Wortartenannotation wurde das *Stuttgart-Tübingen Tagset* (STTS; Schiller et al. 1999) verwendet, das sich zum Standard für die morphosyntaktische Annotation des Deutschen entwickelt hat. Die Wortartenannotation wird durch eine morphologische Annotation ergänzt (die in Abbildung 4.7 nicht dargestellt ist). Im Vergleich zum Baum aus der Penn Treebank fallen vor allem zwei Unterschiede auf: Der erste Unterschied betrifft die grammatischen Funktionen. Durch die wesentlich freiere Wortstellung im Deutschen ist es nicht möglich, die grammatische Funktion einer Nominalphrase aus der Konfiguration des Satzes zu bestimmen. Aus diesem Grund werden grammatische Funktionen auf allen Ebenen annotiert. Innerhalb der flach annotierten Präpositionalphrasen z. B. dienen die grammatischen

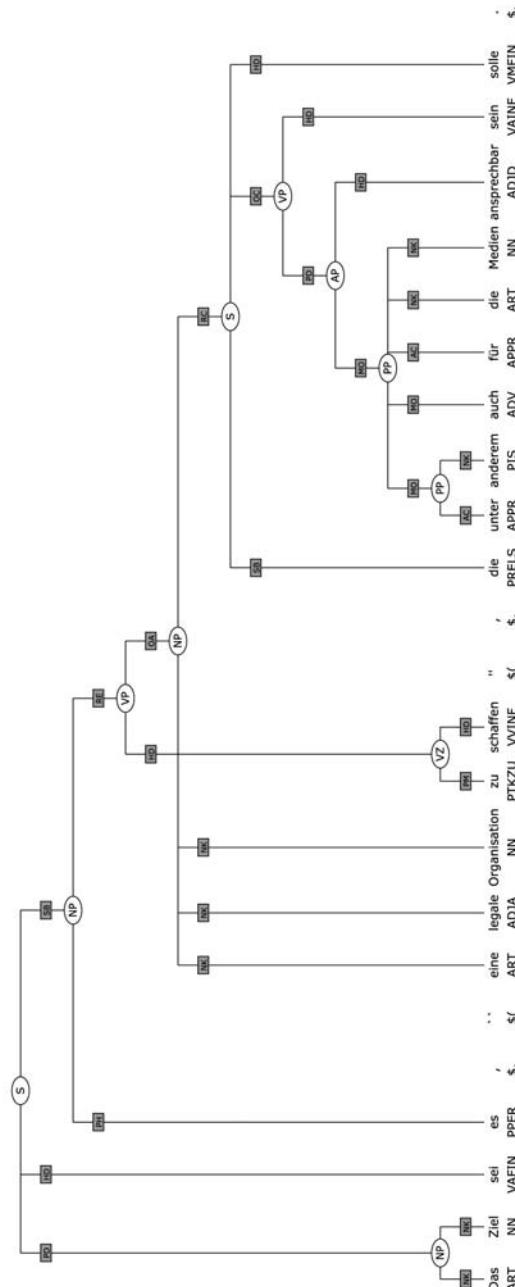


Abbildung 4.7: Ein Satz aus dem TIGER Korpus

Funktionen dazu, den Nominalkern (grammatische Funktion NK) der Phrase zu bestimmen.

Der zweite Unterschied zur Penn Treebank besteht darin, dass aus linguistischen Gründen darauf verzichtet wurde, eine reine Baumstruktur zu annotieren. Die freiere Wortstellung im Deutschen erlaubt die Extraposition verschiedener Konstituenten. Statt der Spurenannotation aus der Penn Treebank wird im TIGER Korpus die Anbindung an die Mutterkonstituente direkt vorgenommen. Dies führt zu kreuzenden Kanten. In Abbildung 4.7 liegt ein extraponierter Relativsatz vor (grammatische Funktion RC), der über eine kreuzende Kante mit der Nominalphrase *eine legale Organisation* gruppiert ist.

Es fällt weiterhin auf, dass die phrasale Annotation sehr flach ist: Nominalphrasen innerhalb von Präpositionalphrasen werden nicht gruppiert, sie sind nur über die bereits erwähnten grammatischen Funktionen zu erschließen. Außerdem werden keine unäre Knoten annotiert. Das bedeutet, dass alle Phrasen mit nur einer Tochter nicht als Knoten im Baum repräsentiert sind. In Abbildung 4.7 wird das Personalpronomen *es* nicht zur Nominalphrase projiziert, bevor es auf der Satzebene angebunden wird.

Die Tübinger Baumbank des Deutschen / Schriftsprache, TüBa-D/Z. Dies ist die zweite große Baumbank des Deutschen. Sie basiert auf der Berliner Zeitung „die tageszeitung“ (taz). In Version 4 umfasst sie circa 36 000 Sätze. Abbildung 4.8 zeigt einen Beispielsatz aus der TüBa-D/Z.

Die syntaktische Annotation besteht aus den selben Ebenen, die auch in der TIGER Baumbank auftreten. Auf der morphosyntaktischen Ebene wird ebenfalls das STTS verwendet. Die anderen Ebenen unterscheiden sich allerdings deutlich. Der erste Unterschied besteht darin, dass in der TüBa-D/Z eine zusätzliche Ebene annotiert wird: topologische Felder. Diese sind ein allgemein verwendetes Mittel, um die grundlegende Wortstellung im Deutschen zu charakterisieren (Drach 1937; Höhle 1986). Die topologischen Felder sind direkt unter der Satzebene eingegliedert. In dem Satz in Abbildung 4.8 teilen sie den Matrixsatz in das *Vorfeld VF*, die *linke Satzklammer LK*, die das finite Verb enthält, das *Mittelfeld MF*, den *Verbkomplex VC* und das *Nachfeld NF* mit dem extraponierten Relativsatz *R-SIMPX*.

Ein weiterer Unterschied zum TIGER Annotationsschema besteht darin, dass TüBa-D/Z keine kreuzenden Kanten verwendet. Stattdessen werden die extraponierten Konstituenten an Ort und Stelle angehängt; über das Label der grammatischen Funktion wird dann darauf verwiesen, zu welcher Konstituente sie gehört. In dem Satz in Abbildung 4.8 hat der Relativsatz die grammatische Funktion *OA-MOD*, die kennzeichnet, dass der Relativsatz das Akkusativobjekt *OA* modifiziert.

Es fällt außerdem auf, dass TüBa-/Z mehr Struktur innerhalb der Phrasen annotiert. Zum einen werden unäre Knoten annotiert (vgl. z. B. das Relativpronomen *die*, das erst zur Nominalphrase *NX* und dann zum Complementizerfeld *C* projiziert wird). Zum anderen werden Nominalphrasen innerhalb von Präpositionalphrasen annotiert (vgl. die Phrase *mit den Probenbesuchern*), und postmodifizierte Phrasen werden erst gruppiert, bevor die Postmodifikation angebunden

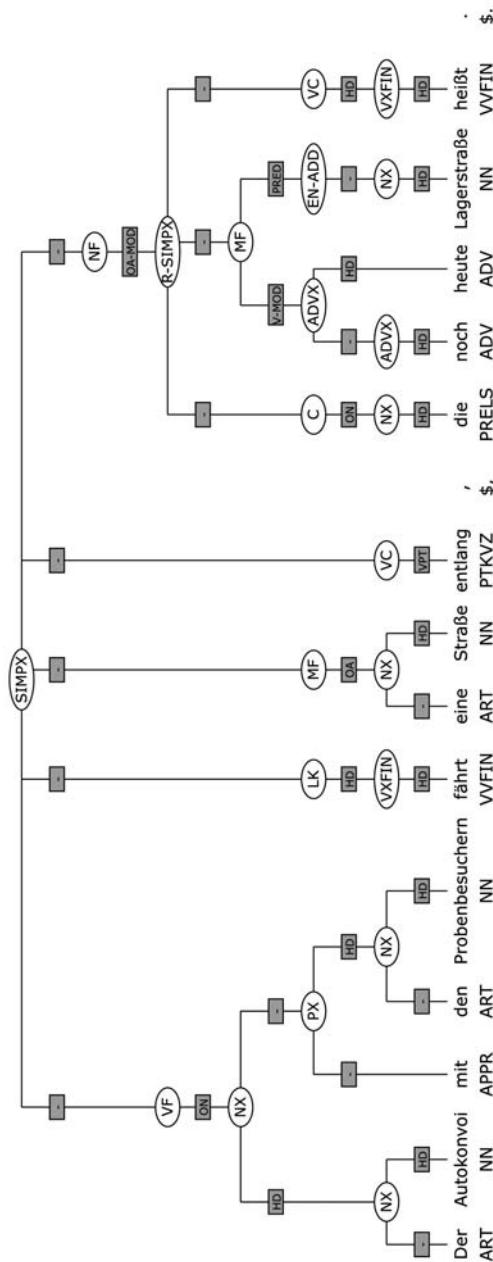


Abbildung 4.8: Ein Satz aus der TüBa-D/Z

wird (vgl. die komplexe Nominalphrase *Der Autokonvoi mit den Probenbesuchern*).

Ein weiterer Unterschied, der hier nicht weiter beschrieben werden soll, besteht in der Auswahl der syntaktischen Kategorien und grammatischen Funktionen. Dies wird z. B. darin deutlich, dass der Relativsatz in TIGER durch eine grammatische Funktion **RC** als solcher gekennzeichnet wird, während TüBa-D/Z dies direkt in der syntaktischen Kategorie **R-SIMPX** kennzeichnet. Die Anbindung des Relativsatzes erfolgt in TIGER direkt, TüBa-D/Z verwendet dafür die grammatische Funktion.

TüBa-D/Z verwendet ein Annotationsschema, das mit geringen Abweichungen schon vorher für die Annotation der *Tübinger Baumbank des Deutschen / Spontansprache* (TüBa-D/S; Stegmann et al. 2000) verwendet worden war. Letztere Baumbank wurden in Tübingen im Rahmen des Projekts VerbMobil (Wahlster 2000) annotiert. Sie umfasst Transliterationen spontansprachlicher Dialoge aus der Domäne Terminvereinbarungen, in einem Umfang von circa 38 000 Äußerungen. Zusammen mit den VerbMobil Baumbanken für Englisch und Japanisch und den Switchboard-Daten in der Penn Treebank ist dies eine der wenigen Baumbanken für gesprochene Sprache.

4.2.3 Suche in Baumbanken

Die Suche nach syntaktischen Phänomenen ist wesentlich komplexer als die Suche nach Sequenzen von Wörtern oder wortbasierten Annotationen, wie Wortarten, da hier die Beziehungen zwischen Knoten wichtig sind, bzw. u. U. nach Sätzen gesucht wird, die bestimmte Elemente *nicht* enthalten. Ein Beispiel für letzteres wäre die Suche nach subjektlosen Sätzen wie *Mir ist übel*.

Die Komplexität einer solchen Suche soll hier kurz anhand der Suche nach extrapolierten Relativsätzen illustriert werden. Hierzu muss zuerst bekannt sein, wie dieses Phänomen in der Baumbank annotiert ist, d. h. diese Suchanfrage sieht für die TIGER Baumbank anders aus als für TüBa-D/Z. In TIGER muss nach einem Satzknosten **S** gesucht werden, der von einer Nominalphrase **NP** abhängt und die grammatische Funktion **RC** hat, der aber nicht adjazent zum Kopf der **NP** ist. Hieraus wird deutlich, dass das Suchwerkzeug zumindest Dominanz- und Adjazenzrelationen zwischen allen Knoten abbilden muss.

Es gibt mehrere Suchtools, die Suchen nach syntaktischen Phänomenen erlauben: Neben *tgrep*, einem Suchtool, das mit der Penn Treebank mitgeliefert wird, existieren verschiedene Derivate wie *tgrep2* (Rohde 2001) oder *tregex* (Levy und Andrew 2006). Das bekannteste und flexibelste Tool ist jedoch TIGERSearch (König und Lezius 2000). TIGERSearch erlaubt die Suche in zwei verschiedenen Formaten: mit einer formalen Abfragesprache oder mittels einer graphischen Oberfläche, in der die Suchabfrage als partieller Baum dargestellt wird. Letzteres Vorgehen erlaubt die Benutzung des Tools, ohne dass die Abfragesprache beherrscht werden muss. Dies befreit den Suchenden jedoch nicht von der Pflicht, sich mit dem Annotationsschema der Baumbank vertraut zu machen.

4.2.4 Literaturhinweise

Eine Übersicht über englischsprachige Baumbanken und deren grundlegenden Prinzipien wird in Leech und Eyes (1997) gegeben, ein internationalerer Überblick findet sich in Abeillé (2003). Weitere Ansatzpunkte für relevante Literatur bieten die zahlreichen Workshops zu Baumbanken, die sich in den letzten Jahren etabliert haben: Frontiers in Linguistically Annotated Corpora (FLAC), Linguistic Annotation (LAW), Linguistically Interpreted Corpora (LINC) und Treebanks and Linguistic Theories (TLT).

4.3 Lexikalisch-semantische Ressourcen

Claudia Kunze

4.3.1 Lexikalisch-semantische Wortnetze

In diesem Kapitel werden lexikalisch-semantische Wortnetze im Stile des Princeton WordNet (Miller et al. 1990; Fellbaum 1998) als eine besondere Spielart elektronischer Ressourcen vorgestellt. Solche Wortnetze bilden die häufigsten und wichtigsten Wörter einer Sprache und ihre bedeutungstragenden Beziehungen zu anderen Wörtern der Sprache ab. Im Wortnetz ist ein Wort als Konzeptknoten mit seinen semantischen Verknüpfungen repräsentiert: z. B. *Stuhl* mit dem Oberbegriff *Sitzmöbel* und seinen Unterbegriffen *Drehstuhl*, *Klappstuhl*, *Kinderstuhl* etc. Der Oberbegriff ist darüberhinaus mit den Konzepten *Lehne*, *Sitzfläche* und *Bein* verbunden, die Teile eines Sitzmöbels repräsentieren.

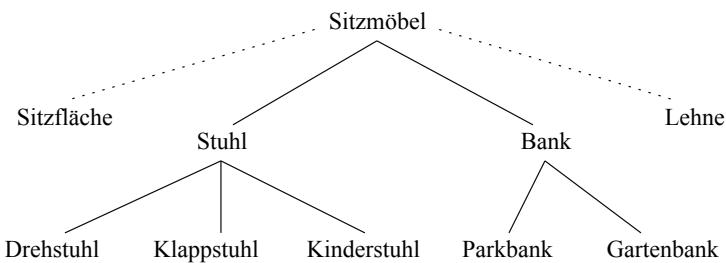


Abbildung 4.9: Semantisches Netz für *Sitzmöbel*

Ein Konzept ist also nicht nur über seinen Knoten, sondern auch über seine Relationen zugreifbar. Da die zugrunde liegende Repräsentationseinheit, das so genannte **Synset**, gleiche Bedeutungen, die Synonyme, zu einem Konzeptknoten zusammenfasst und nicht etwa gleiche Wörter, werden in Wortnetzen Lesarten unterschieden. Diese Lesartendisambiguierung ist eine unabdingbare Voraussetzung für Anwendungen im Bereich der Maschinellen Übersetzung und der Informationserschließung, für die semantische Annotierung von Sprachkorpora und für die Entwicklung verschiedener Werkzeuge zum Sprach- und Informationserwerb und für die Übersetzung. Der folgende Abschnitt 4.3.1 beschreibt detailliert das lexikalisch-semantische Wortnetz GermaNet, das den deutschen Grundwortschatz abdeckt (Kunze und Wagner 1999). Abschnitt 4.3.1 gibt Aufschluss über die Einbindung des deutschsprachigen GermaNet in das polylinguale EuroWordNet, das im Rahmen eines europäischen Projektes 1996–1999 für acht Sprachen aufgebaut wurde (Vossen 1999). Abschließend werden in Abschnitt 4.3.1 exemplarisch zwei verschiedene computerlinguistische Anwendungsszenarien für semantische Wortnetze skizziert.

GermaNet – ein deutsches Wortnetz

Mit GermaNet ist ein computertechnisch verfügbares semantisches Lexikon aufgebaut und ein wichtiger Beitrag zur wissensbasierten Ressourcenbildung für das Deutsche geleistet worden (<http://www.sfs.uni-tuebingen.de/GermaNet>). Im Wesentlichen orientiert sich das deutschsprachige Wortnetz am Datenbankformat und an den Strukturierungsprinzipien des Princeton WordNet 1.5, das als „Mutter aller Netze“ eine initiale Rolle für viele einzelsprachliche Initiativen spielte.⁵ GermaNet ist jedoch keine pure Übersetzung des WordNet, sondern setzt eigene Schwerpunkte in der Konzeptrepräsentation (vgl. Hamp und Feldweg 1997).

Abdeckung und Ressourcen GermaNet ist aus verschiedenen lexikografischen Quellen, z. B. dem DEUTSCHEN WORTSCHATZ und dem BROCKHAUS/WAHRIG, unter der Berücksichtigung von Korpusfrequenzen von Hand aufgebaut worden. In GermaNet sind bislang die bedeutungstragenden Kategorien Nomen, Verben und Adjektive modelliert. Zentrales Repräsentationskonzept ist das Synset, welches die Synonymenmenge eines gegebenen Konzeptes bereitstellt, z. B. *{Streichholz, Zündholz}*, *{fleißig, eifrig, emsig, tüchtig}* und *{vergeben, verzeihen}*. Im Wortnetz sind semantische Relationen zwischen den Konzepten (Synsets) oder einzelnen Varianten (Synonymen aus den Synsets) kodiert. Zurzeit enthält GermaNet knapp 58 000 Synsets mit ca. 82 000 Varianten, davon ca. 43 100 Nomen, 9 400 Verben und 5 500 Adjektive. Das deutsche Wortnetz wird durch den Abgleich der Datenbankeinträge mit Frequenzlisten aus Korpora systematisch um fehlende Konzepte ergänzt. GermaNet repräsentiert nur wenige Mehrwortlexeme wie *gesprochene Sprache* oder *Neues Testament*. Eigennamen treten hauptsächlich im Wortfeld der Geografie auf, z. B. als Städtenamen, und werden speziell markiert.

Relationstypen Die Aussagekraft semantischer Netze liegt in den zahlreichen sinnhaften Verknüpfungen zwischen den repräsentierten Knoten. GermaNet unterscheidet zwischen **lexikalischen** und **konzeptuellen Relationen**:

- Lexikalische Relationen sind bidirektionale Beziehungen zwischen Wortbedeutungen wie die synset-interne **Synonymie** (Bedeutungsgleichheit zwischen *Ruf* und *Leumund*) und die **Antonymie** (Gegenteiligkeit), etwa zwischen *Geburt* und *Tod*, *glauben* und *zweifeln*, *schön* und *hässlich*.
- Konzeptuelle Relationen wie **Hyponymie**, **Hyperonymie**, **Meronymie**, **Implikation** und **Kausation** bestehen zwischen Konzepten, gelten also für alle Realisierungen innerhalb eines Synsets. Sowohl Hyponymie und

⁵Das Urmodell semantischer Netze entwickelte Quillian (1968) zur Modellierung des semantischen Gedächtnisses anhand von semantischen Verknüpfungen zwischen Lexikoneinträgen und Definitionswörtern. Prinzipiell sind lexikalisch-semantische Wortnetze aber von den semantischen Netzen zur Wissensrepräsentation aus der Informatik und KI (vgl. Helbig und Gnörlisch 2002) abzugrenzen.

Hyperonymie als auch Meronymie und Holonymie sind **konverse Relatonspaare**: so ist *Dach* Meronym zu *Gebäude* und *Gebäude* das Holonym von *Dach*.

Das wichtigste Strukturierungsprinzip in semantischen Netzen stellt die hierarchiebildende **Hyponymierelation**, wie sie zwischen *Rotkehlchen* und *Vogel* besteht, dar. Besonders die Nomina haben Ketten mit tiefen Hierarchien, wie z. B. das Konzept *Kieferchirurg* mit 15 Dominanzstufen. In GermaNet sind auch die Verben und Adjektive hyponymisch gegliedert.

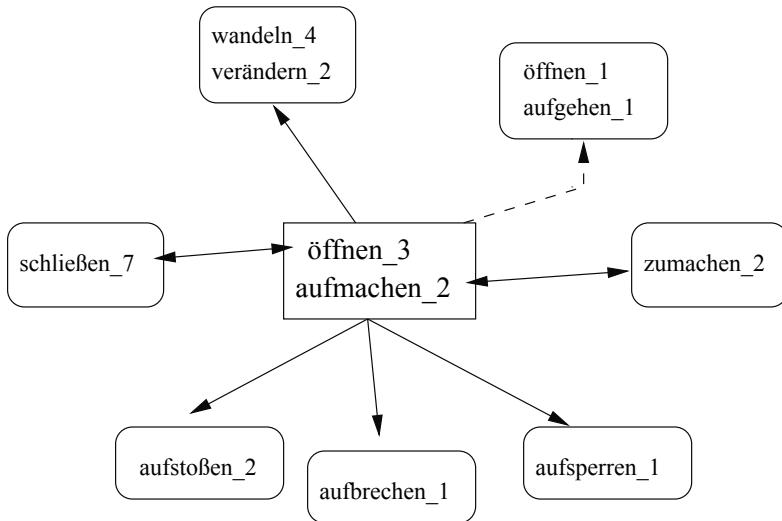


Abbildung 4.10: Semantisches Netz für das Verb *öffnen*

Die **Meronymierelation** (Teil-Ganzes-Beziehung) wird nur für Nomina angenommen: Ein *Dach* kann nicht angemessen als eine Art *Gebäude* klassifiziert werden, sondern ist *Teil* eines *Gebäudes*. Teil-Ganzes-Beziehungen können auch abstrakter Natur sein, z. B. in bezug auf die Mitgliedschaft in einer Gruppe (*Vorsitzender* einer *Partei*) oder als Material in einer Komposition (*Fensterscheibe* aus *Glas*).

Typischerweise wird die Verknüpfung zwischen lexikalischen Resultativen wie *töten* und *sterben* oder *öffnen* und *offen* als **Kausationsrelation** spezifiziert. Die kausale Relation kann klassenübergreifend zwischen allen Kategorien kodiert werden.

Seltener hingegen wird von der **Implikationsbeziehung** oder dem **Entailment** Gebrauch gemacht, wie etwa zwischen *gelingen* und *versuchen*.

Neuerdings werden auch syntagmatische Beziehungen, genauer gesagt einschlägige Kollokationsbeziehungen (Subjekt-Verb- und Objekt-Verb-Paare) systematisch aus Korpora extrahiert und in GermaNet kodiert, vgl. Lemnitzer et al. (2008).

Die Bedeutung eines Wortes ist durch die Gesamtheit der Relationen, die sie zu anderen Wortbedeutungen aufweist, gekennzeichnet.⁶ Abbildung 4.10 zeigt das kausative Verb *öffnen* mit allen semantisch korrelierten Konzepten. Synsets und Varianten sind mit den entsprechenden Lesartennummern aus GermaNet aufgeführt.

Die Verbindung des Synset *{öffnen_3, aufmachen_2}* mit seinem Hyperonym *{wandeln_4, verändern_2}* wird durch den nach oben weisenden Pfeil repräsentiert, mit den drei Hyponymen *{aufstoßen_2}*, *{aufbrechen_2}* und *{aufsprennen_1}* durch jeweils abwärts gerichtete Pfeilspitzen, und die kausale Relation zum intransitiven Konzept *{öffnen_1, aufgehen_1}* durch den Pfeil mit gestrichelter Linienführung. Die beiden Varianten im Synset haben unterschiedliche Antonyme: *öffnen_3* hat als Antonym *schließen_7*, und *aufmachen_2* das Antonym *zumachen_2*. Die Antonymierelation ist durch den Doppelpfeil gekennzeichnet.

Kreuzklassifikation und künstliche Konzepte Ein Konzept wie *Banane* kann ebenso wie eine Reihe weiterer Früchte gleichermaßen als *Pflanze* und als *Nahrungsmittel* klassifiziert und somit unterschiedlichen semantischen Feldern zugeordnet werden. Um diese Information zugreifbar zu machen, empfiehlt sich die **Kreuzklassifikation** solcher Konzepte in verschiedenen Hierarchien.

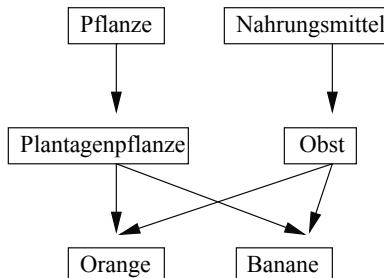


Abbildung 4.11: Kreuzklassifikation in GermaNet

Wortnetze sollen nur tatsächlich vorkommende Lexeme einer Sprache abbilden. In GermaNet wird jedoch Gebrauch von künstlichen Konzepten gemacht, wenn diese geeignet sind, die Hierarchie besser zu strukturieren und unmotivierte Ko-Hyponymie zu vermeiden. Nach Cruse (1986) (S. 88ff.) sollten Ko-Hyponyme auf einer Basis von Ähnlichkeit, die durch den gemeinsamen Mutterknoten gegeben ist, möglichst inkompaktil zueinander sein, vgl. *Säugling*, *Kleinkind*, *Vorschulkind*, *Schulkind* als Unterbegriffe zu *Kind*, die einander wechselseitig ausschließen. Im Wortfeld Lehrer sind Unterbegriffe wie *Fachlehrer*, *Berufsschullehrer* und *Konrektor* nicht sinnvoll auf einer gemeinsamen Hierarchieebene anzunehmen.

⁶Es gibt in GermaNet über die ausführlich beschriebenen Relationen hinausgehend noch die Pertonymie (eine Art semantischer Derivationsbeziehung wie z. B. zwischen *finanziell* und *Finanzen*) und eine Ähnlichkeitsrelation (*see also*), die assoziativen Verknüpfungen Rechnung trägt wie zwischen *Weltrangliste* und *Tennis* oder *Talmud* und *Judentum*.

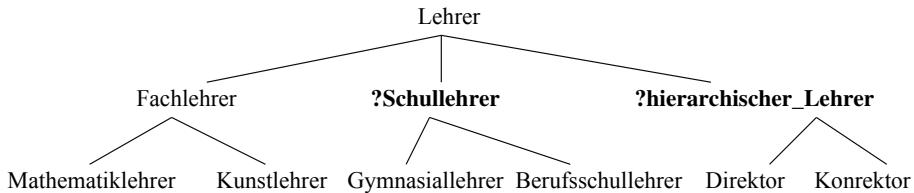


Abbildung 4.12: Künstliche Konzepte im Wortfeld ‚Lehrer‘

siedeln. Um das Teilnetz symmetrischer zu gestalten, werden mit *?Schullehrer* und *?hierarchischer Lehrer* zwei künstliche Konzepte eingeführt.

EuroWordNet, ein polylinguales Wortnetz

Das Basisvokabular des GermaNet, etwa 15 000 Synsets, ist in das polylinguale EuroWordNet (<http://www.illc.uva.nl/EuroWordNet/>) für acht europäische Sprachen integriert worden, vgl. Wagner und Kunze (1999). EuroWordNet modelliert die wichtigsten Konzepte des Englischen, Spanischen, Holländischen, Italienischen, Französischen, Deutschen, Tschechischen und Estnischen in ihren semantischen Relationen (vgl. Vossen 1999). Kernkomponente der Datenbankarchitektur ist der **Interlinguale Index (ILI)**, an den die einselsprachlichen Wortnetze geknüpft sind. Der ILI fungiert als sprachunabhängige Komponente und besteht aus einer unstrukturierten Liste von ILI-Records, die an WordNet Synsets (und somit englischen Lesarten) orientiert sind und durch einen eindeutigen Code („unique identifier“), gekennzeichnet sind. Konzepte der einzelnen Sprachen werden mit sprachübergreifenden Relationen an passende Übersetzungäquivalente aus dem ILI geknüpft. Über den ILI können dann mittelbar spezifische Sprachpaare zu erfragten Konzepten gebildet werden, z. B. *guidare:conducir* (Italienisch:Spanisch) für das Konzept *drive* in Abbildung 4.13. Zu den sprachunabhängigen Komponenten zählen neben dem ILI die **Top Ontologie** mit 63 semantischen Merkmalen und die **Domänen Ontologie**, die semantische Felder zur Verfügung stellt wie etwa ‚food‘ oder ‚traffic‘.

Alle einselsprachlichen Wortnetze enthalten eine gemeinsame Menge so genannter **Base Concepts**, 1 000 Nomen und 300 Verben, die als zentrales Vokabular des polylingualen Wortnetzaufbaus fungieren und die Kompatibilität der einzelnen Sprachnetze gewährleisten. Base Concepts weisen die folgenden Eigenschaften auf:

- Sie dominieren viele Knoten und/oder eine hierarchisch vielstufige Kette von Unterbegriffen;
- Sie sind häufig auftretende Konzepte zweier oder mehrerer Sprachen;

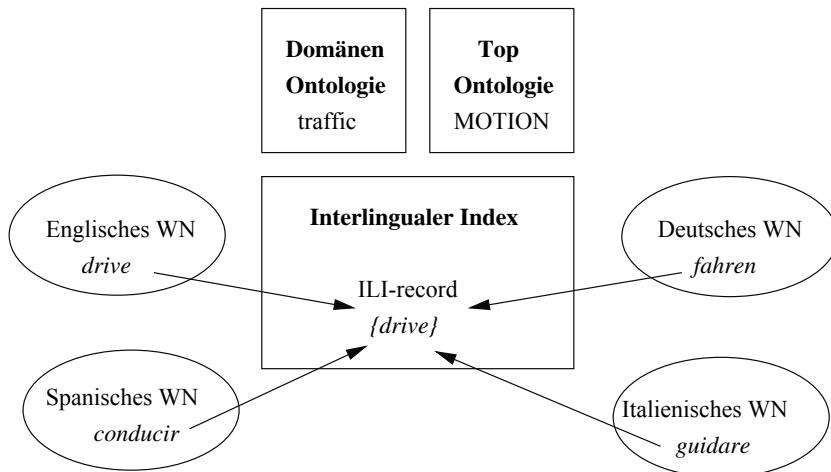


Abbildung 4.13: EuroWordNet-Architektur

- Sie sollen konkreter als die semantischen Merkmale der Top Ontologie wie DYNAMIC, FUNCTION und PROPERTY sein, aber auch abstrakter als die von Rosch (siehe Rosch 1978) postulierten **Basic Level Concepts**, z. B. *Tisch* und *Hammer*. Der angemessene Abstraktionsgrad für Base Concepts wird von den jeweiligen Oberbegriffen der Basic Level Concepts, z. B. *Möbel* für *Tisch* und *Werkzeug* für *Hammer* erreicht.
- Base Concepts werden durch Merkmale oder Merkmalskombinationen aus der Top Ontologie charakterisiert, z. B. *Werkzeug* durch ARTEFACT, INSTRUMENT, OBJECT.

Nachdem das Inventar der Base Concepts mit dem ILI verknüpft war, konnten Top-Konzepte und Hyponyme erster Ordnung gelinkt werden, was zu einem ersten Datenensemble von ca. 7 500 Synsets führte. Der Aufbau einzelsprachlicher Netze konnte dann unabhängig erfolgen, zumal die Vererbung der semantischen Merkmale der Top Ontologie ermöglicht, die Abdeckung der Netze in einzelnen semantischen Feldern statistisch zu untersuchen.

Aufgrund unterschiedlicher Lexikalisierungsmuster der einzelnen Sprachen, die auf sprachliche und kulturelle Unterschiede zurückgehen und aufgrund von Kodierungslücken im WordNet (das ja die Basisressource für den ILI darstellt), können nicht immer angemessene Übersetzungen der einzelsprachlichen Konzepte gefunden werden. Daher sind auch nicht-synonymische sprachübergreifende Verknüpfungen sowie die Kombination mehrerer nicht-synonymischer Links möglich. Z. B. ist für das Konzept *Sportbekleidung* kein synonymisches Targetkonzept *sports garment* im ILI verfügbar. Ersatzweise können zwei sprachübergreifende Links zum Hyperonym *garment* (*Kleidung*) und zum Holonym *sports equipment* (*Sportausrüstung*) etabliert werden.

Die internationale Zusammenarbeit im EuroWordNet-Projekt hat mit dem ILI als grundlegendem Lesarten-Inventar einen Quasi-Standard für den Aufbau von Wortnetzen entwickelt. Dieser liegt auch dem BalkaNet-Projekt (vgl. Pala et al. 2002) für die Integrierung osteuropäischer Sprachen zugrunde. Zur Entwicklung von Richtlinien und Standards für Wortnetze ist die ‚Global Wordnet Association‘ (GWA) gegründet worden (<http://www.globalwordnet.org/>). Eine darüber hinaus gehende zukünftige Forschungsperspektive liegt in der Adaption von Wortnetzen für Ontologien (siehe Unterkapitel 4.6) und das **Semantic Web**, vgl. Lemnitzer und Kunze (2002).

Anwendungsperspektiven semantischer Netze in der CL

Lesartendisambiguierung In der CL kommt der **Lesartendisambiguierung** („Word Sense Disambiguation“) eine zentrale Rolle zu. Zur Erfassung der Bedeutung eines Satzes oder Kontextes müssen darin auftretende mehrdeutige Wörter eindeutig gemacht werden. Wortnetze liefern mit den Synsets bereits semantisch disambiguerte Einheiten, da mehrdeutige Wörter ihren Lesarten entsprechend zu mehreren synsets gehören (vgl. hierzu Abschnitt 3.6.5). Ein Wort wie *Ton* ist in den Synsets {*Ton*, *Laut*} und {*Ton*, *Tonerde*} enthalten und weist somit zwei Lesarten auf. Aber auch die semantischen Beziehungen zwischen den Konzepten sind aufschlussreich für die Disambiguierung. So können die (hyponymisch ermittelten) Hierarchien in Wortnetzen zur Ermittlung der semantischen Ähnlichkeit von Konzepten ausgenutzt werden, etwa zur semantischen Disambiguierung von bedeutungstragenden Wörtern in syntaktisch analysierten Texten (vgl. Stetina et al. 1998).

Mit SENSEVAL wurde ab 1998 ein internationales Programm zur Evaluierung von Lesartenunterscheidungen aufgelegt, welches Disambiguierungssysteme sowie den Übereinstimmungsgrad menschlicher Annotierer testet (vgl. Kilgarriff und Edmonds 2003).

Informationserschließung Für die Informationserschließung („Information Retrieval“) ist ein semantisches Netz von großem Nutzen. Aus einem umfangreichen Inventar an Texten sind hier diejenigen Dokumente zu ermitteln, welche die angefragte Information enthalten. Sind sowohl die Anfrage als auch die zu durchsuchenden Dokumente disambiguierbar, so kann gezielt nach Vorkommen der gewünschten Lesart gesucht werden, etwa nach dem Stichwort *Bank* in der Lesart *Geldinstitut* und nicht als *Sitzmöbel*, was die Treffergenauigkeit („precision“) steigert. Darüberhinaus kann mit Hilfe eines semantischen Netzes die Anfrage mit semantisch korrelierten Konzepten des Suchbegriffs erweitert werden. So würden dann bei der Anfrage nach *Bank* auch solche Dokumente ermittelt, die den Suchbegriff zwar nicht enthalten, aber seine semantisch korrelierten Begriffe *Geldinstitut* oder *Sparkasse*. Die Anzahl der korrekt ermittelten Dokumente („recall“) wird dadurch erhöht. Mit EuroWordNet ist auch polylinguale Informationserschließung in Texten unterschiedlicher Sprachen möglich, und zwar durch die Expansion der Suchbegriffe mit ihnen äquivalenten Konzepten aus anderen Sprachen. Die Ergebnisse in der Informationserschließung verbessern sich deut-

label	meaning
HEALER	individual who tries to bring about an improvement in the PATIENT
PATIENT	individual whose physical wellbeing is low
DISEASE	sickness or health condition that needs to be removed or relieved
WOUND	tissue damage in the body of the PATIENT
BODYPART	limb, organ etc. affected by the DISEASE OR WOUND
SYMPTOM	evidence indicating the presence of the DISEASE
TREATMENT	process aimed at bringing about recovery
MEDICINE	substance applied or ingested in order to bring about recovery

Tabelle 4.1: Frame zum Thema HEAL, nach Lowe et al. (1997)

lich, wenn sowohl die Anfrage als auch die Dokumente mit WordNet Synsets indiziert sind, wie Gonzalo et al. (1999) durch Experimente belegen. Kategorisierungsverfahren im Bereich der Textklassifikation und der Textzusammenfassung (vgl. Unterkapitel 5.3) profitieren ebenfalls von bereit gestellten Synonymen und semantisch verknüpften Konzepten der Synsets.

4.3.2 FrameNet

FrameNet wird am ICSI in Berkeley seit 1997 (vgl. Baker et al. 2003, Ruppenhofer et al. 2006) als elektronische Ressource nach den Prinzipien der Fillmoreschen Frame-Semantik entwickelt (Fillmore 1968, Fillmore und Atkins 1992). Während Wortnetze sprachliches Wissen kodieren und unmittelbar die semantischen Beziehungen zwischen lexikalischen Einheiten abbilden, modelliert FrameNet Konzepte, auf die sprachspezifische Lexikalisierungen bezogen werden. Ein **semantic frame** stellt eine konzeptuelle Struktur zur Beschreibung eines spezifischen Situationstyps, Objekttyps oder Ereignistyps mit seinen jeweilig verknüpften Partizipanten und Eigenschaften dar. Der Frame HEAL enthält z. B. als Rollen die Frame-Elemente (FE) HEALER, PATIENT, DISEASE, WOUND, BODYPART, SYMPTON, TREATMENT und MEDICINE (vgl. Abb. 4.1 unter LABEL) und wird durch Begriffe wie *Arzt*, *Therapeutin*, *Kranker*, *Unfallopfer*, *Verletzung*, *Platzwunde*, *Oberschenkelhals*, *Wadenmuskel*, *Schmerzen*, *Herzrasen*, *Therapie*, *Kur*, *Spritze*, *Tablette* etc. aktiviert. D.h. ein Frame weist eine kompakte Strukturierung konzeptueller und sinnrelationaler Bezüge auf, wie sie unter MEANING für die entsprechenden Frame-Elemente im Frame HEAL in Abb. 4.1 exemplarisch definiert sind, – und stellt gegenüber den beliebigen Bezügen in WordNet eine Weiterentwicklung dar.

Prädizierende Kategorien⁷ wie Verben, Adjektive sowie deverbale, adjektivische und relationale Nomina dienen als Strukturelemente für semantische Frames, während referierende Nomina (**common nouns**) gewöhnlich als Filler der Frame-Elemente fungieren (vgl. Fillmore et al. 2002: „Predicating words [...] can be thought of as representing a kind of questionnaire listing the expected types of participants and props (frame elements) in their respective frames.“).

Die an WordNet orientierte Namensgebung deutet bereits an, dass die FrameNet-Entwickler ebenfalls einen Online-Thesaurus aufbauen. Dieser ist allerdings nicht synsetbasiert, sondern framebasiert. Im Unterschied zu WordNet erfolgte der Aufbau des FrameNet komplett korpusbasiert – jede Lesart und grammatische Variante wird durch ein Korpusbeispiel belegt⁸. Weniger reichhaltig strukturierte Nomina werden in FrameNet nicht annotiert, so dass FrameNet keine umfassende konzeptuelle Abdeckung aufweist. Diese kann durch die komplementären WordNet Hierarchien ergänzt werden. Anders als die wortart-internen Synsets können Frames Relationen zwischen Wörtern unterschiedlicher Wortarten aufweisen. Basiseinheit oder Frame-auslösendes Element ist die LU (**lexical unit**), die ebenfalls unter Rückgriff auf Cruse⁹ als ein Form-Bedeutungspaar, also eine disambiguierter Lesart, definiert wird. Polyseme Wortformen gehören also zu unterschiedlichen Frames. FrameNet enthält mehr als 10 000 LUs, die in ca. 800 hierarchisch relationierten Frames organisiert und mit mehr als 135 000 annotierten Sätzen belegt sind. Das FrameNet-Modell ist mittlerweile Ausgangspunkt für die Entwicklung framesemantischer Lexika in verschiedenen Sprachen, z. B. für das Deutsche („German FrameNet“ in Austin, (<http://gframenet.gmc.utexas.edu/>), „SALSA“ in Saarbrücken (<http://www.coli.uni-saarland.de/projects/salsa/>), für das Spanische („Spanish FrameNet“ in Barcelona, (<http://gemini.uab.es:9080/SFNsite>) und das Japanische („Japanese FrameNet“, (<http://jfn.st.hc.keio.ac.jp/>)). Außerdem ist mit BiFrameNet (Chen und Fung 2004) ein statistikbasierter Ansatz zur lexikalischen Beschreibung des Chinesischen und Englischen in Hinblick auf die maschinelle Übersetzung aufgebaut worden. Durch die Ausrichtung auf die natürlichsprachlichen Sätzen zugrunde liegenden konzeptuellen Strukturen eignet sich FrameNet vor allem für grammatische und diskursorientierte Fragestellungen in multilingualen computerlinguistischen Szenarien.

⁷Verben, Adjektive und relationale Nomina werden als Prädikate aufgefasst, d. h. als Elemente, welche die Realisierung von Partizipanten, den Argumenten, erfordern. Prädikate werden z. B. nach ihrer Stelligkeit unterschieden: so verlangen einstellige Prädikate wie *schön* oder *schlafen* genau ein Argument, zweistellige Prädikate wie *betrachten* zwei Argumente, usw. Vor allem bei Verben spricht man von Prädikaten mit obligatorisch zu realisierenden Argumenten. Die Schreibweise orientiert sich an der prädikatenlogischen Notation: *schön(Peter)* bzw. *betrachten(Peter, Carla)*; solange die Argumente nicht gebunden sind, werden Individuenvariablen wie *x* und *y* verwendet.

⁸Allerdings ist dies kein prinzipieller Unterschied, da WordNet seit 1985 entwickelt wurde, als noch nicht so umfangreiche digitale Sprachdatenkollektionen vorlagen wie heute. Neuere Wortnetzentwicklungen, z. B. das polnische Wortnetz, werden ebenfalls korpusbasiert aufgebaut, vgl. Derwojedowa et al. (2007).

⁹Vgl. Cruse (1986).

Informationen zu Lexical Units in FrameNet

FrameNet beschreibt die syntaktische und semantische Valenz bedeutungsträgender prädizierender Kategorien. Dazu werden für eine gegebene lexikalische Einheit (LU) alle syntaktischen und semantischen Rahmen, die möglich und im *British National Corpus* und anderen Referenzkorpora belegt sind, annotiert.

Für die Annotierung von Frame-Element-Realisierungen im laufenden Text sind Tripel vorgesehen, die z. B. das Frame-Element FOOD, seine grammatische Funktion (z. B. OBJECT) und den Phrasentyp (NP) spezifizieren. Eine vollständige Annotierung enthält

1. die o.g. dreischichtige Repräsentation für alle Sätze;
2. die vollständigen Frames und Frame-Element-Beschreibungen;
3. die Beziehungen zwischen den Frames
4. sowie lexikalische Einträge, die Valenzpattern für jede annotierte LU zusammenfassen.

Die lexikalischen Einheiten (LUs) in FrameNet werden wie folgt behandelt:

- LUs erhalten eine Definition, die entweder aus dem COD (Concise Oxford Dictionary) übernommen oder von den FrameNet-Lexikographen erstellt wird;
- alle kombinatorischen Möglichkeiten der LU in Bezug auf syntaktische und semantische Valenz werden exemplifiziert;
- die Beispiele sind allesamt Korpusbelege aus dem *British National Corpus* oder aus repräsentativen Korpora des *Linguistic Data Consortium* (LDC) und keine Konstrukte;
- jede LU ist mit einem passenden semantischen Frame verknüpft und somit auch zu anderen Einheiten, die diesen Frame aktivieren, wodurch sich Gruppen semantisch ähnlicher oder korrelierter LUs herausbilden: dies unterstreicht die konzeptuell basierte und nicht wortformbezogene Vorgehensweise.

Darüber hinaus werden hierarchische Beziehungen zwischen einzelnen Frames und Frame-Elementen etabliert, welche die in FrameNet kodierte Information vielfältig verknüpfen.

Relationen zwischen Frames und Frame-Elementen

Die konzeptuelle Modellierung semantischer Felder in Frames legt nahe, die semantischen Relationen zwischen Frames nutzbar zu machen. Unter anderem gibt es die folgenden charakteristischen Beziehungen:

1. Vererbung zwischen übergeordnetem („parent frame“) und untergeordnetem („child frame“) Frame in Form einer IS-A Hierarchie, wobei auch die Frame-Elemente des übergeordneten an diejenigen des untergeordneten Frames gebunden sind: so erbt der REVENGE-Frame die Frame-Elemente vom REWARDS _AND _PUNISHMENT-Frame.
2. Enthaltensein eines Subereignisses in einem übergeordneten komplexen Ereignis, ausgedrückt durch eine SUBFRAME-Beziehung zwischen Child Frame und Parent Frame. Ein Beispiel ist hier der CRIMINAL _PROCESS-Frame mit den Subframes ARREST, TRIAL und SENTENCING.
3. Präsupposition (Voraussetzen) eines übergeordneten Frames durch einen untergeordneten Frame, wobei aber nicht alle Frame-Elemente des Parent Frame an diejenigen des Child Frame gebunden sein müssen. z. B. setzt der SPEED-Frame den MOTION-Frame voraus.
4. Auch die Perspektivierung der Child Frames in Bezug auf einen neutralen Parent Frame kann erfasst werden: z. B. beim EMPLOYMENT _START-Frame mit den gerichteten Subframes GET _A _JOB aus Arbeitnehmersicht vs. HIRE aus Arbeitgeberperspektive.

4.3.3 Literaturhinweise

Einen umfassenden Überblick über den Aufbau des Princeton WordNet geben Miller et al. (1990) und Fellbaum (1998). Das deutsche Wortnetz, seine Strukturierungsprinzipien und Anwendungsperspektiven werden in Kunze und Wagner (1999) eingehend abgehandelt. Die Akquisition neuer Relationstypen wird in Lemnitzer et al. (2008) beschrieben. Vossen (1999) thematisiert die Integration einzelnsprachlicher Wortnetze in die polylinguale EuroWordNet-Architektur, und Wagner und Kunze (1999) exemplifizieren diese Integration für das deutsche Wortnetz. Publikationen über spezifische Forschungen zu und Anwendungen mit WordNet sind in einer ständig aktualisierten Bibliographie verzeichnet: <http://lit.csci.unt.edu/~wordnet/>. Eine gute theorieunabhängige Abhandlung über Wortbedeutungen und verschiedene semantische Relationen bietet Cruse (1986). Zu FrameNet gibt es viel mehr zu sagen, als an dieser Stelle möglich ist. Den besten und detailfreudigsten Überblick über FrameNet gibt das von den Entwicklern am ICSI erstellte, auf der FrameNet-Webseite verfügbare Manual (vgl. Ruppenhofer et al. 2006). Eine knappe Darstellung bieten Kunze und Lemnitzer (2007). Boas (2005) diskutiert Frames als interlinguale Strukturen zum Aufbau multilingualer Datenbanken.

4.4 Lexika für multimodale Systeme

Dafydd Gibbon

Ein **multimodales System** ist ein elektronisches Kommunikationssystem, das verschiedene menschliche Ein-/Ausgabekanäle (z. B. Stimme-Ohr, Gestik-Auge) verwendet. Ein multimodales System ist in der Regel auch ein **multimediales System**, d.h. ein System, das mehrere Übertragungskanäle (akustisch, visuell, taktil) einsetzt, die Unterscheidung „multimodal“ – „multimedial“ wird jedoch nicht immer klar eingehalten. Der Kern jedes Sprachverarbeitungssystems, auch eines multimodalen Systems, ist ein Lexikon. Ob sprechende Computer, CD-ROM-Enzyklopädien, Bibliotheksrecherchen, Internet-Suchmaschinen – alle verfügen über Inventare von Wörtern, Redewendungen und ihren Eigenschaften, die die Verarbeitung in wesentlichen Hinsichten steuern. Zweck dieses Beitrags ist es, einen Einblick in die Praxis der Entwicklung von Lexika verschiedener Arten zu geben, mit besonderem Augenmerk auf Lexika für multimodale, speziell audiovisuelle Systeme, weil typische sprachverarbeitende Systeme, die die gesprochene Sprache (also die akustische Modalität) verarbeiten, auch die geschriebene Sprache (z. B. bei Diktiersoftware) und eventuell auch Bildinformationen verarbeiten (also in beiden Fällen die visuelle Modalität). Für ausführliche Beiträge zu den hier behandelten Fragen sei auf Gibbon et al. (2000) und van Eynde und Gibbon (2000) verwiesen. Im Folgenden werden zunächst einige grundlegende Begriffe eingeführt, um dann detaillierter auf Eigenschaften von Lexika und die Konstruktion von Lexika als möglichst hochqualitative Ressourcen einzugehen.

4.4.1 Grundlagen

Etwas wie „das Lexikon“, in der Alltagssprechweise, gibt es nicht: Es gibt eine große Vielfalt von Lexikontypen, die alle jedoch aus folgenden drei systematischen Blickwinkeln betrachtet werden können, ob sie klassische gedruckte Lexika oder elektronische multimodale und multimediale Lexika sind: **Lexikoninhalt**, **Lexikonstruktur**, **Lexikonimplementierung**.

Die Inhalte eines Lexikons umfassen Metadaten über Art und Entstehung des Lexikons, sowie die lexikalischen Informationen, die in den einzelnen Einträgen enthalten sind.

Die Struktur eines Lexikons ist komplex und umfasst folgende Substrukturen:

1. die **Megastruktur** (die Gesamtorganisation des Lexikons, einschließlich Verlags- und Produktinformationen),
2. die **Makrostruktur** (die Organisation der Lexikoneinträge),
3. die **Mikrostruktur** (die Organisation der Informationen zu einzelnen Lexikoneinträgen),
4. die **Mesostruktur** (die Querverweise zwischen den Einträgen, den Einträgen und einem Korpus, den Einträgen und einleitenden Erläuterungen zu Abkürzungen und Notation).

Die Implementierung eines Lexikons kann sich auf visuelle Darstellungen beschränken, wie die traditionellen dicken Druckbände oder Textlexika auf CD-ROM oder im Internet. Die elektronischen Formate bieten aber die Möglichkeit, das Lexikon in Hypertextformat als **Hyperlexikon** zu gestalten, das heißt, als Grundtext mit Verweisen auf einzelne Einträge, die ihrerseits Verweise auf zusätzliche mesostrukturelle multimediale Informationen wie Bilder oder Audio- und Video-Clips enthalten. In der Architektur eines sprachverarbeitenden Systems kann ein Lexikon als Datenbanktabelle oder als programmiersprachen-spezifische Datenstruktur, und modular als eine Systemkomponente oder auf verschiedene Systemmodule verteilt vorliegen.

Sprachverarbeitende Systeme enthalten ebenfalls unterschiedliche Lexikonarten, je nachdem welche Art von Sprachverarbeitung involviert ist. Spracherkennungssysteme mit Textausgabe enthalten Inventare von Einheiten unterschiedlichster Art, wie z. B. Informationen über akustische Parameter des Sprachsignals, neben Aussprachelexika, in denen Graphem-Phonem-Relationen und Häufigkeitsinformationen enthalten sind, und grammatisch orientierte Lexika mit statistischen Informationen, die aus einem großen Korpus von annotierten Sprachaufnahmen gewonnenen werden. Heutige Sprachsynthesesysteme benötigen einerseits Lexika, mit denen Eingabetexte geparst und im Hinblick auf Submodalitäten wie Akzentuierung und Intonation sowie visuelle Modalitäten, z. B. Gestenpositionen, analysiert werden können. Andererseits werden Inventare von Ausspracheeinheiten benötigt (z. B. Diphonen oder größeren Phonsequenzen, Gestensequenzen), mit statistischen Informationen, die aus einem Korpus gewonnen werden. Multimodale Systeme, die visuelle Eingaben zur Handschrifterkennung oder zur Gestenerkennung erfordern, unterscheiden sich nicht grundsätzlich in diesen Anforderungen von den rein sprachverarbeitenden Systemen, außer dass Aufnahmen von Signalen in den einzelnen Modalitäten benötigt werden und komplexere Strukturen zu bearbeiten und zu koordinieren sind.

In einem multimodalen sprachverarbeitenden System kann das Systemlexikon für Ein- und Ausgaben in verschiedenen menschlichen Handlungs- und Wahrnehmungsmodalitäten ausgelegt sein: Tast-Eingaben (Tastatur, Handschrift), Sprache, Bildeingaben, sowie visuelle, akustische, taktile Ausgaben, letztere in der einfachsten Form durch Vibrationssignale, aber auch in komplexen Formen wie Blindenschrift oder Roboterbewegungen. Das Korpus, auf dem das Lexikon basiert, muss diese unterschiedlichen Signaltypen enthalten.

Eine wichtige Etappe in der Erstellung eines Korpus ist, neben Planung und Durchführung der Korpusaufnahmen, die Weiterverarbeitung, zunächst die **Annotation** des aufgenommenen Sprachsignals mit einer **Transkription** der Lautsequenzen, der Wörter, der Sätze, sowie gegebenenfalls der Wortarten und der relevanten semantischen und pragmatischen Kategorien, z. B. mit Dialogaktypen. Bei der Annotation werden mit geeigneter Software der Transkription Zeitstempel zugeordnet, die die zeitliche Zuordnung der Transkriptionseinheit zum akustischen (oder visuellen oder taktilen usw.) Signal definieren. Für die sprachliche Transkription werden in der Regel orthographische Transkriptionen verwendet, die anhand eines Graphem-Phonem-Übersetzungssystems in eine Aussprachere-präsentation unter Verwendung eines **phonetischen Alphabets** umgewandelt

werden, die bei der Annotation verwendet wird. Für die gestische Transkription existiert noch keine einheitliche Methode für die Repräsentation von Gesichtsmimik, Handformen, -stellungen und -bewegungen, Körperhaltungen usw., obwohl die Forschung in diesem Bereich sehr aktiv ist und einige geeignete Konzepte wie **Viseme**, analog zu Phonemen oder Morphemen in der gesprochenen Sprache, entwickelt worden sind und in ein gestisches Lexikon aufgenommen werden müssen.

Der Umfang des Lexikons eines multimodalen Systems kann wenige Wörter enthalten, wie in einer akustischen Maschinensteuerung, mehrere Zehntausende, wie in einer Diktiersoftware, oder Hunderttausende und mehr, wie in einer Forschungs- und Entwicklungsressource.

Die Makrostruktur eines Lexikons kann einfach aufgebaut sein, wie in einer Aussprachewortliste, komplexer, wie in einem Terminologie-Handbuch (vgl. Gibbon et al. 2000), oder sehr komplex, wie in einem Übersetzungslexikon, in dem subtile grammatische und semantische Unterschiede zwischen Sprachen definiert werden, etwa: Wieso geht im Deutschen „Ich schlage Dir vor, früher zu kommen“, während es im Englischen „I suggest you come earlier“ oder „I suggest that you come earlier“, aber auf keinen Fall „I suggest you to come earlier“ heißt, und im Deutschen wiederum „Ich suggeriere Dir, früher zu kommen“ erst recht nicht möglich ist? Ein Beispiel aus dem Bereich der aktuellen Wortprägung ist das Wort „Handy“: Bekanntlich sieht das deutsche „Handy“ = „Mobiltelefon“ wie ein englisches Wort aus, wird wie ein eingedeutschtes englisches Wort ausgesprochen („Hendi“), ist es aber nicht, sondern eine deutsche Erfindung. Im Englischen existiert das Wort nur als Adjektiv in der Bedeutung „nützlich“, „leicht handhabbar“; ein „handyman“ ist ein Handwerker und nicht etwa ein Vieltelefonierer, und ein Mobiltelefon heißt „cellphone“ oder „mobile (phone)“. Übersetzungssysteme müssen solche Unterschiede durch Regeln oder statistische Abbildungen in großer Zahl berücksichtigen. In multilingualen multimodalen Systemen müssen auf ähnliche Weise nichtkongruente Gesten berücksichtigt werden, die keinesfalls eins zu eins zwischen verschiedenen Kulturgemeinschaften übertragen werden können.

Schließlich kann auch ein elektronisches Nachschlagewerk als eine Art sprachverarbeitendes System angesehen werden, in dem ein Lexikon als einfache oder komfortable Datenbank vorliegt, im Internet oder auf CD-ROM, rein auf Sprachliches bezogen oder ausgebaut als Hyperlexikon zu einer umfangreichen elektronischen **Enzyklopädie**. Als ein spezielles lexikalisches Nachschlagewerk, nämlich als eine **Konkordanz**, also ein Lexikon, in dem die Einträge mit ihren Verwendungskontexten aufgelistet werden, kann auch eine Suchmaschine im Internet oder in lokalen Systemen angesehen werden: Hypertextseiten werden (unter anderem) nach lexikalischen Kriterien verschiedenster Art in Datenbanken zusammengefasst, nach lexikalischen und anderen Suchkriterien abgefragt, und mit Quellenangaben zurückgegeben.

4.4.2 Die Lexikographie

Das Lexikon, mit dem der Systemanwender direkt oder indirekt in Berührung kommt, ist immer das Endergebnis eines sehr aufwändigen Forschungs- und Ent-

wicklungsprozesses, der dem Anwender selbst meist verborgen bleibt. Dies gilt für klassische Drucklexika, deren Hauptvertreter mehr als ein Jahrhundert Entwicklungszeit benötigten und in ähnlicher Weise, wenngleich dank elektronischer Unterstützung im Zeitraffer und mit kleineren Teams, für moderne Systemlexika. Lexikographen sind die Spezialisten, die herkömmliche Lexika entwickeln, heute mit der Unterstützung moderner Datenbank- und Versionierungstechniken, also die Lexika, die sich als schöne und stattliche Bücher präsentieren, in denen ein Briefschreiber, ein Romancier, oder ein *Scrabble*- oder Quiz-Spieler nachschlägt, ob es das Wort „serendipity“ oder „Infometrik“ gibt. Erweiterungen des Tätigkeitsbereichs des Lexikographen sind die Entwicklung von Lexika für Diktiersoftware, für die Rechtschreib- oder Grammatikkorrekturfunktionen in Textverarbeitungssystemen, oder von Schlüsselwortsystemen, um die Datenbestände von Bibliotheken oder die Dokumentationsbeständen großer Firmen zu erschließen.

Die Lexikographie für multimodale Systeme umfasst, wie bereits angedeutet, **endosystemische Lexika** (Lexikonsysteme als Komponente eines sprachverarbeitenden bzw. multimodalen Systems) oder **exosystemische Lexika** (Lexikonsysteme zur Verwendung als Nachschlagewerk durch externe Nutzer).

Die Entwicklung von beiden Lexikonsystemtypen kann anhand der Abstraktionsstufen systematisiert werden, die die lexikographische Arbeit von der Korpusbearbeitung ausgehend über unterschiedlich abstrahierte Informationstypen bis zum formalen, theoretisch fundierten und formalisierten Lexikon charakterisieren (vgl. Abbildung 4.4.2). Ein Systemlexikon kann auf jeder der Lexikonebenen implementiert werden. Das klassische Lexikon als Nachschlagewerk ist ein Lexikon 3. Ordnung, mit lokalen Generalisierungen, die durch die Querverweise der Mesostruktur ausgedrückt werden. Das Lexikon 4. Ordnung ist z. B. ein Lexikon, das als Vererbungssystem oder semantisches Netz mit lexikalischen Informationen als Merkmalsstrukturen (Attribut-Wert-Strukturen) formalisiert ist.

Der Lexikon-Akquisitionsprozess wird heute durch Software-Werkzeuge zur automatischen Suche im Internet, zur automatischen Analyse von Texten im Hinblick auf Wörter, Redensarten, Terminologie, besonderen Wortzusammensetzungen, bevorzugten Wortkombinationen und Bedeutungszusammenhängen beschleunigt. Die Software-Werkzeuge und Datenbestände (Sprachkorpora, lexikalische Datenbanken, Termbanken), die in diesem Forschungs- und Entwicklungsprozess entstehen, werden als **lexikalische Ressourcen** bezeichnet. Am Aufbau von solchen lexikalischen Ressourcen sind viele Disziplinen beteiligt: Linguistik, Phonetik, Psycholinguistik, Informatik, die Sprachtechnologien und zunehmend, mit der Einbeziehung von aufgabenorientierter multimodaler Kommunikation, auch Teildisziplinen der Psychologie und der Soziologie.

Eine etwas andere Klassifikation lexikographischer Arbeit unterscheidet drei Typen: die **Lexikographie**, die **Lexikologie** und die **Lexikontheorie**.

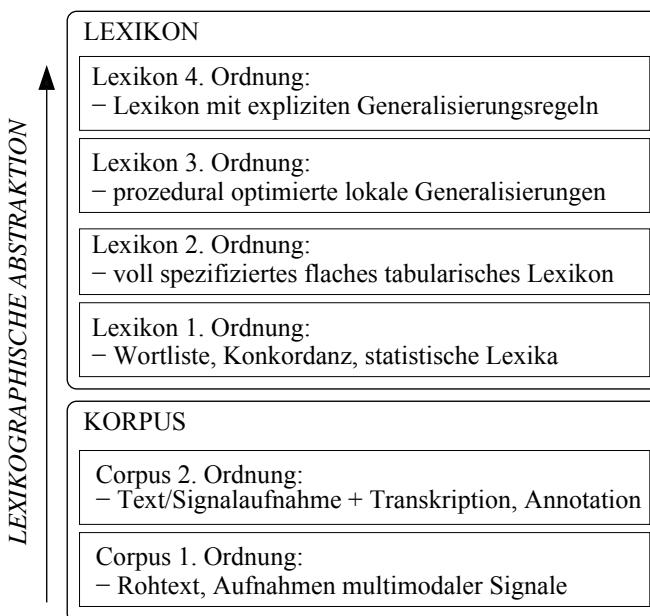


Abbildung 4.14: Arbeitsbereiche des Lexikographen: Korpora und Lexika unterschiedlicher Ordnung.

Die **Lexikographie** ist die praktische Arbeit an Texten, Aufnahmen gesprochener Sprache, Schaffung von Archiven, Produktion von Datenbanken und Büchern. Analog dazu ist in der multimodalen Lexikographie z. B. die Erstellung und Bearbeitung audiovisueller Korpora. Die **Lexikologie** befasst sich mit der linguistisch fundierten Beschreibung der Eigenschaften von Lexikon-Einträgen und stellt damit eine der wichtigsten Voraussetzungen für eine hochqualitative Lexikographie dar. Traditionell ist die Lexikologie in erster Linie mit Wortbedeutungen, Definitionen, festen Wortfügungen befasst; streng genommen können aber alle Arten lexikalischen Wissens darunter fallen, also auch z. B. die historische Entwicklung von Wörtern, ihren Bedeutungen und ihren Aussprachen, etwa die Beziehungen zwischen deutsch „Schürze“, englisch „shirt“ und „skirt“, die im Laufe der Zeit als Spezialisierungen auf verschiedene Teile des altgermanischen Kittels ihre heutigen Bedeutungen erlangten. Die **Lexikontheorie** bietet den Rahmen für konsistente, allgemeingültige Forschungsergebnisse, indem sie nicht nur die sprachübergreifenden Eigenschaften lexikalischer Informationen sowie die formalen mathematischen und logischen Grundlagen lexikalischer Repräsentationen untersucht, sondern unter anderem auch die kognitiven Eigenschaften des menschlichen mentalen Lexikons (vgl. Handke 1995) erforscht.

4.4.3 Lexikalische Struktur- und Informationstypen

Einige grundlegende Unterscheidungen sollen an dieser Stelle etwas ausführlicher als in den einleitenden Absätzen dargestellt werden.

„Types and Tokens“: Dieses englische Begriffspaar hat eine lange Vorgeschichte nicht nur in Lexikographie und Korpuslinguistik, sondern auch in der Sprachphilosophie. Ein **Token** wird allgemein als das einmalige konkrete Vorkommen eines Zeichens (ob Laut, Wort, Satz, Text, ...) an einem Ort zu einer Zeit definiert. Für schriftliche **Tokens**, die in einem festen Medium gespeichert und daher ganze Zeiträume überdauern können, wird manchmal die Bezeichnung **Inskription** verwendet. Analog dazu muss in multimodalen Systemen zwischen dem raumzeitlichen Vorkommen einer Geste und dem im Gistenlexikon enthaltenen Gestentyp unterschieden werden. Ein **Type** ist eine Klasse von Tokens, die von ihren Sprechern und Hörern (bzw. ihren Autoren und Lesern) nicht unterschieden werden und daher als gleich, z. B. als Kopien, wahrgenommen werden. Die ursprüngliche Unterscheidung durch den Philosophen C. S. Peirce war in eine komplexe Zeichentheorie eingebettet und hat in der Lexikographie im Laufe der Zeit etwas andere Bedeutungsnuancen angenommen. Die **Type-Token-Ratio**, das Verhältnis der Typen zu deren Vorkommen im Text, ist ein Indiz der Repräsentativität eines Korpus. Typischerweise flacht diese Funktion nach einer steilen Anfangskurve asymptotisch ab; wenn dies eintritt, ergibt sich die Frage, inwieweit sich die Erweiterung des Korpus noch lohnt. Im Kontext der Lexikographie kann ein Type als **Lexikonwort** und ein Token als **Korpuswort** bezeichnet werden.

Makrostruktur: Die Makrostruktur eines Lexikons ist seine Gesamtstruktur, also die Anordnung der Lexikoneinträge. Die Makrostruktur kann einfach eine Liste sein (wenn es z. B. nur um eine Rechtschreibprüfung geht), oder eine Tabelle (so kann man sich eine Lexikondatenbank vorstellen), oder eine Baumstruktur (beispielsweise zur Anordnung von Begriffen nach Bedeutungsfeldern in einem Thesaurus oder Synonymlexikon, die immer weiter verfeinert werden). Die Wahl einer Makrostruktur hängt von der Art der lexikalischen Einträge ab: Alltagswörter, Terme (Fachwörter), Redewendungen und Redensarten, Begriffsfelder. Eine klassische Einteilung (die aber für viele Zwecke nicht mehr ausreicht) ist die Einteilung in **semasiologische** Lexika, in denen die Makrostruktur an den Wortformen (Rechtschreibung, Aussprache) orientiert ist, denen Bedeutungen zugeordnet werden, und **onomasiologische** Lexika, in denen die Makrostruktur an den Bedeutungsfeldern orientiert ist, denen Wortformen zugeordnet werden. Auch andere Arten von Makrostrukturen, beispielsweise für Übersetzungswörterbücher, werden in sprachverarbeitenden multimodalen Systemen benutzt. Aus der Sicht der Datenbanktechnologie stellt die Makrostruktur das zugrunde liegende Datenmodell dar, z. B. als Relation oder als Objekthierarchie. Der Umfang der Makrostruktur eines Lexikons (im einfachsten Fall die Anzahl der Einträge in einer Liste) wird die **extensionale Abdeckung** des Lexikons genannt.

Mikrostruktur: Die Struktur eines einzelnen Lexikoneintrags heißt Mikrostruktur. Die Mikrostruktur kann sehr simpel sein, z. B. eine Einzelinforma-

tion, etwa die Rechtschreibung, oder eine Liste von Informationen verschiedener Art, typischerweise die *Rechtschreibung*, die *Aussprache*, die *grammatische Kategorie* und *weitere grammatische Eigenschaften*, eine *Definition* der Bedeutung, *Verweise* auf Synonyme, Antonyme, oder auf Wörter, die in anderen Relationen zum Lexikoneintrag stehen, und eine Liste der *Kontexte*, in denen der Eintrag verwendet wird (in Hyperlexika auch auf relevante Korpusstellen). In einem multimodalen System enthält das Lexikon darüber hinaus Informationen über die Ein-/Ausgabemodalitäten der Kommunikation: nicht nur akustisch interpretierbare Kategorien, sondern auch visuelle, beispielsweise für die Sprach- und Schrift-Ein-/Ausgabe oder für die Generierung von gestischen und mimischen Konfigurationen und Abläufen. In einer Lexikondatenbank wird ein Eintrag als Record dargestellt und die Mikrostruktur durch die Felder eines Records. Die Anzahl der Typen lexikalischer Information, die die Mikrostruktur der Einträge in einem Lexikon bestimmen, wird die **intensionale Abdeckung** des Lexikons genannt. Für die Verarbeitung der Mikrostruktur werden in multimodalen Lexika aufwändige statistisch unterstützte Suchfunktionen eingesetzt (vgl. Jurafsky und Martin 2009)

Mesostruktur: Lexikalische Ressourcen und Lexika für sprachverarbeitende Systeme sind nicht mit den Begriffen Makrostruktur und Mikrostruktur erschöpfend charakterisiert. Die vollständige Charakterisierung der Eigenschaften lexikalischer Einträge in der Mikrostruktur führt bei Lexika mit sehr großer extensionaler und intensionaler Abdeckung zu extrem großen lexikalischen Datenbeständen. Aus diesem Grund werden Kompressionstechniken verwendet, von denen einige auf linguistisch fundierten systematischen Ähnlichkeiten unter den Einträgen beruhen. Zusammenfassungen von Ähnlichkeiten in der Mikrostruktur verschiedener Wörter (Wortklassen, Bedeutungsfelder usw.) werden der **Mesostruktur** eines Lexikons zugeordnet. Schon traditionelle Lexika verwenden diese Technik: Es wird nicht bei jedem Verb erklärt, was ein Verb ist, sondern es werden Abkürzungen verwendet, die gesondert erläutert werden. In einer Datenbank (und auch in einem traditionellen Lexikon) wird die Mesostruktur durch vielfältige Arten von Querverweisen dargestellt. In modernen lexikalischen Ressourcen wird im Prinzip dieselbe Strategie angewendet, außer, dass dafür Techniken aus der künstlichen Intelligenz verwendet werden, um Ähnlichkeiten als Netzwerke unterschiedlicher Art darzustellen (vgl. Fellbaum 1998, Evans und Gazdar 1996)

Metadaten: Eine Inhaltskategorie, deren Wichtigkeit für lexikalische Ressourcen erst in den letzten Jahren voll erkannt worden ist, weil sie für Suchfunktionen unerlässlich ist, betrifft die Dokumentation der Eigenschaften des Lexikons selbst, nicht nur der Einträge. Die Definition von Makrostruktur, Mesostruktur und Mikrostruktur, von verwendeten Text- und Aufnahmedaten, verantwortlichen Lexikographen usw. stellen die **Metadaten** eines Lexikons dar. Zur Lexikondokumentation werden Auszeichnungskonventionen, z. B. im XML-Format entwickelt (vgl. Thompson und McKelvie 1997, Lobin 2000), die den Aufbau von wiederverwendbaren, und damit qualitativ und quantitativ überprüfbaren, recycling-fähigen lexikalischen Ressourcen ermöglichen, beispielsweise in Archiven, die viele verschiedene Lexikondatenbanken enthalten, oder in Web-Portalen, die einen einheitlichen Zugriff auf verschiedene lexikalische Informationsquellen im Internet bieten.

Schritte im Lexikonaufbau

Abbildung 4.15 zeigt die wichtigsten Lexikonaufbauverfahren. Angefangen mit einer systematischen Datensammlung (vgl. Gibbon et al. 1997) wird eine konsistente Version der Schriftformen und Transkriptionen vorgenommen, heutzutage oft als DTD (Document Type Description) in XML (oder in daraus weiterentwickelten Datenstrukturen wie RDF oder XML-Schema), erstellt. Die für die Lexikonerstellung relevanten Einheiten werden identifiziert und einer linguistischen Analyse unterzogen. Es sollen z. B. nicht unbedingt alle Flexionsformen eines Verbs getrennt in die Ressource – im Deutschen auch bei regelmäßigen Verben nicht wenige: *bilden*, *bilde*, *bildest*, *bildet*, *bildete*, *bildestest*, *bildeten*, *bildend*, *gebildet*. Es genügt, die Wurzel *bild* zu erfassen, eine Normalform zu Identifikationszwecken auszuwählen (z. B. Infinitiv bei Verben, Nominativ Singular bei Substantiven), und alles andere wird über die Mesostruktur geerbt, in der die Eigenschaften von Hunderten von ähnlichen Verben zusammengefasst werden können. Dieses Verfahren kann formal insgesamt als Operationalisierung einer Funktion definiert werden, die zunächst Korpuswörter auf voll flektierte Lexikonwörter, und diese dann auf nichtflektierte Stämme (morphologische Grundformen) abbildet.

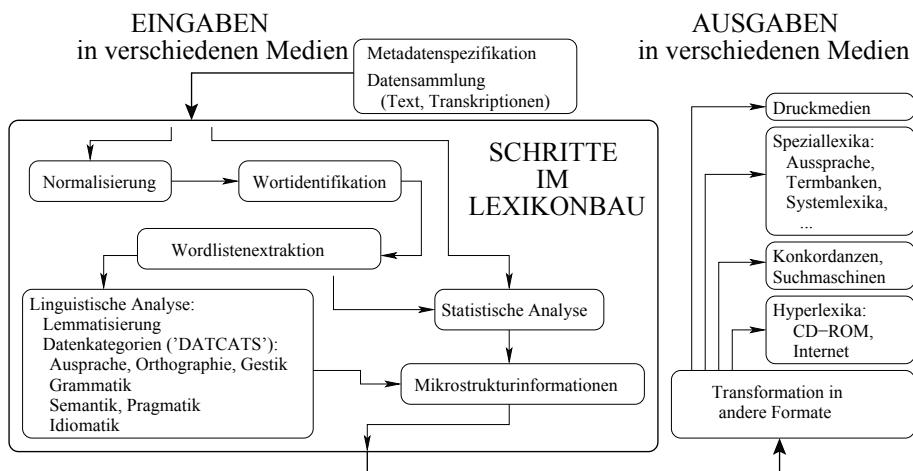


Abbildung 4.15: Phasen der Lexikonentwicklung

Parallel zur linguistischen Analyse wird eine statistische Analyse (vgl. Manning und Schütze 2003) durchgeführt und ein statistisches Modell für das Verhalten lexikalischer Einheiten im Kontext erstellt. Schließlich wurden die gewonnenen lexikalischen Informationen in geeigneter Weise transformiert und in Datenbanken archiviert, um je nach Bedarf in unterschiedlichen Anwendungskontexten in verschiedene mediale Formen transformiert zu werden: für den herkömmlichen Markt der Druckmedien, oder als elektronische Medien wie Hyperlexika oder dynamische, online erzeugte Konkordanzen.

4.4.4 Literaturhinweise

Einige Referenzwerke zu Lexika für gesprochene Sprache und multimodale Systeme sind bereits im Diskussionskontext angeführt worden (vgl. Gibbon et al. 1997 und Gibbon et al. 2000). Im Bereich Gestik sind die Arbeiten der Chicagoer Gruppe um McNeill wegweisend (McNeill 2005). Für die Definition grammatischer Kategorien kann auf Butt et al. (2000) verwiesen werden. Für die praktische Arbeit mit Netzwerklexika sowohl für gesprochene als auch für geschriebene Sprache stellen Evans und Gazdar (1996) und Gibbon und Strokin (1998) ein praktisches Werkzeug zur Verfügung. Systemorientierte Überblicke und Vertiefungsinformationen bieten Jurafsky und Martin (2009) und van Eynde und Gibbon (2000). Und schließlich bieten Internet-Suchmaschinen als „Megakonkordanzen“ eine Fülle von Informationen dazu, wenn die Suche nur systematisch und reflektiert gestaltet wird.

4.5 Sprachdatenbanken

Christoph Draxler

Gesprochene Sprache und geschriebener Text unterscheiden sich nach Tillmann (1997) in vielfacher Hinsicht. Ein wesentlicher Unterschied ist, dass gesprochene Sprache ein *Signal* ist und geschriebener Text ein *System von Symbolen*. Zwischen einem Sprachsignal und dem es repräsentierenden Text besteht eine *empirische Beziehung*, die erlernt werden muss.

Diese Beziehung ist der Forschungsgegenstand der Phonetik, der Psycho-, Neuro-, Sozio- und Computerlinguistik sowie der angewandten Linguistik. Auch in der Sprachtechnologie steht diese Beziehung im Fokus: bei der Spracherkennung wird ein Sprachsignal maschinell auf linguistische Einheiten abgebildet, bei der Sprachsynthese werden linguistische Einheiten als computergeneriertes Sprachsignal ausgegeben (vgl. 5.4 bzw. 5.5).

Sprachdatenbanken bilden die empirische Basis für die Forschung und sind Ausgangspunkt für die Entwicklung von Sprachtechnologie und sprachverarbeitenden Anwendungen.

4.5.1 Definition

Eine **Sprachdatenbank** ist eine wohlstrukturierte und auf Dauer angelegte Sammlung von Sprachsignal- und Textdaten in digitaler Form (Gibbon, Moore und Winski 1997). Sie besteht aus drei Klassen von Daten:

- **Primärdaten:** Signaldaten in Form von Audio-, Video- und Sensordaten.
- **Sekundärdaten:** Annotationstexte, d. h. beschreibende symbolische Daten.
- **Tertiärdaten:** Lexika, Metadaten, Protokolldaten, Dokumentation, Validierungsberichte sowie Vereinbarungen zu Urheber- und Nutzungsrechten.

Die Primärdaten sind unveränderlich: einmal aufgenommen, werden sie, abgesehen von einfachen Formatkonvertierungen, nicht mehr verändert. Im Gegensatz dazu sind Sekundär- und Tertiärdaten veränderlich: neue Daten können zu bestehenden hinzugefügt werden, bestehende korrigiert oder sogar gelöscht werden.

Primär- und Sekundärdaten bilden den eigentlichen Inhalt der Sprachdatenbank, Tertiärdaten dienen hauptsächlich der Information über die Sprachdatenbank und dem Auffinden der darin enthaltenen Daten.

4.5.2 Primärdaten

Ein **Signal** ist eine sich verändernde physikalische – und damit messbare – Größe. Ein **Sprachsignal** ist periodisch schwingender oder sich aperiodisch verändernder Luftdruck, hervorgerufen durch die Artikulationsorgane des Menschen und wahrgenommen über das Ohr.

Die Anzahl der Schwingungen pro Zeiteinheit ist die **Frequenz**; sie wird üblicherweise in Schwingungen pro Sekunde oder Hertz (abgekürzt Hz) angegeben. Der Frequenzbereich menschlicher Sprache liegt im Bereich von ca. 50 bis über 6.000 Hz. Die **Amplitude** bezeichnet die Auslenkung, die Stärke der Schwingung. Mit **Dynamik**, angegeben in Dezibel (dB) bezeichnet man das Verhältnis zweier Signalstärken zueinander.

Die Grundform einer Schwingung ist die Sinusschwingung. Durch Addition mehrerer Sinusschwingungen unterschiedlicher Frequenz, Phase und Amplitude können zusammengesetzte Schwingungen erzeugt werden, in komplexeren Transformationen auch **Sprachlaute**. Diese sind von vielen Faktoren abhängig und höchst variabel: Die drei /ɪ/ in Abb. 4.16 a) unterscheiden sich deutlicher voneinander als die kategorial verschiedenen Phoneme /ɪ/ und /e/ in derselben Äußerung. In Abb. 4.16 b) überlappen sich die akustischen Eigenschaften selbst unterschiedlicher Phonemklassen erheblich (vgl. /ɪ/ und /ɛ/ bzw. /a/ und /ɔ/).

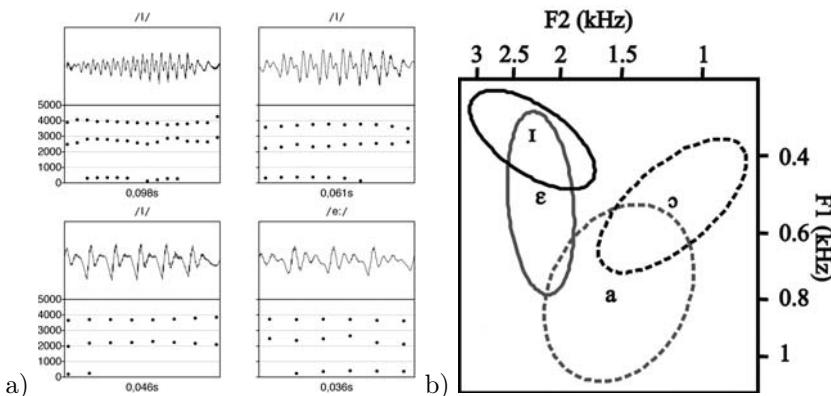


Abbildung 4.16: a) Oszillogramm und Formantverlauf der ersten vier Vokale aus dem Satz „Ich bin mit dem Wagen nach Bonn gefahren“, b) Messwerte der Formanten F1 und F2 für ausgewählte Vokale des Deutschen für einen männlichen Sprecher (nach Draxler 2008 bzw. aus Harrington 2009).

Bei der **Digitalisierung** wird der Signalwert eines analogen Signals zu bestimmten Zeitpunkten abgetastet. Die Anzahl Abtastpunkte pro Zeiteinheit ist die **Abtast- oder Samplerate** (ebenfalls in Hertz angegeben), der Wertebereich der Signalwerte die **Quantisierung**.

Gemäß dem **Nyquist-Theorem** muss die Abtastrate höher sein als das Doppelte der höchsten abzutastenden Frequenz. Daher ist für Sprache eine Abtastrate von mindestens 16 kHz üblich. Eine Quantisierung von 16 Bit erlaubt eine Dynamik von 96 dB und ist damit für Sprachaufnahmen meist ausreichend. Die **Datenrate** ist das Produkt aus Abtastrate und Quantisierung. Sie beträgt für Sprachsignale somit $16\text{kHz} * 16\text{Bit} = 256\text{kBIt/s}$ pro Aufnahmekanal.

Eine verlustfreie **Kompression** des Sprachsignals reduziert die Datenrate um 10–20 % ohne Qualitätsverlust. Verlustbehaftete Kompressionsverfahren errei-

chen Kompressionsraten von 25 % (ISDN) bis über 90 % (GSM Mobiltelefon, MiniDisk, mp3 u. a.), allerdings mit zunehmend schlechter werdender Signalqualität. Verlustbehaftete Kompressionsverfahren dürfen daher in Sprachdatenbanken nur in besonderen Fällen verwendet werden.

Grundvoraussetzung einer Sprachdatensammlung ist eine technisch ausgereifte digitale Aufnahmetechnik: dem Anwendungsgebiet entsprechende Mikrofone mit geeigneten Mikrofonverstärkern, portable digitale Recorder, Laptops oder PCs mit hochwertigen Soundkarten oder externen Audio-Interfaces bzw. digitale Videorecorder. Telefonaufnahmen erfolgen auf einem Sprachserver über eine ISDN-Karte.

Neben dem Audiosignal werden häufig auch Video- und Sensordaten aufgenommen. Sie erfassen physiologische Eigenschaften wie die Schwingungen der Stimmlippen (Laryngographie), Kontakt der Zunge mit dem Gaumen (Palatographie), Bewegungen und Form der Zunge (Artikulographie) usw.

4.5.3 Sekundärdaten

Eine **Annotation** ist eine symbolische Repräsentation des Inhalts einer Äußerung gemäß einem standardisierten Verfahren unter Verwendung eines vorgegebenen Symbolinventars. Erst mit der Annotation werden Sprachsignale einer weiteren systematischen Bearbeitung zugänglich – Klassifikation, Analyse, Suche usw. setzen einen annotierten Datenbestand voraus.

Das Erstellen einer Annotation ist eine Zuordnung, ein *Kategorisierungsprozess*: Einem Signal wird aufgrund bestimmter Kriterien ein Symbol oder eine Symbolfolge zugeordnet. Eine Annotation ist stets theorieabhängig und wird von menschlichen Experten erstellt. Sie ist daher subjektiv und kann somit niemals korrekt, sondern höchstens plausibel sein.

Annotation von Sprachsignalen

Die Annotation von Sprachsignalen erfolgt in zwei Schritten: **Segmentierung** und **Etikettierung**. Bei der Segmentierung wird das Sprachsignal in disjunkte oder überlappende Abschnitte unterteilt, bei der Etikettierung wird diesem Abschnitt ein **Label** zugeordnet, z. B. ein Phon- oder Phonemsymbol, eine Silbe, ein Wort, o. ä. Ein **Segment** bezeichnet ein Tupel aus Signalabschnitt und zugeordnetem Label.

Ein Sprachsignal kann auf verschiedenen Ebenen annotiert werden. Diese Ebenen können **zeitbezogen** oder **symbolbezogen** sein. Bei zeitbezogenen Annotationsen ist der Signalabschnitt des Segments durch Zeitangaben bestimmt, z. B. Anfangs- und Endpunkt im Signal, bei symbolbezogenen durch einen Verweis auf ein Element auf derselben oder einer anderen Annotationsebene. Die Annotationsebenen können **hierarchisch** oder **netzwerkartig** organisiert sein.

Abb. 4.17 zeigt eine Äußerung mit drei Annotationsebenen. Die phonetische Segmentation enthält Zeitmarken des Sprachsignals. So wird dem Signalabschnitt von t_1 bis t_2 das phonetische Label [i] zugeordnet, dem Abschnitt von t_2 bis t_3 das Label [ç].

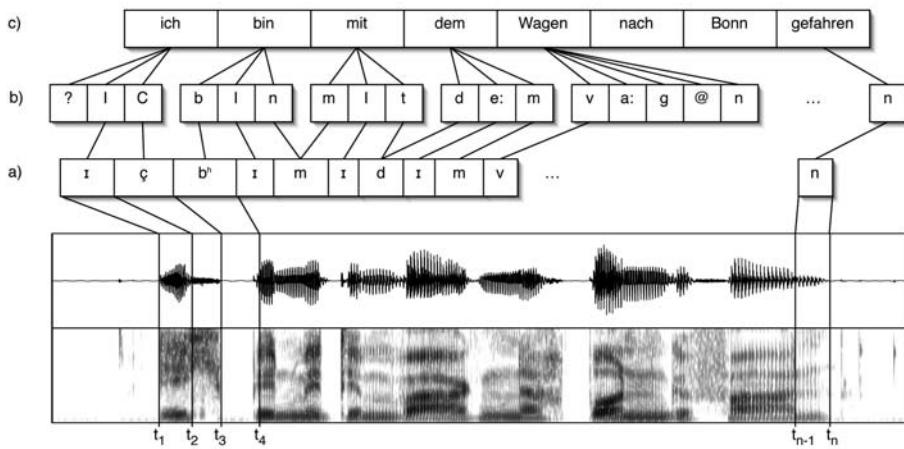


Abbildung 4.17: Oszillogramm und Sonagramm der Äußerung „Ich bin mit dem Wagen nach Bonn gefahren“ mit den Annotationsebenen a) phonetische Segmentation, b) kanonische Aussprache, c) orthographische Transliteration.

Die Segmente der phonetischen Segmentation sind mit Phoneminstanzen der kanonischen Aussprache und diese wiederum mit Wörtern der orthographischen Transliteration verknüpft. Zwischen der phonetischen Segmentation und der kanonischen Aussprache besteht eine netzwerkartige Beziehung: eine Phoneminstanz kann durch null oder mehrere Phone realisiert werden, und ein Phon kann mit null oder mehreren Phoneminstanzen verknüpft sein. So wird im Beispiel der erwartete Glottalverschluss /ʔ/ gar nicht realisiert, die Phonemfolge /n m/ aus „bin mit“ dagegen durch ein einziges [m]. Zwischen der orthographischen Transliteration und der kanonischen Aussprache besteht eine hierarchische Beziehung: ein Wort besteht aus mehreren Phonemen, und jedes Phonem gehört zu genau einem Wort.

Die IPA empfiehlt für eine Sprachdatenbank eine Annotation auf mindestens drei Ebenen: orthographische Transliteration der Äußerung, kanonische Aussprache und zeitbezogene phonetische Segmentation (Esling 1990). Darüberhinaus werden Sprachdatenbanken meist noch auf weiteren Ebenen annotiert: prosodisch, syntaktisch, diskursanalytisch usw., wobei diese Ebenen häufig erst nach und nach im Laufe der Zeit hinzukommen.

Erstellung und Speicherung

Die Erstellung einer Annotation ist sehr zeitaufwändig. Mit dem **Echtzeitfaktor** gibt man an, um wieviel länger als die Dauer des Sprachsignals seine Annotation auf der jeweiligen Annotationsebene dauert. Für eine einfache „ok/nicht ok“-Beurteilung beträgt er > 1, für eine orthographische Transliteration 10–20, für eine breite phonematische Transkription 100–250 und eine enge phonetische Segmentierung 500–1000.

Semi-automatische Annotationsverfahren erstellen eine Rohannotation, die in einem weiteren Arbeitsschritt manuell überprüft wird. Auf diese Weise lässt sich die Annotation zwar prinzipiell erheblich beschleunigen, aber solche Verfahren stehen nur für wenige Annotationsebenen zur Verfügung.

Da die Sekundärdaten veränderlich sind, ist es sinnvoll, Annotationsdaten Plattform-unabhängig und getrennt von den Signaldaten in eigenen Textdateien oder in Datenbankverwaltungssystemen abzulegen. Aufgrund der Theorieabhängigkeit ist trotz der weiten Verbreitung von XML eine Datenkonvertierung von Annotationen problematisch (vgl. hierzu Bird und Liberman 1999).

4.5.4 Tertiärdaten

Das **Aussprachelexikon** enthält für sämtliche in der orthographischen Transliteration vorkommenden Wörter und Einheiten eine Angabe der Standardaussprache, auch **kanonische Aussprache** genannt. Zusätzlich kann das Aussprachewörterbuch Aussprachevarianten, Frequenzangaben für die Wortformen sowie Angaben zur Morphologie, zur Prosodie oder grammatischen Funktion wie Part-of-Speech umfassen. Für die Aussprache wird dabei das sprachunabhängige IPA-Alphabet verwendet, alternativ auch das sprachspezifische SAMPA.

Die **Dokumentation** einer Sprachdatenbank besteht aus der Spezifikation und einem Validierungsbericht.

Die **Spezifikation** wird vor der eigentlichen Datensammlung erstellt. Sie legt fest, welche Anforderungen eine Sprachdatenbank erfüllen muss. Für die systematische Beschreibung eignen sich die Attribute bibliografische Angaben, demografische Vorgaben, Vokabular und Sprechweise, Annotationsebenen, Signalqualität, Datenformate und Speichermedien sowie Verfügbarkeit.

Die bibliografischen Angaben umfassen die Bezeichnung der Sprachdatenbank, Urheber und Herausgeber sowie das Publikationsdatum, eine Versionsnummer und eine Kurzbeschreibung.

Zu den demografischen Vorgaben zählen die Anzahl der Sprecher sowie eine Auflistung der notwendigen Angaben, z.B. Geschlecht und Alter zum Zeitpunkt der Aufnahme. Häufig kommen noch Angaben über die regionale Herkunft (Muttersprache, Dialekt), Bildungs- und Sozialstand (Schulabschluss, Studium, Beruf) sowie biometrische Daten (Größe, Gewicht, Gesundheitszustand) hinzu.

Das Vokabular wird entweder durch Auflistung oder durch eine Anleitung oder Umschreibung festgelegt. Zur Sprechweise zählen Angaben über die Sprechsituation, die Evozierung der gesprochenen Sprache und zum Sprechstil. Die Sprechsituation ist entweder ein Monolog oder ein Dialog. Der Dialog kann frei, moderiert oder programmgesteuert sein. Die Äußerungen reichen von gelesenen Einzelworten bis hin zu völlig freier Spontansprache. Der Sprachstil reicht von extrem sorgfältig artikulierter Hochsprache über normale Umgangssprache hin zu starkem Dialekt, Slang oder Berufssprache. Sprechgeschwindigkeit, Intonation und Vokabular hängen stark vom Sprachstil ab.

Zur Spezifikation gehören außerdem zur Nutzung der Sprachdatenbank hilfreiche Dokumente wie Listen der verwendeten Alphabete, Codetabellen und Programmksripte.

Nach Abschluss der Datensammlung wird die Sprachdatenbank validiert. In einem **Validierungsbericht** werden Abweichungen gegenüber der Spezifikation aufgelistet und die technische Qualität der Datenbank beurteilt.

Die **Metadaten** umfassen alle bei der Datensammlung systematisch erfassten Daten über Sprecher, Aufnahmen und Verarbeitungsschritte. Metadaten erlauben das Browse in Sprachdatenbanken und die Formulierung von Selektionsabfragen (siehe 4.5.5). Da Metadaten nicht zum eigentlichen Inhalt von Sprachdatenbanken zählen, sind sie in Datenbankkatalogen häufig frei verfügbar.

Sprachdatenbanken sind eine kommerziell wertvolle Ressource, denn ihre Erstellung ist mit hohem Aufwand verbunden. Zu ihrer Nutzung werden Vereinbarungen zwischen den beteiligten Parteien Sprecher, Datenbankeigentümer und Nutzer getroffen. Sprecher erhalten für ihre Aufnahmen ein Honorar und übertragen damit die **Urheberrechte** an den Sprachaufnahmen an den Produzenten der Datenbank. Eine **Nutzung** der Datenbank durch Dritte ist über die Vergabe von **Lizenzen** möglich, wobei meist zwischen akademischer, Forschungs- oder kommerzieller Lizenz unterschieden wird.

Um Sprachdatenbanken langfristig verfügbar zu halten, wurden mit der ELRA/ELDA (European Language Resources Association, bzw. Distribution Agency) in Europa und dem LDC (Linguistic Data Consortium) in den USA Institutionen zur Distribution von Sprachdatenbanken gegründet. Ähnliche Institutionen gibt es auch auf nationaler Ebene, z. B. das BAS (Bayerisches Archiv für Sprachsignale) in Deutschland. Diese Agenturen fördern die Erstellung und Veröffentlichung von Sprachdatenbanken, sie organisieren die Distribution und sie halten Kataloge verfügbarer und geplanter Sprachdatenbanken bereit.

4.5.5 Software

Sprachaufnahmesoftware wie SpeechRecorder (Draxler und Jänsch 2004) oder andere ist speziell an den Workflow beim Erstellen von Sprachdatenbanken angepasst und daher normalen Aufnahmeprogrammen vorzuziehen.

Annotationsseditoren erlauben dem Benutzer eine komfortable und effiziente Bearbeitung und Analyse von Annotationstexten.

Solche Editoren verfügen in der Regel über eine grafische und akustische Ausgabe, Editierfelder und interaktive Funktionselemente wie Schaltflächen und Menüs für die Annotation. Zu den Funktionen gehören die Selektion von Signalabschnitten, Editierhilfen und formale Konsistenzkontrollen. Es gibt viele, z.T. frei erhältliche Annotationsseditoren für alle gängigen Plattformen und Annotationsverfahren, und trotz des hohen Implementationsaufwands werden laufend neue entwickelt (z. B. Cassidy und Harrington 1996, Boersma 1999, Draxler 1999, Schmidt und Wörner 2005 oder Sloetjes et al. 2007).

Zugriffssoftware

Der Zugriff auf Sprachsignaldatenbanken erfolgt entweder navigierend mit einem Browser oder direkt mittels einer Suchfunktion über eine Abfragesprache.

Ein Browser ermöglicht das interaktive Blättern in einem Datenbestand. In einem ersten Schritt erfolgt der Zugriff über Metadaten, z. B. durch Auswahl einer Sprache und weiteren Angaben; in weiteren Schritten werden die gewünschten Daten ermittelt. Die zum Blättern notwendigen Metadaten können lokal oder extern in Metadaten-Katalogen gespeichert sein (Broeder et al. 2000).

Für eine gezielte Suche nach Daten in einer Sprachdatenbank ist eine Abfragesprache notwendig. Diese Abfragesprache sollte an die jeweilige Annotationsebene angepasst sein, z.B. die *Suche nach Sätzen, in denen der Wortstamm ‚Abfahrt‘ vorkommt* auf orthographischer oder nach a: vor bilabialem Plosiv auf phonetischer Ebene.

Moderne Zugriffssoftware bietet grafische oder textbasierte Abfragesprachen mit logischen und anwendungsspezifischen Operatoren sowie statistischen Auswertungs- und visuellen Darstellungsmöglichkeiten. So ist Abb. 4.16 b) das Ergebnis einer Auswertung von Formantfrequenzen der langen Vokale des Deutschen für eine männliche Versuchsperson in der Software R und dem phonetischen Datenbanksystem Emu (Harrington 2009).

Wegen der Vielzahl der in der Praxis verwendeten Annotationssysteme und -verfahren gibt es keine einheitliche Abfragesprache für alle Sprachdatenbanken.

4.5.6 Anwendungsbereiche

Die folgenden Abschnitte beschreiben beispielhaft drei ausgewählte Anwendungsbereiche von Sprachdatenbanken.

Spracherkennung

Eine Sprachdatenbank für die Spracherkennung – analoges gilt für Sprecher- und Sprachenerkennung – enthält als Primärdaten Aufnahmen gesprochener Sprache, die das intendierte Anwendungsbereich möglichst vollständig abdecken. Konkret heißt dies, dass die Sprecherpopulation gemäß Geschlecht, Alter, regionaler Herkunft und weiteren Merkmalen den zukünftigen Nutzern des Systems entspricht, das Vokabular alle in der Anwendung benötigten Wörter, Phrasen usw. enthält und die technischen Signaleigenschaften denen der Anwendung entsprechen. Sekundärdaten sind mindestens eine orthographische Transliteration mit Markern für Geräusche und eine phonemische oder phonetische Segmentation. Tertiärdaten sind die demographischen Daten der Sprecherpopulation, ein Aussprache- und Variantenlexikon, Beschreibungen der verwendeten Annotationskonventionen sowie Validierungsberichte.

Soziolinguistik

Die Soziolinguistik beschäftigt sich mit dem Einfluss sozialer Faktoren auf die Sprache. Dieser Einfluss zeigt sich u. a. in Syntax, Lexik, Prosodie und Phonetik.

Die Primärdaten einer soziolinguistischen Sprachdatenbank umfassen Sprachaufnahmen von nach sozialen Kriterien ausgewählten Sprecherpopulationen. Zu diesen Kriterien zählen regionale Herkunft, Muttersprache, Bildungs- und Sozialstand, Informationen zu Migration, Familiensituation, usw. Das Sprachmaterial

besteht aus Audio- und auch Videoaufnahmen von sowohl freien Interviews als auch konstruiertem Material wie Wortlisten, oder aus regional- oder schichttypischem Sprachstil. Die Sekundärdaten sind orthographische Transliterationen und Diskursrepräsentationen, häufig auch phonemische Transkriptionen, seltener phonetische Segmentierungen. Tertiärdaten sind umfangreiche standardisierte demographische Angaben zu den Sprechern, Aufnahmeprotokolle sowie detaillierte Dokumentationen der Aufnahmeorte.

Phonetik

In der Phonetik bilden Sprachdatenbanken die empirische Basis von Untersuchungen zur Artikulation, Akustik und Perzeption gesprochener Sprache. Phonetische Sprachdatenbanken enthalten entweder speziell für eine Forschungsfrage erhobene Sprachdaten, aber auch historische Aufnahmen oder Daten, die für die Forschung interessant, aber aus Gründen des Aufwands und der Gesundheitsgefährdung (z. B. Röntgen) nur selten durchgeführt werden können. Es ist notwendig, diese Aufnahmen möglichst breit und dauerhaft nutzbar zu machen.

Die Primärdaten einer phonetischen Sprachdatenbank sind Audio-, Video- und Sensoraufnahmen, die inhaltlich und technisch sehr exakt spezifiziert sind. Sekundärdaten sind orthographische Transliteration, phonemische Transkription und eine phonetisch enge Segmentierung, häufig auch noch prosodische Annotation. Tertiärdaten sind auf die Untersuchung abgestimmte demographische Sprechermerkmale, die Spezifikation des aufgenommenen Materials und der Technik, sowie Aufnahme- und Verarbeitungsprotokolle.

4.5.7 Literaturhinweise

Eine umfassende Übersicht zum Thema Sprachdatenbanken gibt Gibbon, Moore und Wiski (1997). Esling (1990) beschreibt knapp die Mindestanforderungen an phonetisch annotierte Sprachdatenbanken. Die Erstellung von Sprachdatenbanken beschreiben Schiel et al. (2003) und ausführlich Draxler (2008).

Bird und Liberman (1999) entwickeln einen formalen Rahmen für die Beschreibung von Annotationen und geben dabei eine gute Übersicht. Kohler et al. (1994), Geumann et al. (1997) und Senia und van Velden (1997) sind Beispiele für unterschiedliche projektspezifische Annotationskonventionen.

Das Handbuch der IPA (1999) und die Berichte der SAM-Arbeitsgruppen (Tomlinson et al. 1988; Wells 1997), des NIST (National Institute for Standards and Technology, Garofolo und Fiscus 1996) und des BAS (Schiel et al. 1997) sowie Carletta et al. (2003) beschreiben phonetische Alphabete und Datenformate.

ELRA/ELDA (<http://www.elda.fr>), LDC (<http://www.ldc.upenn.edu>), BAS (<http://www.bas.uni-muenchen.de>), MPI (<http://www.mpi.nl>) und andere bieten auf ihren Webseiten ausführliche Informationen zu den verfügbaren Sprachdatenbanken.

4.6 Nicht-sprachliches Wissen

Kai-Uwe Carstensen

4.6.1 Die Relevanz nicht-sprachlichen Wissens für die CL

Nicht-sprachliches Wissen ist sicherlich kein zentraler Untersuchungsgegenstand der Computerlinguistik im Allgemeinen. Auch für die Linguistik scheint es nur insofern relevant zu sein, als sprachliche Zeichen grundsätzlich sowohl eine Form- wie auch eine –nicht-sprachliche– Inhaltsseite aufweisen und die Modellierung der einen Seite somit von den Erkenntnissen über die andere profitiert (dies gilt für beide Richtungen, s. hierzu als Beispiel Lang et al. 1991). In der formalen Semantik war nicht-sprachliches Wissen lange Zeit als sog. „Weltwissen“ verpönt, das keinen sprachwissenschaftlichen Mehrwert darstellt.

Tatsächlich ist die aus der Vernachlässigung der Wortsemantik resultierende Gleichsetzung nicht-sprachlichen Wissens mit Weltwissen aber unangemessen: nicht-sprachliches Wissen umfasst nicht nur die Kenntnis, was der Fall ist (Faktenwissen, episodisches Wissen), sondern ebenfalls, was es überhaupt gibt/wie unsere Auffassung der Welt strukturiert ist (konzeptuelles Wissen). Angelehnt an die philosophische Teildisziplin **Ontologie** („Lehre vom Seienden“) spricht man daher bei Letzterem auch von **ontologischem Wissen**. Moderne semantische Ansätze berücksichtigen diesen Umstand (vgl. die Qualiastruktur in Unterkapitel 3.6 oder die „ontological semantics“ von Nirenburg und Raskin 2004).

Nicht-sprachliches Wissen ist außerdem zentraler Bestandteil intelligenter Systeme (daher: „wissensbasierte“ Systeme). Es wird zur Kategorisierung sensorischen Inputs („Daten“), zur Problemlösung, zur Handlungsplanung und Kommunikation benötigt. *Konzeptuelle* Repräsentationen vermitteln zwischen Wahrnehmung, Handlung und Sprache.

In frühen Anwendungssystemen der Künstlichen Intelligenz (KI), den Expertensystemen, führte das Fehlen von Wissen über die Welt zur so genannten „Zerbrechlichkeit“ (*brittleness*): Schon die geringsten Abweichungen von vorgegebenen Eingabemustern führ(t)en zu Fehlern und Systemabstürzen, die für die Benutzer nicht nachvollziehbar waren. Hieraus entstand das Desiderat allgemein- und wiederverwendbarer Wissensressourcen (sog. *(common sense) knowledge sharing and reuse*), heute allgemein **Ontologien** genannt.

Für **Anwendungen** der Computerlinguistik ist nicht-sprachliches Wissen daher essentiell: im Verlauf des *Textverstehens* müssen Textrepräsentationen mit Hintergrundwissen verrechnet werden (z. B. für die Auflösung von Ambiguitäten und bei der Präsuppositionenrechtfertigung); nicht-sprachliche Wissensstrukturen sind Grundlage und Ausgangspunkt für die *Sprachgenerierung* (s. Unterkapitel 5.6); in der *maschinellen Übersetzung* (s. Unterkapitel 5.7) werden konzeptuelle Repräsentationen als Interlingua verwendet. Insbesondere basieren zukunftsweisende Versionen des World Wide Web wie z. B. das **Semantic Web** (s. Berners-Lee et al. 2001) auf Ontologien.

4.6.2 Was ist „Wissen“ (nicht)?

Was eigentlich ist „Wissen“: die Menge der Daten, auf die ein (natürliches oder künstliches) System zugreifen kann, die Menge an Information, die ihm zur Verfügung steht, die Menge der von ihm begründeten Annahmen, die wahr sind? Keine von diesen knappen Charakterisierungen ist zutreffend.

Zunächst einmal ist Wissen abstrakt und nicht in Form konkreter **Daten** zu erfassen; es ist etwas, das einem System zugeschrieben werden kann, ohne auf die konkrete Form zu verweisen, in der es realisiert ist. Zudem ist Wissen unendlich und zeigt so das Vorhandensein sowohl von Struktur- als auch von Verarbeitungsaspekten: Wissen liegt nicht nur explizit vor, sondern kann durch Inferenzprozesse erschlossen werden („Ich weiß, dass du weißt, dass ich weiß... dass ich existiere“). Gleichwohl bilden Daten eine wichtige Grundlage für Wissen, da Wissensstrukturen zum Teil anhand des systematischen Auftretens von Daten konstruiert werden (→ Lernen).

Wissen ist ebenfalls nicht gleichzusetzen mit „**Information**“, auch wenn das Paradigma der Informationsverarbeitung zentral für diesen Bereich ist. Information ist vielmehr das Bindeglied zwischen Daten und den Strukturen, die abstraktes Wissen realisieren: Daten sind dann Information, wenn sie als Instanzen schematischer Strukturen erkannt werden (dies charakterisiert die semantische Auffassung von „Information“ innerhalb der Kognitionswissenschaft, die von der syntaktischen, inhaltsleeren Auffassung der Informationstheorie zu unterscheiden ist). Entsprechend wird deutlich, dass beispielsweise **Informationsextraktion** (s. Unterkapitel 5.3) als wichtige Anwendung der Computerlinguistik mehr ist als reines Sammeln von Daten, indem es zwingend Wissen voraussetzt. Gleichzeitig würde „Wissensextraktion“ implizieren, dass zusätzlich Wissenstrukturen aufgebaut werden.

Im Bereich der Computerlinguistik und der Künstlichen Intelligenz wird im Gegensatz zur philosophischen Tradition ein erweiterter Wissensbegriff verwendet (~ **Kenntnis**): so ist z. B. „syntaktisches Wissen“ nicht mithilfe von begründeten wahren Annahmen oder ähnlichen, auf rationalen Erwägungen basierenden Konzepten zu beschreiben. Auch hier zeigt sich *Wissen* als abstrakte Beobachtungskategorie, wodurch offen bleibt, wie es realisiert ist.

4.6.3 Wissen und Wissensrepräsentation

Kerngebiet der Beschäftigung mit **nicht-sprachlichem Wissen** ist der Bereich der **Wissensrepräsentation** innerhalb der KI bzw. der Kognitionswissenschaft. Interessanterweise waren es vor allem sprachlich orientierte Ansätze, die die Notwendigkeit und das Potential der Repräsentation nicht-sprachlichen Wissens aufgezeigt haben (z. B. Quillian 1968; Schank 1975), sowie das bekannt-berüchtigte ELIZA-Programm Joseph Weizenbaums als Karikatur eines Systems, das gerade nicht über Wissensrepräsentationen verfügt.

„Wissensrepräsentation“ bezeichnet einerseits die Realisierung abstrakten Wissens in einem konkreten (physikalischen) System (also Mensch oder Maschine) und andererseits die formal zu erfassenden Strukturen, die sich aus der Interak-

tion eines vorstrukturierten informationsverarbeitenden Systems mit seiner Umwelt (→ Lernen) ergeben. Das Verhältnis von Wissen zu dessen Repräsentation ist am markantesten in der sogenannten *Knowledge Representation Hypothesis* ausgedrückt:

Any mechanically embodied intelligent process will be comprised of structural ingredients that a) we as external observers naturally take to represent a propositional account of the knowledge that the overall process exhibits, and b) independent of such external semantical attribution, play a formal but causal and essential role in engendering the behavior that manifests that knowledge. (Smith 1982, S. 22).

Den „structural ingredients“ entsprechen in *symbolischen Ansätzen* zur Wissensrepräsentation Systeme von Symbolen mit einem jeweils spezifischen Bedeutungsgehalt, wobei die Symbole als in irgendeiner Weise physikalisch realisiert aufgefasst werden (sog. *Physical symbol system hypothesis*, s. Newell und Simon 1976). Ein Beispiel hierfür sind Symbole für Konzepte, denen in der realen Welt Einzeldinge oder Mengen solcher Dinge entsprechen. Die Frage jedoch, wie diese Symbolsysteme genau in den Erfahrungen bzgl. der Umwelt verankert sind, ist als das „symbol grounding problem“ (Harnad 1990) bekannt geworden.

Dem gegenüber stellen *subsymbolische* bzw. *konnektionistische Ansätze* das Verkörpersein (embodiment) und die Situiertheit neuronaler Netze, die von vornherein auf einer Sensor/Input-Aktor/Output-Korrelation beruhen und in denen sich ein von außen zugeschriebener Bedeutungsgehalt (z. B. „Dies ist das Konzept für X“) aus den Aktivationsmustern einer Vielzahl von Neuronen/Einheiten ergibt.

Beide Ansätze haben ihre Stärken und Schwächen und sind gegenwärtig daher als komplementär zueinander anzusehen (s. aber das in Hitzler und Kühnberger 2009 formulierte Desiderat der Synthese beider Bereiche). Beispielsweise erfassen subsymbolische Ansätze sehr viel besser die graduellen Unterschiede, kontextuellen Abhängigkeiten und impliziten Zusammenhänge in eng begrenzten Anwendungsbereichen. Symbolische Ansätze bleiben jedoch vorerst für die Entwicklung komplexer natürlich-sprachlicher Systeme besser geeignet. Dies gilt insbesondere für die Erstellung umfangreicher Ressourcen nicht-sprachlichen Wissens.

4.6.4 Aspekte der Wissensrepräsentation

Allgemeine Aspekte

Kern der Wissensrepräsentation ist die Darstellung der im Folgenden aufgeführten generischen Wissensrepräsentationskonstrukte: **Konzepte** (dt. Begriffe) als Repräsentanten von Entitäten der Welt (zu unterscheiden sind hier Klassenkonzepte (→ generisches Wissen über Dinge) und Individuenkonzepte/**Instanzen** (→ Wissen über Einzeldinge); **Attribute** als Repräsentanten der Eigenschaften solcher Entitäten; **Relationen** als Repräsentanten von Beziehungen zwischen Dingen; **Regeln** als Repräsentanten der Beziehungen zwischen Sachverhalten. Die Aufgabe der Wissensrepräsentation ist die formale Explikation dieser

Aspekte, so dass alles relevante Wissen – je nach Anspruch eingeschränkt auf bestimmte Bereiche (**Domänen**) oder Verwendungszwecke – entweder direkt repräsentiert ist oder anhand von Schlussfolgerungen (**Inferenzen**) systematisch erschlossen werden kann. Hierbei stellen sich viele Detailfragen (z. B.: Welche Repräsentationskonstrukte entsprechen dem Ausdruck „ist ein“?), deren Beantwortung die Kenntnisse der kognitionswissenschaftlichen Disziplinen (u. a. Informatik, Linguistik, Psychologie, Philosophie) erforderlich macht.

Paradigmen

Verschiedene Sichtweisen darauf, wie sich aus solchen allgemeinen Wissensrepräsentationskonstrukten konkrete Wissensrepräsentationsstrukturen konstruieren lassen (und welche Prozesse darüber ablaufen sollen) haben zu unterschiedlichen Paradigmen der Wissensrepräsentation geführt.

Den **semantischen Netzwerken** liegt die Idee der kognitiven Vernetztheit konzeptuellen Wissens zugrunde. Ihren Ursprung hat diese Auffassung in den Arbeiten Quillians (z. B. Quillian 1968), der entsprechende Repräsentationen zur Berechnung der inhaltlichen Beziehung sprachlicher Elemente (daher: „semantische“ Netzwerke) verwendete. Den „Knoten“ („nodes“) des semantischen Netzwerks entsprechen die Konzepte, den „Kanten“ („links“) die vielfältigen Beziehungen zwischen ihnen. Entsprechend lassen sich in einem solchen Netzwerk „Nähe“ bzw. „Ferne“ von Konzepten über die Länge der Pfade verbindender Kanten verstehen und auch psychologisch relevante Prozesse wie „Aktivationsausbreitung“ definieren.

Das **Frame-Paradigma** betont den objekt-orientierten und schematischen Aspekt der Wissensrepräsentation, wonach das relevante Wissen über eine Entität direkt an ihrem Stellvertreter verfügbar ist. Zentrales Konstrukt dieses Paradigmas ist das des Frames („Rahmen“, s. Minsky 1975) als Repräsentation schematischen Wissens über Entitäten der Welt (Stühle, Kindergeburtstage etc.). Frames sind im Wesentlichen Attribut-Wert-Paare, wobei die Werte („Filler“) der Attribute („Slots“) bei Fehlen genauerer Information durch *typische* Information („per Default“) instanziert werden können. Solche Defaults können durch aktuell vorliegende Information überschrieben werden, charakterisieren aber insgesamt den *Prototyp* eines Frames. Frames dienen einerseits der Klassifikation vorliegender Information und andererseits dem Inferieren weiterer Information: da sie in hierarchischer Beziehung zueinander stehen (und somit *Taxonomien* darstellen), kann Information an untergeordnete Frames „vererbt“ werden.

Das **Logik-Paradigma** betont die Uniformität der Darstellung von Wissen (in erster Linie mit Hilfe der Prädikatenlogik erster Stufe oder Varianten davon), insbesondere auch für die Anwendbarkeit allgemeiner Inferenzmechanismen (Theorembeweiser). Sein Nachteil besteht in der Strukturarmut logischer Repräsentationen.

Das (**Produktions-**)**Regel-Paradigma** fokussiert auf den Aspekt der Steuerung des Verhaltens eines Systems durch die Anwendung von (Wenn-Dann-) Regeln (Inferenzregeln) auf jeweils aktuelle Daten.

Moderne Wissensrepräsentationssysteme wie z. B. PowerLoom® (<http://www.isi.edu/isd/LOOM/PowerLoom/index.html>) stellen oft eine Mischung aus diesen Paradigmen dar.

Struktur und Aufbau von Wissensbasen

Insbesondere den frühen semantischen Netzwerken mangelte es generell an formaler Klarheit der in ihnen verwendeten Repräsentationskonstrukte, vor allem der Kanten (vgl. den Titel von Woods (1975), „What's in a link“). Auch wenn der Einsatz der Netzwerke teilweise zu beeindruckenden Ergebnissen führte (wie die Verwendung der *konzeptuellen Dependenzstrukturen* Roger Schanks für das Textverstehen, s. Schank 1975), blieb ein wesentlicher Teil ihrer Bedeutung oft in ihrem Verarbeitungsmechanismus versteckt.

Ein abschreckendes Beispiel dafür ist das in Abb. 4.18 dargestellte Netzwerk. Hier stellt sich die Frage, welches Wissen repräsentiert sein soll: Dass alle Menschen, die Peter heißen, Popcorn essen? Oder handelt es sich um einen bestimmten Peter? Ist Popcorn notwendigerweise etwas Essbares oder so zufällig wie das Menschsein von Peter (der ja auch ein Hund sein könnte)? Findet die Handlung immer im Kino statt oder handelt es sich um eine spezifische Situation? Von welchen Kanten kann noch eine loc-Kante ausgehen?

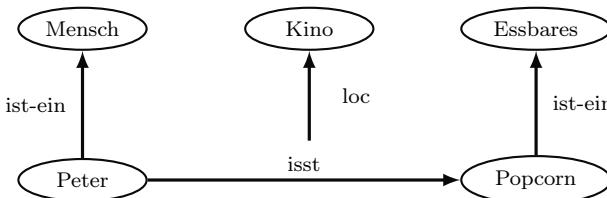


Abbildung 4.18: Beispiel für ein inadäquates semantisches Netzwerk

Nach Brachman (1979) muss stattdessen systematisch zwischen den inhaltlichen und strukturellen Aspekten von Wissensrepräsentationen unterschieden werden: Während der *Inhalt* nicht-sprachlichen Wissens auf der ***konzeptuellen Ebene*** eines Wissensrepräsentationssystems spezifiziert wird, wird dessen *Form* bzw. *Struktur* auf der ***epistemologischen Ebene*** determiniert.

Die epistemologische Ebene adressiert somit das strukturelle Inventar von Netzwerken und deren Wohlgeformtheit. Entsprechend werden hier die Kanten definiert, die die hierarchischen Beziehungen zwischen Konzepten (Subkonzept-Superkonzept, Individuenkonzept-Klassenkonzept) oder deren Eigenschaften (die „Rollen“ von Konzepten (Relationen, Attribute)), darstellen. Als Resultat dieser Überlegungen ergibt sich, dass einige der Kanten „objektiviert“ werden: z. B. vermitteln Ereigniskonzepte, denen räumliche/zeitliche Information attribuierbar sind, zwischen Nominalkonzepten.

Dieser grundlegende Unterschied zu simplen Netzwerken ist in Abb. 4.19 auf dieser Seite anhand eines KL-ONE-ähnlichen Netzwerks (s. Brachman und Schmolze 1985) veranschaulicht. Sie zeigt, dass sich selbst Rollen als Objekte auffassen lassen, die über rein strukturelle Kanten mit Konzepten verbunden sind. Dies hat zwei Vorteile: Erstens können Rollen selbst Eigenschaften haben, z. B. „v/r (value restriction)“ als Beschreibung ihres Wertebereichs, den typischen Wert oder ihre Häufigkeit. Zweitens können sie, wie die Konzepte, hierarchisch organisiert sein.

In Bezug auf das konzeptuelle Wissen muss nach Helbig (2001) zwischen dem, was es gibt und was immer wahr ist (*immanentes Wissen*), und dem, was (nur) in einer bestimmten Situation der Fall ist (*situatives Wissen*), unterschieden werden. Das immanente Wissen umfasst sowohl die notwendigen Merkmale eines Konzepts (z. B. dass Popcorn etwas Essbares ist) wie auch die das *Defaultwissen* ausmachenden (z. B. dass Menschen im Kino Popcorn essen).

Gleichzeitig muss es möglich sein, Strukturen zu bilden, die den Defaultannahmen widersprechen, u. a. um spezifische Aussagen (*Assertionen*) zu ermöglichen. Dies ist in Abb. 4.19 am Beispiel des Satzes „Peter isst im Cinestar Erdnüsse“ dargestellt. Man beachte, dass hier jedoch nicht alle relevanten Aspekte expliziert sind.

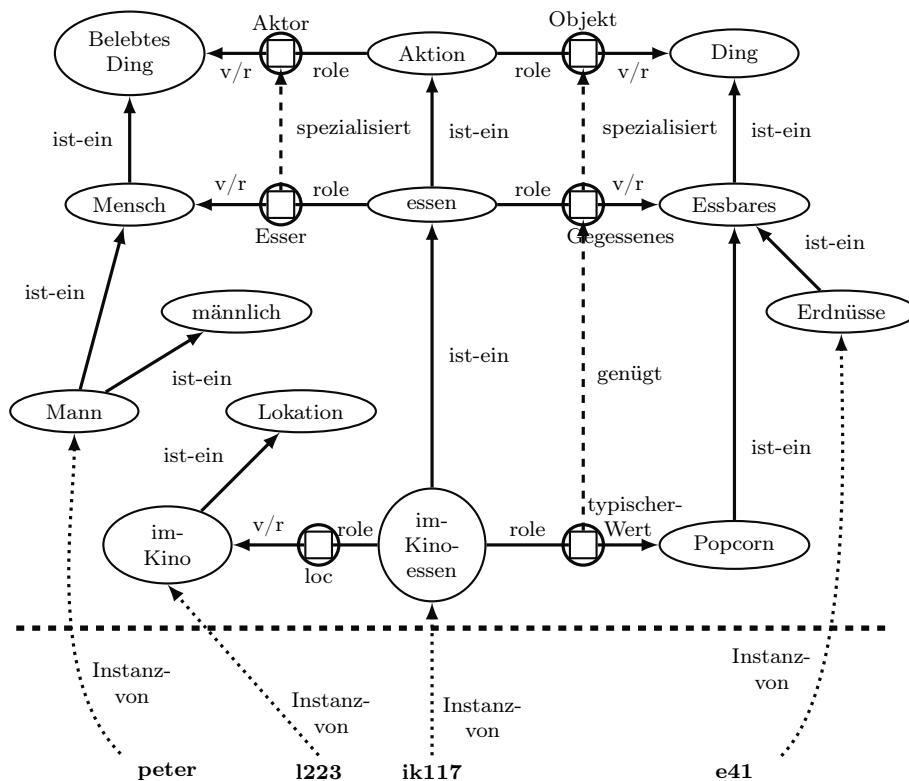


Abbildung 4.19: Elaborierte Netz-Darstellung repräsentierten Wissens

Hieraus resultiert die Unterscheidung einer terminologischen Komponente („**T-Box**“) und einer assertionalen, Faktenwissen repräsentierenden Komponente („**A-Box**“) in Frame-basierten Wissensrepräsentationssystemen wie KL-ONE (Brachman und Schmolze 1985) – eine Zweiteilung, die in Abb. 4.19 durch eine waagerechte, gestrichelte Linie dargestellt ist. Die T-Box enthält danach die Konzeptdefinitionen, die A-Box die mithilfe dieser Konzepte gebildeten Aussagen über Individuen einer gegebenen Domäne. Aus dieser Konstellation ergeben sich Informationen über Individuen als Instanzen bestimmter Konzepte sowie (hier nicht dargestellt) von Paaren als Instanzen bestimmter Rollen.

Üblicherweise werden T-Box und A-Box, ggf. zusammen mit einer Regelkomponente, als die **Wissensbasis** eines entsprechenden Systems bezeichnet. **Wissensbasierte Systeme** verfügen neben der Wissensbasis über einen **Inferenzmechanismus**, der die Wissensbasis manipulieren kann (zum Aufbau einer Wissensbasis s. Brachman, McGuiness, Patel-Schneider und Resnick 1990). Natürlichsprachliche Systeme verfügen zudem über ein **Lexikon** (mit Verweisen sprachlicher Ausdrücke auf Konzepte in der T-Box) und ein **Onomastikon** (mit Verweisen von Namen auf bestimmte Instanzen in der A-Box).

Dem Desiderat formaler Klarheit folgend gilt es seither als Maxime, Sprachen zur *Beschreibung* von Repräsentationskonstrukten mit einer entsprechend expliziten Semantik zu versehen, die einen Bezug der Konstrukte zu ihrer Interpretation in der Welt herstellt (z. B. die Interpretation von Konzepten als Mengen von Individuen und von Rollen als Mengen von Paaren). Heraus resultiert der Terminus **Beschreibungslogik** (*description logic*) für entsprechende formale Sprachen zur Spezifizierung von Wissensrepräsentationsformalismen.

Description logics (DL) sind – üblicherweise stark restriktierte – Versionen der Prädikatenlogik, mithilfe derer sich Wissensrepräsentationskonstrukte formal darstellen lassen. Abb. 4.20 zeigt einen Ausschnitt einer DL-Formalisierung von Abb. 4.19.

T – Box :
<i>Mann</i> ≡ <i>Männlich</i> \sqcap <i>Mensch</i>
<i>essen</i> ≡ <i>Aktion</i> \sqcap \forall <i>Gegessenes.Essbares</i> \sqcap \forall <i>Esser.Mensch</i>
<i>Popcorn</i> \sqsubseteq <i>Essbares</i>
...

A – Box :
<i>Mann(peter)</i>
<i>im – Kino – essen(ik117)</i>
<i>Esser(ik117, peter)</i>
<i>Gegessenes(ik117, e41)</i>
<i>Erdnüsse(e41)</i>
<i>loc(ik117, l223)</i>
...

Abbildung 4.20: T-Box und A-Box- Ausschnitte zu Abb. 4.19

Konzepte können über eine Konjunktion von Konzeptbeschreibungen *definiert* werden, wobei Konzeptbeschreibungen u. A. Konzeptnamen (wie bei der Definition von *Mann*) oder Rollenbeschreibungen (wie bei der Definition von *essen*) sein können. Die angegebenen Rollenbeschreibungen formalisieren die Wertebeschränkungen für die jeweilige Rolle. So ist *essen* definiert als eine Aktion, für deren Rolle *Gegessenes* alle Werte Instanzen von *Essbares* und für deren Rolle *Esser* alle Werte Instanzen von *Mensch* sind.

Werden Konzepte nicht definiert, so lassen sich wie bei *Popcorn Subsumptionsbeziehungen* angeben (alle Instanzen von *Popcorn* sind Instanzen von *Essbares*). Subsumptionsbeziehungen bilden das Gerüst hierarchisch organisierter Netzwerke. Daher wird beim Aufbau einer Wissensbasis durch einen Mechanismus (sog. „**classifier**“) systematisch gecheckt, ob diese Beziehungen gelten und wo ggf. ein neues Konzept in der Hierarchie angesiedelt ist. Konzept *definition* entspricht im Übrigen einer gegenseitigen Subsumption von Definiens und Definiendum.

Eine dritte Möglichkeit, Konzepte einzuführen, besteht darin, sie nicht weiter zu beschreiben, sondern sie als gegeben/primitiv zu deklarieren. In einer ontologischen Struktur führt das dazu, dass sie direkt unter dem allgemeinsten Konzeptknoten angesiedelt sind.

Ontologien

Die Konstruktion von Wissensbasen erweist sich als ein komplexes Unterfangen, das prinzipiell einen großen Spielraum für Beliebigkeit und nachfolgende Inkonsistenzen lässt. Gerade unter dem Aspekt der Wiederverwendbarkeit von Wissensbasen ist es daher wichtig, Einheitlichkeit und Konsistenz zu gewährleisten.

Die Lösung besteht darin, sich auf eine gemeinsame Sicht der Welt (sog. „Konzeptualisierung“) zu einigen und ebendiese zu formalisieren, wie es seit langem in der philosophischen Disziplin der *Ontologie* versucht wird: „*Ontology as a branch of philosophy is the science of what is, of the kinds and structures of objects, properties, events, processes and relations in every area of reality.*“ (Smith 2003, S. 155).

In der Wissensrepräsentation wird daran angelehnt unter einer **Ontologie** ein aus solchen Überlegungen resultierendes System von Konzepten (Begriffen) verstanden, das unser Wissen über die (Struktur der) Welt enthält (dabei werden auch Fragen der *kognitiven* Repräsentation von Wissen relevant, s. dazu z. B. Carstensen 2009a, Evermann 2005 und Markman 1999).

Diese in der CL, KI und verwandten Disziplinen mittlerweile weit verbreitete Auffassung von „*Ontologie*“ wurde in Gruber (1995) durch die Formulierung „*explizite Spezifikation einer gemeinsamen Konzeptualisierung*“ geprägt, d.h. als eine formale, sprachunabhängige Beschreibung einer intersubjektiven Sicht der Welt, die als gemeinsamer Kern verschiedener Wissensbasen dienen kann (s. aber Smith 2003 zu Unterschieden dieser und einer realistisch-philosophischen Auffassung von „*Ontologie*“).

Eine Ontologie, die den Oberbau von T-Boxen darstellt (daher auch: *upper-level* oder *top ontology*) beschränkt sowohl die Primitive der konzeptuellen Ebene bzgl. ihrer möglichen Interpretationen als auch die möglichen Optionen auf der epistemologischen Ebene. Kriterien für deren Wohlgeformtheit lassen sich nach Guarino (1995) aus philosophischen und linguistischen Überlegungen gewinnen. Zwischen diesen beiden Ebenen ist deshalb nach Guarino eine eigenständige **ontologische Ebene** anzunehmen.

Mittlerweile versteht man unter einer Ontologie nicht nur diesen Kern, sondern auch die übrigen Wissensrepräsentationsstrukturen, also auch bereichsspezifische (*domain ontologies*), zwischen upper-level und domain-level vermittelnde (*mid-level ontologies*) oder aufgabenspezifische (*task ontologies*) (s. hierzu auch die EuroWordNet-Architektur in Unterkapitel 4.3). In jedem Fall kommt aber der upper/mid-level ontology als wiederverwendbarer Ressource eine besondere Bedeutung zu (Beispiele sind die Suggested Upper Merged Ontology SUMO und die Mid-Level Ontology MILO, s. hierzu <http://www.ontologyportal.org>).

In einem noch weiter gefassten Sinn werden zum Teil alle Wissensressourcen eines wissensbasierten Systems – beispielsweise lexikalisch-semantische Ressourcen wie WordNet (s. Unterkapitel 4.3) – als Ontologien bezeichnet.

Von der Wissensrepräsentation zum Semantischen Web

Nach dem Aufkommen der semantischen Netzwerke (Ende der 60er Jahre) und des Frame-Paradigmas (Anfang der 70er Jahre) setzte eine „Logifizierung“ der Wissenrepräsentationsformalismen ein, aus der verschiedene (Default-)Logiken sowie die Beschreibungslogiken (auch: *terminologische Logiken*) resultieren. Die deutlich werdende „Zerbrechlichkeit“ der wissensbasierten Systeme (vor allem der Expertensysteme) führte 1984 zu dem Projekt Cyc (von „Encyclopedia“, s. Lenat und Guha 1989), in dem, ausgehend von der Entwicklung einer top ontology, eine umfassende, „common sense knowledge“ repräsentierende Wissensbasis entwickelt werden sollte (und immer noch wird). Die 90er Jahre sahen ein rasant ansteigendes Interesse an Ontologien, ihrer Erstellung und Verarbeitung (sog. „ontological engineering“), insbesondere auch unter dem Aspekt der Wiederverwendbarkeit von Wissen im oder für das WWW.

Berners-Lee et al. (2001) veranschaulichen das Problem für Computer, die Daten einer Webseite zu verstehen und entsprechend zu bearbeiten. Sie entwickeln die Vision eines **Semantic Web**, in dem die Daten des Web mit Information über ihre jeweilige Bedeutung annotiert sind. Mittlerweile existieren hierfür Vorschläge, die auf einer systematischen Schichtung (sog. „Semantic layer cake“) immer mächtigerer Beschreibungssprachen basieren. Etwa in der Mitte zwischen Webdaten-Implementation und spezifischen Anwendungen ist die Web Ontology Language (OWL, s. McGuinness und van Harmelen 2004) angesiedelt, mit der sich (onto)logische Ausdrücke der description logic darstellen lassen. Sie basiert auf dem Resource Description Format (RDF; vgl. Unterkapitel 2.5), mithilfe dessen Wissensrepräsentationskonstrukte (Konzepte, Rollen, Instanzen) als Webdaten definiert und schließlich im XML-Format für Annotierungen zur Verfügung gestellt werden können.

Eines der drängenden Probleme bei der Entwicklung des Semantic Web ist die automatisierte Erstellung von Ontologien. Zu dessen Lösung etabliert sich ein zwischen Information Retrieval, Künstlicher Intelligenz und Computerlinguistik angesiedelter Bereich namens **ontology learning and population from text**, in dem ontologisches Wissen aus den Texten von Webseiten induziert wird (s. Cimiano (2006) sowie Unterkap. 4.7 zum WWW als Ressource).

Als separate Entwicklung lässt sich auch eine Renaissance früher Wissensrepräsentationsideen für die Klassifikation und das selektive Retrieval von Web-Dokumenten unter dem Stichwort **Topic maps** beobachten: Topic maps sind das Analogon der frühen semantischen Netzwerke im Bereich des WWW. Sie bestehen aus einer abstrakten Ebene von *topics* (Themen, von denen Webseiten handeln können), die durch *associations* miteinander verknüpft sind. Insbesondere das Konzept der associations weckt allerdings die Erinnerung an die klassischen Probleme der Wissensrepräsentation (derer sich seine Erfinder möglicherweise nicht bewusst sind), weswegen Topic Maps kaum eine ernsthafte Rolle in zukünftigen Versionen des Web spielen werden.

4.6.5 Wissensrepräsentation für die CL

Probleme

Die Verwendung nicht-sprachlichen Wissens in natürlichsprachlichen Systemen ist grundsätzlich nicht-trivial, da die existierenden Ressourcen weder ausgereift noch generell mit sprachlichen Komponenten kompatibel (d.h. für natürlichsprachliche Zwecke geeignet) sind. Im Einzelnen lassen sich die folgenden Probleme nennen:

Die notwendige **Trennung nicht-sprachlicher und sprachlicher Konzepte** wird nicht immer strikt eingehalten. Zum einen werden in der Wissensmodellierung sprechende Bezeichner (d. h. sprachliche Symbole) verwendet, was zumindest potentiell (wohl aber sogar faktisch) zu einem sprachlichen und auch kulturspezifischen „Bias“ führt (wodurch insbesondere bei Interlingua-basierter Übersetzung Probleme auftreten können).

Zum anderen werden sprachliche Ressourcen wie WordNet nicht selten als Ontologien im engeren Sinne verwendet. Durch Anwendung ihrer OntoClean-Methodologie zeigen Gangemi et al. (2003), dass WordNet für diesen Zweck nicht geeignet ist, da dessen Konzeptstruktur einigen grundlegenden ontologischen Wohlgeformtheitsbedingungen nicht genügt.

Dem Leser wird nicht entgangen sein, dass diese Aspekte nur Spezialfälle der allgemeinen terminologischen Verwirrung in der Redeweise darüber sind, wie die Welt (→ ontologisch), unser Wissen über die Welt (→ konzeptuell, epistemologisch) und die sprachlichen Ausdrücke (→ terminologisch) beschaffen bzw. zu modellieren sind.

Die **Nicht-Berücksichtigung sprachlich relevanter Differenzierungen** in Ontologien bedeutet deren Unbrauchbarkeit für computerlinguistische Zwecke. Entsprechend müssen ausschließlich von spezifischem Domänenwissen abstrahierende Top-Ontologien um sprachlich relevante Konzepte erweitert werden. Hier-

zu gehört z. B. die systematische Unterscheidung (quantifizierbarer) Massen von (zählbaren) Objekten, um die Inadäquatheit von **Peter isst viel Apfel.* und **Viel Menschen isst einen Popcorn.* erklären und systematisch ausschließen zu können. Ein Vorschlag für eine entsprechende, an linguistischen Phänomenen orientierte upper-level Ontologie ist das *Generalized Upper Model* von Bateman et al. (1994).

Grundlegende Probleme der Wissensmodellierung, wie z. B. die **Bestimmung eines einheitlichen Inventars an Repräsentationsprimitiven**, sind immer noch nicht gelöst. So zeigt ein Vergleich von Wissensrepräsentationssystemen eine erschreckende Uneinheitlichkeit schon bei grundlegenden Repräsentationskonstrukten.

Ein besonderes Problem ergibt sich aus der weitgehenden **Arbitrarität konzeptueller Slots** im Frame-Paradigma. Ein Beispiel dafür findet sich im System Cyc: Es enthält tatsächlich Wissen darüber, wie man Popcorn isst, und zwar in Form eines entsprechenden ‚*EatingPopcorn*‘-Frames. Hierzu gehört nach Lenat und Guha (1989, S. 192) ein Slot ‚*bodyPartsRequiredOfPerformer*‘, für den die Werte ‚*Teeth, Mouth, Throat, Stomach, Brain*‘[!] angegeben sind.

Slots wie diese verstecken komplexe Strukturzusammenhänge hinter sprachlich eingängigen Bezeichnungen. Deren Beziehung zum restlichen Wissen (bzw. deren Semantik) ist entweder nicht vorhanden oder nur durch ein komplexes „Slot-Bookkeeping“ herzustellen. Solche Probleme haben übrigens zur Aufgabe der Frame-Basiertheit in Cyc zugunsten einer logisch verteilten Repräsentation (sog. „knowledge soup“) geführt. Wie Mahesh et al. (1996) in ihrer Analyse der NLP-Tauglichkeit von Cyc zeigen, hat jedoch die daraus resultierende Unstrukturiertheit der Repräsentation ebenfalls negative Konsequenzen für deren Verwendbarkeit in natürlichsprachlichen Systemen.

Ein immer noch bestehendes praktisches Problem ist die **mangelhafte Verfügbarkeit** (vor allem kommerziell entwickelter) vorhandener Ressourcen. Ausnahmen sind beispielsweise die Suggested Upper Merged Ontology (SUMO, s. <http://www.ontologyportal.org/>) und der frei verfügbare Teil der Ontologie aus Cyc (OpenCyc, s. <http://www.opencyc.com/>)

Perspektiven

Die Nutzung nicht-sprachlichen Wissens als computerlinguistische Ressource wird, nicht zuletzt durch den bislang ausgebliebenen Erfolg von Cyc, in der nächsten Zeit voraussichtlich vor allem pragmatisch ausgerichtet sein. Dies bedeutet in erster Linie, dass die (schnelle) Erfüllung einiger Erwartungen aufgegeben werden muss, nämlich die Erwartung einer *perfekten* Gesamt-Ontologie, einer *idealen* Interlingua, einer *alles abdeckenden* Top-level Ontologie oder einer *allumfassenden* Wissensbasis. Stattdessen werden existierende Ressourcen trotz partieller Inkompatibilitäten zusammengefügt („ontology merging“) wie z. B. bei der Entwicklung der *Sensus*-Ontologie (Knight und Luk 1994).

Aufgrund der festgestellten Lücken und Qualitätsunterschiede in der Wissensbasis von Cyc schlägt Mahesh (Mahesh und Nirenburg 1995; Mahesh 1996) im Rahmen der auf maschinelle Übersetzung ausgerichteten Entwicklung der *Mikro-*

kosmos-Ontologie einen *situierten* Ansatz der Ontologie-Konstruktion vor, bei dem die Wissensmodellierung insbesondere von der Aufgabe und dem frühen praktischen Einsatz der Wissensbasis geleitet wird. Dabei werden sprachlich relevante Unterscheidungen einerseits von vornherein und andererseits so weit wie nötig bei der Ontologie-Konstruktion berücksichtigt, so dass die Erstellung einer ontologischen Interlingua einen *approximativen* Charakter erhält. Im Gegensatz zur manuellen Erstellung von Ontologien wird die Bildung/Population von Ontologien mittlerweile allerdings automatisch anhand von aus Texten gewonnenen Informationen vorgenommen („ontology learning“, s. Cimiano 2006).

Komplementär dazu verfolgt Guarino eine formal-ontologisch gesteuerte Wissensmodellierung (Guarino 1995), indem er ontologische Meta-Eigenschaften als Constraints für wohlgeformte Ontologien verwendet. Zusammen mit den intensivierten informatischen Bestrebungen auf den Gebieten Standardisierung von Wissensrepräsentationsformalismen und Tool-Development (Ontologie-Browser etc.) lassen diese Entwicklungen erhebliche Qualitätsverbesserungen erwarten.

Angesichts der Tatsache, dass auch der Ansatz Guarinos stark von linguistischen Überlegungen geprägt und dass das ontology learning im Kern ein sprachtechnologischer Zweig ist, sind es paradoxe Weise die (Computer-)Linguisten, die Wesentliches im Bereich nicht-sprachlichen Wissens beitragen.

4.6.6 Literaturhinweise

Standardwerke der Einführung in die Wissensrepräsentation im Hinblick auf die Verarbeitung natürlicher Sprache sind Sowa (1984) und Sowa (2000). Eine lesbare Einführung mit vielen Beispielen ist Reimer (1991). Empfehlenswert im Hinblick auf Aktualität, Umfang und Detailreichtum sprachlich orientierter Wissensrepräsentation ist Helbig (2001). Interessierte finden die klassischen Papiere zur Wissensrepräsentation in Brachman und Levesque (1985) und eine umfassende Einführung in die Wissensrepräsentation und -verarbeitung aus Sicht der Künstlichen Intelligenz in Brachman und Levesque (2004). Einen Einstieg in das Thema „description logics“ liefert Baader et al. (2003) sowie die Webseite <http://dl.kr.org/>.

John Bateman (<http://www-user.uni-bremen.de/~bateman/>) bietet eine Sammlung relevanter Information zu Ontologien und weiteren Aspekten der Wissensrepräsentation an, die insbesondere im Hinblick auf die Entwicklung von Grundlagen für das Semantic Web relevant ist (s. dazu Hitzler et al. 2007, <http://www.w3.org/RDF/FAQ> sowie auch Unterkapitel 4.7). Verschiedenste Perspektiven auf den Bereich Semantic Web (und Ontologien) finden sich in Pellegrini und Blumauer (2006). Aus (computer)linguistischer Perspektive besonders interessant ist die Ontologie DOLCE („Descriptive Ontology for Linguistic and Cognitive Engineering“, s. <http://www.loa-cnr.it/DOLCE.html>).

Die praktische computerlinguistische Verwendung solcher Ontologien wird z. B. in Oberle et al. (2007) präsentiert. Nirenburg und Raskin (2004) enthält eine umfassende computerlinguistische Darstellung der Rolle von Ontologien für die Computerlinguistik und Sprachtechnologie.

4.7 Das World Wide Web als computerlinguistische Ressource

Iryna Gurevych

4.7.1 Einleitung

Das *World Wide Web* (WWW) hat sich in den letzten Jahren einerseits zur wichtigsten Informations- und Kommunikationsstruktur und andererseits zur wichtigsten computerlinguistischen Ressource entwickelt. Durch Soziale Software ist für die Computerlinguistik ein benutzerdefiniertes semantisches Tagging-System von bisher nicht dagewesener Größe entstanden. Diese Entwicklung birgt das Potential, den Wissensakquisitionsproblemen in der Computerlinguistik Abhilfe zu schaffen. Zum einen handelt es sich beim Web um einen *enormen und reichen Datenbestand*. Zum anderen können aus diesem Datenbestand aufgaben-spezifische Korpora für unterschiedliche Sprachen, Domänen, Textsorten, usw. gewonnen werden, um die Durchführung von computerlinguistischen Untersuchungen zu ermöglichen. Eine besondere Bedeutung spielen dabei die benutzer-generierte Inhalte. Mit Benutzer-Tags ausgezeichnet, bilden sie die sogenannten **Folksonomien**. Sie beinhalten sehr wertvolle semantische Informationen, die mit computerlinguistischen Methoden weiter analysiert und erschlossen werden können. Das resultierende lexikalische und semantische Wissen sowie das Welt-wissen kann in umgekehrter Richtung den computerlinguistischen Algorithmen zugeführt werden, um neue Anwendungen, z. B. im Bereich textbasierten Infor-mationsmanagements, zu ermöglichen.

4.7.2 Aspekte des Web als Korpus

Beim Web handelt es sich um ein **multilinguales Korpus**. Laut einer Unter-suchung von Xu (2000) waren im Jahr 2000 71% aller Webseiten, die von der Suchmaschine Excite indiziert wurden, auf Englisch verfasst, gefolgt von Japa-nisch (6,8%), Deutsch (5,1%), Französisch (1,8%), Chinesisch (1,5%), Spanisch (1,1%), Italienisch (0,9%) und Schwedisch (0,7%). 2002 waren laut einer ande-ren Untersuchung von Ebbertz (2002) die Sprachen wie folgt verteilt: Englisch 56,4%, Deutsch 7,7%, Japanisch 4,9%, Spanisch 3,0%, Französisch 5,6%, Chine-sisch 2,4% und Italienisch 2% .

Grundsätzlich können zwei Vorgehensweisen bei der Nutzung des WWW als computerlinguistisches Korpus unterschieden werden: (i) über Programmier-schnittstellen gängiger Suchmaschinen kann auf statistische Informationen, bei-spielsweise die *Anzahl von Treffern*, zugegriffen werden, und (ii) die Program-mierschnittstellen können eingesetzt werden, um *aufgabenspezifische Korpora* nach vorgegebenen Anforderungen zu erstellen. Neben vielen in der Einführung aufgezählten Vorteilen sind bei der erstgenannten Vorgehensweise eine Reihe von

Herausforderungen zu berücksichtigen. Insbesondere sind experimentelle Ergebnisse auf der Grundlage des WWW *unzuverlässig* und oft *nicht reproduzierbar*. Sie können je nach Suchmaschine und je nach Ausführungszeit der Anfrage stark schwanken. Die genaue Zusammensetzung des Korpus, das von Suchmaschinen erfasst und indiziert wird, ist nicht bekannt. Das erschwert die Interpretation der Ergebnisse. Ebenso unbekannt ist, wie *vollständig* die Ergebnisse sind, da die Anzahl von korrekten Suchergebnissen unbekannt ist. Aus diesen Gründen kann es für bestimmte Einsatzszenarien sinnvoll sein, die zweite Vorgehensweise zu wählen. Diese hat zum einen den Vorteil, dass zuverlässige Statistiken auf konstanter Datengrundlage berechnet werden können, zum anderen hat der Korpusersteller zumindest eingeschränkt Kontrolle über die zugrundeliegenden Daten. Der Einsatz des Korpus kann dann je nach Domäne und Aufgabenstellung fokussierter erfolgen.

Bei der Nutzung des WWW als Korpus müssen einige Problemfelder addresiert werden: Zum einen werden erhebliche *Speicherplatz- und Rechenkapazitäten* benötigt. Texte sind oft mit html-Code und anderen irrelevanten Inhalten wie sogenannten *boilerplates*, d. h. Navigationsmenüs, Werbung, usw. vermischt. Es ist wichtig, *Duplikate* in den Daten zu identifizieren und zu entfernen, um Bias zu vermeiden. Um ein monolinguales Korpus zusammenzustellen, müssen Webseiten in der Zielsprache zunächst *automatisch identifiziert* werden. Webseiten sind in der Regel nur *teilweise* mit Metainformationen versehen, und diese sind oft *uneinheitlich*. Auch *Autorenschaft* von Texten kann oft nicht hergestellt werden. Die Inhalte sind stark *diversifiziert* und von *unterschiedlicher sprachlicher Qualität*, die Daten müssen teilweise von *Spam* bereinigt werden. Je nach Art der Entstehung, z. B. Emails, Blog-Einträge oder Wikis, liegen verschiedene *Sprachregister* mit speziellen Eigenschaften vor. Es ist wünschenswert, die Webseiten mit Metadaten, beispielsweise in Bezug auf die *Genre* oder die *Themen*, automatisch zu annotieren. Die Nutzung von den gesammelten Daten sowie von den damit verknüpften Metainformationen und Medien wie Bildern für die computerlinguistische Forschung bedarf oft einer *rechtlichen Absicherung*, die in der Praxis problematisch ist. Nichtsdestoweniger sind die Webdaten im Hinblick auf ihre *Aktualität* sehr wertvoll und ermöglichen z. B. die Erforschung von neuen Wörtern und sprachlichen Phänomenen. Die Weiterentwicklung von Methoden und Standards für die Erstellung webbasierter Korpora ist insofern sehr berechtigt.

Keller und Lapata (2003) beschreiben einen Ansatz, bei dem die Frequenzen für Bigramme auf Grundlage des WWW approximiert werden, die in einem herkömmlichen Korpus nicht vorkommen. Sie erhalten die Frequenzen für ‘Adjektiv-Nomen’, ‘Nomen-Nomen’ und ‘Verb-Objekt’-Bigramme aus dem Web via Anfragen an eine Suchmaschine. Die Evaluierung dieser Methode zeigt u. a., dass (i) die webbasierten Frequenzen eine hohe Korrelation mit den Korpusbasierten Frequenzen aufweisen, (ii) die webbasierten Frequenzen zuverlässig mit den menschlichen Bewertungen korrelieren, und (iii) die webbasierten Frequenzen als gute Indikatoren für Disambiguierungsaufgaben dienen können.

Lapata und Keller (2005) beschreiben eine systematische Untersuchung der Nützlichkeit von webbasierten Modellen für eine Reihe von computerlinguisti-

schen Aufgaben, indem (i) Syntax und Semantik, (ii) Generierung und Analyse, und (iii) ein breites Spektrum an N-Grammen und Wortarten einbezogen sind. Für eine Mehrzahl der Aufgaben weisen einfache, unüberwachte Modelle der N-Gramme eine bessere Performanz auf, wenn sie auf den WWW-Daten und nicht auf einem Standard-Korpus berechnet werden. Eine weitere Verbesserung kann in einigen Fällen durch die Kombination von webbasierten und Korpus-basierten Frequenzen mittels Back-off und Interpolierungstechniken erzielt werden.

Ein weiterer Ansatz für die Nutzung des WWW als Korpus ist die Sammlung von sogenannten *Text-Snippets*, also Textfragmenten. Die verfügbaren Programmierschnittstellen wie die Google API erlauben, den Kontext von Suchwörtern zu erhalten. So können Suchbegriff-zentrierte, aufgabenspezifische Korpora aufgebaut werden. Des Weiteren können die Webseiten komplett heruntergeladen werden. Baroni und Bernardini (2004) stellen ein System namens **BootCaT** vor, welches dazu dient, themenspezifische Web-Korpora zu erstellen. Der Benutzer legt im ersten Schritt die Suchbegriffe fest. Dann werden die Webseiten gesammelt, die die Kombination von diesen Suchbegriffen für eine gegebene Domäne enthalten. Anschließend werden Kollokationsstatistiken erstellt, um z. B. domänen spezifische Begriffe zu finden (dabei werden die gesammelten Webseiten mit einem allgemeinen Korpus verglichen).

Das **WaCky-System**¹⁰ (Web as Corpus kool ynitiativ) bietet verschiedene Werkzeuge und Programmierschnittstellen, die einem Nutzer ermöglichen, einen Teil des Webs zu crawlern, zu verarbeiten, zu indexieren und darauf zu suchen. Das mit Hilfe dieses Systems erstellte Korpus für Deutsch (deWaC) mit 1,5 Milliarden Token und Italienisch (itWaC) mit 2 Milliarden Token stehen mit annotierten Wortarten und Lemmata zur Verfügung. Das englische Korpus besteht aus über mehr als 2 Milliarden Token und gehört derzeit zu den größten frei verfügbaren linguistischen Ressourcen im Web (Ferraresi et al. 2008).

4.7.3 Sozio-Semantisches Web

Die Entwicklungen im WWW-Bereich waren in den letzten Jahren durch die sogenannten Sozio-Semantischen Technologien gekennzeichnet (Gruber 2008). **Sozio-Semantisches Web** bezeichnet demnach die Vereinigung von umfangreichen Wissensdatenbanken, die von der Internet-Gemeinschaft kollaborativ erstellt werden, mit der Ausdrucksmächtigkeit und den Inferenzmechanismen des **Semantic Web** (s. dazu auch Unterkap. 4.6). Diese vereinigte Vision soll zu neuartigen Webanwendungen führen, die die in den Webdaten implizit repräsentierten semantischen Relationen automatisch identifizieren und daraus ein Netzwerk mit strukturiertem Wissen erstellen.

In der Computerlinguistik wurden im Bereich der lexikalischen Semantik und der semantischen Erschließung von Inhalten wichtige Schritte in Richtung des Sozio-Semantischen Webs gemacht. Insbesondere verschiebte sich der Fokus von herkömmlichen manuell erstellten Ressourcen, z. B. Wortnetzen, zur automatischen Erschließung und Nutzung des Wissens in den sogenannten **kollaborati-**

¹⁰<http://wacky.sslmit.unibo.it/>

Quelle	Art von lexikalisch-semantischen Informationen
Artikel	
- Erster Absatz	Definition
- Volltext	Beschreibung der Bedeutung; verwandte Begriffe; Übersetzungen
- Weiterleitungen	Synonyme; (teilweise inkorrekte) Schreibvarianten; Abkürzungen
- Titel	Eigennamen; domänen spezifische Begriffe und ihre Bedeutungen
Artikel-Links	
- Kontextfenster	verwandte oder zusammen vorkommende Begriffe;
- Label	Synonyme; Schreibvarianten; verwandte Begriffe
- Ziel-Artikel	verwandte Begriffe
Kategorien	
- dort beinhaltete Artikel	semantisch verwandte Begriffe (meist Hyponyme)
- Hierarchie	semantische Relationen, z. B. Hyponyme, Meronyme
Disambiguierungsseiten	
- Artikel-Links	häufigste Bedeutung, Bedeutungsvokabular

Tabelle 4.2: Beispiele der lexikalisch-semantischen Informationen in Wikipedia.

ven Wissensdatenbanken. Letztere entstehen als Folge freiwilliger Benutzerbeiträge im Sozialen Web, also *bottom-up*. Für den Einsatz als computerlinguistische Ressource müssen solche Wissensquellen speziell aufbereitet werden, da sie nicht zu diesem Zweck geschaffen wurden und die Informationen dort meistens nicht geeignet strukturiert sind.

Im Folgenden werden wir uns mit zwei spezifischen Instanzen von kollaborativen Wissensdatenbanken beschäftigen: der multilingualen freien Internet-Enzyklopädie **Wikipedia** und dem freien Internet-Wörterbuch **Wiktionary**. Wikipedia und Wiktionary wurden in jüngster Zeit als besonders vielversprechende Ressourcen identifiziert. Analog zum *Web-Mining* (Chakrabarti 2002) bezeichnen wir die Analyse von Wiki-basierten Wissensdatenbanken als **Wiki-Mining** und unterteilen sie in die folgenden drei Bereiche, die in absteigender Relevanz für die Computerlinguistik aufgeführt werden: (i) Mining von *Inhalten*, (ii) Mining von *Struktur*, und (iii) Mining von *Nutzungsdaten*.

Wikipedia ist eine durch Benutzer erstellte elektronische Enzyklopädie, die eine intensive Verlinkung der Inhalte aufweist. Zesch et al. (2007) analysieren die Inhalte und die Struktur von Wikipedia und identifizieren dort verschiedene Quellen lexikalisch-semantischer Informationen, wie in Tabelle 4.2 dargestellt. Infolge existierender Gestaltungsrichtlinien für Autoren beinhaltet Wikipedia überwiegend Begriffe von enzyklopädischem Interesse. Größtenteils handelt es sich hierbei um Nomen sowie relativ wenige Adjektive und Verben, von denen in den meisten Fällen auf die Nomen mittels der sogenannten Weiterleitungen (Engl. *redirects*) verwiesen wird, z. B. vom Verb „sehen“ auf den Mehrwortbegriff „visuelle Wahrnehmung“.

Der erste Absatz eines Wikipedia-Artikels beinhaltet typischerweise eine kurze Definition des im Artikel beschriebenen Begriffs. Im Volltext eines Artikels sind zahlreiche *verwandte Begriffe* enthalten, die die Bedeutung des Begriffs weiter präzisieren. Zum Teil sind auch *Übersetzungen* des Begriffs mit Links zu den entsprechenden Wikipedias in anderen Sprachen enthalten. Somit stellt Wikipedia eine vielversprechende Ressource für *multilinguale* computerlinguistische Anwendungen dar.

Eine weitere Quelle der lexikalisch-semantischen Relationen in Wikipedia sind die *Links*, die verschiedene Artikel in Wikipedia untereinander verbinden. Ein Link deutet typischerweise auf eine semantische Relation zwischen den beiden verlinkten Begriffen hin. Der Typ dieser Relation sowie ihre Stärke sind jedoch nicht explizit kodiert und müssen ggf. mit computerlinguistischen Methoden automatisch erschlossen werden (Krötzsch et al. 2005). Zusammen bilden alle verlinkten Begriffe und die Links einen **Artikel-Graphen**. Jeder Link hat zusätzlich ein *Label*, dessen Wortlaut sich vom verlinkten Begriff durchaus unterscheiden kann. Beispielsweise haben viele Begriffe, die auf den Artikel „Deutschland“ verweisen, das Label „Bundesrepublik Deutschland“. Infolgedessen können die Labels als Quelle für *Synonyme*, *Schreibvarianten* oder andere *semantisch verwandte Begriffe* genutzt werden. Aus dem **Kontextfenster** um das Label herum können mittels computerlinguistischer Techniken weitere verwandte Begriffe gewonnen werden.

Das Kategoriensystem in Wikipedia resultiert daraus, dass jeder Artikel eine beliebige Anzahl an **semantischen Tags**, also *Kategorien* von Benutzern bekommen kann. Insofern ist das Kategoriensystem eine *Folksonomie*. Jede Kategorie kann eine beliebige Anzahl an Artikeln zugewiesen bekommen. Sie kann auch *Unterkategorien* haben, die typischerweise über die *Hyponymie* oder *Meronymie* mit der *Oberkategorie* verknüpft sind. Die Kategorie „Fahrzeug“ hat beispielsweise Unterkategorien wie „Luftfahrzeug“ oder „Wasserfahrzeug“. Insofern bildet das Kategoriensystem von Wikipedia eine Art **Thesaurus**.

Polyseme, also mehrdeutige Wörter, sind in Wikipedia mittels der Disambiguierungsseiten repräsentiert. Eine **Disambiguierungsseite** listet alle Artikel auf, die für einen mehrdeutigen Begriff vorhanden sind. Da die Bezeichnung jedes Artikels eindeutig sein muss, werden die Artikel für polyseme Begriffe meistens unterschieden, indem jeder Artikel mit dem disambiguierenden Begriff in Klammern versehen wird, z. B. „Wald“ und „Wald (Graphentheorie)“. Der Artikel ohne Disambiguierungstag beschreibt zumeist die *häufigste Bedeutung* eines Begriffs. Alle aufgelisteten Bedeutungen bilden ein **Bedeutungsvokabular** für den gegebenen Begriff.

Wiktionary wird von Nutzern als multilinguales web-basiertes Wörterbuch und Thesaurus im Web kollaborativ erstellt und ist komplementär zur Online-Enzyklopädie Wikipedia. Zesch et al. (2008a) stellen erstmalig eine systematische Analyse von Wiktionary als computerlinguistische Ressource vor. Im Unterschied zur Wikipedia zielt Wiktionary demnach eher auf allgemeines Vokabular ab. Es deckt mehrere Wortarten ab und verzichtet auf detaillierte faktische Informationen enzyklopädischen Charakters, die in Wikipedia zu finden sind.

Im Oktober 2008 beinhaltete Wiktionary etwa 3,5 Mil. Einträge in 272 sprachspezifischen Editionen. Jede solche sprachspezifische Wiktionary-Edition beinhaltet auch Einträge für fremdsprachliche Begriffe. Folglich stellt sie ein **multilinguales Wörterbuch** mit einem substanzialen Anteil an Einträgen in Fremdsprachen dar. Das englische Wiktionary beinhaltet beispielsweise den deutschen Eintrag „*Haus*“, der mit dem englischen Eintrag „*house*“ verknüpft ist. Die Größe von kollaborativ erstellten Ressourcen hängt von der Größe und dem Engagement der Internet-Gemeinde ab, die zum Projekt beiträgt. Die englische Wiktionary-Edition, die am 12. Dezember 2002 ins Leben gerufen wurde, ist die älteste, aber nicht die größte (über 900.000 Einträge im Februar 2008). Die größte Wiktionary-Edition ist die Französische, die ein Jahr später gestartet wurde (über 923.000 Einträge im Februar 2008).

Einträge in Wiktionary beinhalten ein breites **Spektrum an lexikalischen und semantischen Informationen** wie *Wortart*, *Wortbedeutung*, *Gloss*, *Etimologie*, *Aussprache*, *Deklination*, *Beispiele*, *Zitate*, *Übersetzungen*, *Kollokationen*, *abgeleitete Begriffe* und *Hinweise zum Sprachgebrauch*. Ebenso enthalten sind lexikalisch oder semantisch verwandte Begriffe verschiedener Art, wie *Synonyme*, *Antonyme*, *Hyperonyme* und *Hyponyme*. Darüber hinaus beinhaltet Wiktionary eine beeindruckende Menge an Informationen, die in klassischen Wissensdatenbanken nicht immer vorhanden sind. Dazu zählen *Komposita*, *Abkürzungen*, *Akronyme* und *Namensabkürzungen*, verbreitete falsche *Schreibvarianten* (z. B. Engl. *basicly* – *basically*), vereinfachte *Schreibvarianten* (z. B. engl. *thru* – *through*), *Kontraktionen* (z. B. engl. *oſ* – *of*), *Sprichwörter* (z. B. engl. *no pain, no gain*), umstrittene *Wortverwendungen* (z. B. engl. *irregardless* – *irrespective or regardless*), *Protologismen* (z. B. engl. *iPodian*), *Onomatopoeia* (z. B. engl. *grr*), und sogar *umgangssprachliche Formen* oder *Slang*. Die meisten solchen Relationen sind in Wiktionary explizit kodiert. Dies ist ein prinzipieller Unterschied zu Wikipedia, wo die Art der semantischen Relationen zwischen Begriffen meistens mittels spezieller Verfahren inferiert werden muss. Des Weiteren muss berücksichtigt werden, dass sprachspezifische Wiktionary-Editionen uneinheitlich strukturiert sind. Z. B. enthält das deutsche Wiktionary im Unterschied zum englischen Wiktionary charakteristische Wortkombinationen, jedoch keine Zitate, die in der englischen Version sehr wohl vorhanden sind.

Ähnlich wie in Wikipedia, sind Wiktionary-Einträge zusätzlich mit **Kategorien** versehen. Schließlich sind die Einträge massiv mit anderen Einträgen verlinkt, sowohl innerhalb einer sprachspezifischen Wiktionary-Edition als auch sprachenübergreifend. Die Links verweisen zusätzlich auf weitere externe Wissensdatenbanken oder webbasierte Wörterbücher.

Programmatischer Zugriff auf Wikipedia und Wiktionary Die Nutzung von Wikipedia und Wiktionary in computerlinguistischen Anwendungen bedarf effizienter Methoden für den strukturierten Zugriff auf die dort enthaltenen Informationen. Zesch et al. (2008a) beschreiben eine Reihe von spezialisierten Werkzeugen für den Zugriff auf Wikipedia und stellen einen optimierten Ansatz vor, bei dem die Inhalte von Wikipedia und Wiktionary zunächst in eine Datenbank importiert werden. So können spezielle Funktionalitäten von Daten-

banken, z. B. eine effiziente Indexierung von Inhalten, voll ausgenutzt werden. Die für die computerlinguistischen Anwendungen relevanten Informationen, wie Links oder Kategorien, werden explizit auf ein Datenbankschema abgebildet. Das ermöglicht einen verbesserten Zugriff auf diese Informationen in den darauf aufbauenden computerlinguistischen Anwendungen. Die Java-basierten Programmierschnittstellen **JWPL** (Zesch 2008) und **JWKTL** (Müller 2008) sind für die nicht-kommerzielle Forschung frei verfügbar.

4.7.4 Sprachverarbeitungsanwendungen mit Nutzung des World Wide Web als Ressource

Lapata und Keller (2005) geben einen Überblick darüber, in welchen computerlinguistischen Anwendungen das Web als Ressource eingesetzt wurde. Dazu zählen beispielsweise maschinelle Übersetzung, Entdeckung von semantischen Relationen, Disambiguierung von Wortlesarten und die Beantwortung natürlichsprachlicher Fragen. In der **maschinellen Übersetzung** dient das Web als Quelle bilingualer Korpora sowie zur Nachbearbeitung von Übersetzungskandidaten. Andere Arbeiten **entdecken semantische Relationen** wie Hyponymie, Ähnlichkeit, Antonymie oder logische Folgerung mittels lexikalisch-semantischer Muster als Suchanfragen.

Bisherige Anwendungen von Wikipedia und Wiktionary in der computerlinguistischer Forschung sind exemplarisch von Zesch et al. (2008a) beschrieben. So setzen Gabrilovich und Markovitch (2006) Wikipedia für die Aufgabe der **automatischen Textklassifikation** ein. Ruiz-Casado et al. (2005) befassen sich mit **automatischer Informationsextraktion** und beschreiben einen Ansatz, um Wikipedia-Einträge automatisch mit Konzepten in einer Ontologie oder einem lexikalisch-semantischen Wortnetz zu verknüpfen. Ahn et al. (2004) verwenden das Wissen aus Wikipedia im Rahmen des *TREC 2004 Question Answering* Wettbewerbs sowohl als eine Quelle für Antworten auf faktische Fragen. Es existieren dagegen nur noch wenige Arbeiten, in denen Wiktionary als eine lexikalisch-semantische Wissensdatenbank verwendet wird. Chesley et al. (2006) verwenden Adjektive aus Wiktionary für eine Analyse der **Orientierung von Meinungen** in Blogs. Eine weitere Arbeit, die Wiktionary verwendet, ist im Bereich der **diachronischen Phonologie** (Bouchard et al. 2007). Zesch et al. (2008b) verwenden sowohl Wikipedia als auch Wiktionary für die Berechnung der **semantischen Verwandtschaft** zwischen zwei Wörtern. Schließlich wurde das kombinierte Wissen aus Wikipedia und Wiktionary in jüngster Zeit für die Verbesserung **natürlichsprachlicher Informationsrecherche** eingesetzt (Müller und Gurevych 2008).

4.7.5 Computerlinguistik und Sprachtechnologie für das Web

Das vorliegende Unterkapitel fokussierte bisher das Potenzial des World Wide Web als computerlinguistische Ressource. Mit dem Wachstum des WWW werden jedoch computerlinguistische Methoden und Werkzeuge zu einer unabdingbaren Voraussetzung, um dem Benutzer einen *effizienten Umgang* mit explosionsartig

anwachsenden Mengen an Informationen zu ermöglichen. Die Globalisierung der Informationsflut führte mittlerweile dazu, dass **Suchmaschinen** zu einer Grundsatztechnologie geworden sind. Die primäre Aufgabe von Suchmaschinen ist es, die im Web vorhandenen Dokumente nach ihrer Relevanz zur Benutzeranfrage zu ordnen. Für ein optimales Suchergebnis ist es jedoch notwendig, *Informationen über den Benutzer* beim Ranking zu berücksichtigen. Solche Informationen können wiederum mit computerlinguistischen Methoden aus Benutzer-generierten Inhalten gewonnen werden. Eine weitere wichtige Technik ist die Bereinigung der im Retrieval eingesetzten zugrundeliegenden Dokumentenbasis. Hier ist es wichtig, **Spam und Duplikate zu identifizieren** und zu entfernen. Mit der Verbreitung von Sozialer Software hat die Aufgabe der **Qualitätsbewertung der Inhalte** stark an Bedeutung gewonnen. Da es keine redaktionelle Kontrolle über die Inhalte im Web gibt, müssen automatische Verfahren entwickelt werden, um vertrauenswürdige Informationen hoher Qualität dem Benutzer vorrangig anzubieten. Eine weitere Herausforderung, bei deren Bewältigung computerlinguistische Verfahren eine Schlüsseltechnologie darstellen, ist die **kontextbezogene Aufbereitung** und die **Präsentation der Informationen** für den Benutzer. Insbesondere mit starker Zunahme von Technologien des *Ubiquitous Computing* (Mühlhäuser und Gurevych 2007) beim Zugriff auf sprachliche Informationen müssen computerlinguistische Verfahren für die **automatische Zusammenfassung** im Hinblick auf verschiedene Geräte, Formate, Inhalte und weitere Arten von Präsentationskontexten weiterentwickelt und optimiert werden.

Zusammenfassend kann festgehalten werden, dass das *Verhältnis zwischen dem WWW und der Computerlinguistik dualer Natur* ist. Die Computerlinguistik profitierte enorm vom **WWW als Ressource**, indem (i) die dort abrufbaren Informationen als ein einzigartiges, multilinguales Korpus für computerlinguistische Verfahren eingesetzt werden, und (ii) die dort kollaborativ entstehenden Wissensdatenbanken, wie z. B. Wikipedia und Wiktionary, als semantisch strukturierte und teilweise ausgezeichnete Korpora anstelle von konventionellen lexikalisch-semantischen Ressourcen und Korpora eingesetzt werden. Andererseits stellt das **WWW ein äußerst attraktives Anwendungsgebiet** für mehrere zentrale Verfahren der angewandten Computerlinguistik und der Sprachtechnologie mit einem enormen wissenschaftlichen und wirtschaftlichen Potenzial dar. Dies sichert der Computerlinguistik als Forschungsgebiet eine Schlüsselrolle in Gesellschaft und Politik.

4.7.6 Literaturhinweise

Umfangreiche weiterführende Informationen, u. a. die Links zu den Online-Proceedings der einschlägigen Workshops sowie zur Software können auf WAC (2008) und SIGWAC (2008) abgerufen werden. Vor zwei Jahren stellte Google eine Kollektion mit den aus dem Web (ca. 1 Billion Token) gewonnenen N-Grammen über das *Linguistic Data Consortium* zur Verfügung. Das Material zur Nutzung von Wikipedia als computerlinguistische Ressource ist ebenso

im Web zu finden¹¹ sowie in den Online-Proceedings (Bunescu et al. 2008). Die Webseite vom *Ubiquitous Knowledge Processing Lab* enthält einige wichtige Publikationen zu kollaborativen Wissensdatenbanken in der Computerlinguistik (Zesch et al. 2007; Zesch et al. 2008a; Zesch et al. 2008b) sowie die dazugehörige Software (Zesch 2008; Müller 2008).

¹¹http://en.wikipedia.org/wiki/Wikipedia:Wikipedia_in_academic_studies,
[http://en.wikipedia.org/wiki/Wikipedia_as_an_academic_source](http://en.wikipedia.org/wiki/Wikipedia:Wikipedia_as_an_academic_source),
http://meta.wikimedia.org/wiki/Wiki_Research_Bibliography

5 Anwendungen

Kapitelherausgeber: Kai-Uwe Carstensen

Im Grunde stand am Anfang der Computerlinguistik die Anwendung: Die ersten elektronischen Rechner wurden im Wesentlichen dazu hergestellt, um bestimmte Codes (nichts anderes als Geheim-„Sprachen“) zu entschlüsseln, und auch ambitionierte Entwicklungen zu „klassischen“ Themen der Computerlinguistik wie dem der **maschinellen Übersetzung** natürlicher Sprachen fanden statt, bevor die eigentlichen theoretischen und methodischen Grundlagen (oder die Disziplin an sich) geschaffen waren. Nicht von ungefähr führte dies in den 60er Jahren zu dem im ALPAC-Report manifestierten Vertrauensbruch in die damalige Sprachverarbeitung (vgl. auch Unterkapitel 1.2). Jahrelang existierte die mittlerweile gegründete Computerlinguistik danach im Wesentlichen nur im universitären Bereich, gehemmt durch die frappante Diskrepanz zwischen Anspruch (automatisches Sprachverstehen, -produzieren und -übersetzen) und Realität (fehlende Theorien, mangelhafte Methoden, unzureichende Rechnerperformanz).

Dieses Szenario hat sich in den letzten Jahren vollständig geändert: Computerlinguistik ist mittlerweile eine etablierte Disziplin, in der die theoretischen und methodischen Voraussetzungen für ihre praktischen Anwendungen im Rahmen einer wachsenden **Sprachtechnologie** geschaffen werden. Ihre Absolventen finden einen Arbeitsmarkt vor, der —jenseits des universitären Bereichs— wesentlich durch die Entwicklung realistischer Anwendungen geprägt ist. Möglich wurde dies vor allem durch die rasante technische Entwicklung und durch die sozioökonomische Veränderung in Richtung einer Informationsgesellschaft: Sprache und Text (als Träger von „Information“) haben an Wertigkeit zugenommen, und ihre Verarbeitung —der „Kern“ der Computerlinguistik— ist einer der Grundpfeiler der **Informationstechnologie**; PCs und das WWW sind in das Zentrum des persönlichen und gesellschaftlichen Lebens gerückt, und somit auch die Aufgaben, die in zunehmendem Maße von computerlinguistischen bzw. sprachtechnologischen Anwendungen übernommen werden. Diese Anwendungen tragen dazu bei, die Benutzerfreundlichkeit meist komplexer Systeme zu erhöhen und werden daher mit Sicherheit an Bedeutung innerhalb der Informationstechnologie gewinnen. Anstelle des Verfolgens (noch) illusionärer Ziele (wie dem sprechenden und verstehenden Computer) ist folgerichtig das pragmatische Bestreben getreten, auf einem kontinuierlich wachsenden theoretischen und methodischen Fundament effektive Verfahren und effiziente Algorithmen zu entwickeln, um unter Ausnutzung vorhandener oder noch zu erstellender Informationsressourcen dem Bedarf des Benutzers in einer Informationsgesellschaft gerecht zu werden. Aus diesem Bedarf (d.h. z.B. dem schnellen Zugang zu sowie

der angemessenen Selektion und adäquaten Darstellung von relevanter Information) ergeben sich vielfältige Anwendungsgebiete für die Computerlinguistik.

Anwendungen der Computerlinguistik lassen sich in erster Linie entlang eines Spektrums der funktionalen Komplexität unterscheiden, das von der Bewältigung grundlegender — aber nicht unbedingt trivialer — Aufgaben im Umgang mit sprachlichen Daten (dem klassischen Gebiet der „Linguistischen Datenverarbeitung“) bis hin zur immer noch fiktiven Multifunktionalität domänenunabhängiger natürlichsprachlicher Systeme reicht. So finden sich an dem einen Ende des Spektrums vor allem die automatische Erstellung computerlinguistischer Ressourcen (siehe Kapitel 4), Programme zur **Rechtschreibkorrektur** sowie die **computergestützte Lexikografie und Terminologie**. Am anderen Ende ist die **maschinelle Übersetzung** angesiedelt, ebenso wie **angewandte natürlichsprachliche Generierungs- und Auskunftssysteme**, in denen neben den Aspekten der Sprachverarbeitung auch die der Wissensrepräsentation und -verarbeitung (s. Unterkap. 4.6) eine wesentliche Rolle spielen.

Weiterhin lassen sich Anwendungen anhand genereller Struktur- bzw. Architekturgesichtspunkte differenzieren. Handelt es sich um ein vollständiges, wenn auch thematisch eingeschränktes, Gesamtsystem (z.B. ein **Dialogsystem**)? Besteht die Anwendung aus einer idealerweise modular verwendbaren Systemkomponente (z. B. solchen zur **Sprachein- und -ausgabe**)? Oder ist sie durch eine spezifische Aufgabe definiert und im Wesentlichen durch die Verwendung computerlinguistischer Methoden charakterisiert (z.B. **Text-basiertes Informationsmanagement**, **computergestützte Übersetzung**)?

Soweit eine bestimmte Input-/Output-Modalität (gesprochene vs. geschriebene Sprache) primär für eine Anwendung relevant ist, erlaubt sie außerdem eine Zuordnung zu den groben Teilebenen der maschinellen Sprachverarbeitung im Rahmen der Informationstechnologie: **Sprachtechnologie** im engeren Sinne (Spracherkennung, Sprachsynthese, Sprachdialogsysteme) und **Texttechnologie** (Textklassifikation, Textzusammenfassung, Textmining, Informationsextraktion, natürlichsprachliche Suche, Frage-/Antwort-Systeme). Ist dies nicht der Fall, so handelt es sich in der Regel um potenziell **multimodale natürlichsprachliche (Dialog-)Systeme**, die zusätzlich zu gesprochener und geschriebener Sprache auch den Umgang mit Grafik, Mimik und Gestik beherrschen und so eine natürlichere Art der Mensch-Maschine-Interaktion (Kommunikation, Steuerung, Informationsbeschaffung und -präsentation) gewährleisten.

Im folgenden Anwendungskapitel wird — im Wesentlichen aus Platzgründen — der Schwerpunkt auf zentrale Anwendungsgebiete gelegt. Für die Beschreibung anderer Gebiete sei z.B. auf Lobin und Lemnitzer (2004b), Carstensen (2009b) und die im Web verfügbaren Unterkapitel der ersten beiden Auflagen dieses Buchs (<http://www.ifi.uzh.ch/arvo/c1/CLBuch/buch.html>) verwiesen.

5.1 Korrektursysteme

Gerhard Fliedner

Millionen von Textseiten werden heute jeden Tag mit dem Computer erfasst und bearbeitet: Briefe, E-Mails, Aufsätze, Zeitungen, Bücher usw. Dabei schleichen sich Fehler ein. Schon früh entstand die Idee, zur Erkennung und Korrektur dieser Fehler den Computer zu verwenden und Systeme zu entwickeln, die Fehler in Texten automatisch korrigieren oder wenigstens markieren und dem Benutzer Korrekturvorschläge machen.

Nicht nur bei der Bearbeitung von Dokumenten sind solche Systeme hilfreich. Auch bei der Texterfassung mithilfe optischer Zeichenerkennung (*Optical Character Recognition*, OCR) oder bei Abfragen von Datenbanken und Suchmaschinen können automatische Korrekturen die Leistung bestehender Systeme entscheidend steigern. Eine Vorverarbeitungsstufe zur Korrektur von Schreibfehlern (evtl. verbunden mit einer Normalisierung von Abkürzungen und Interpunktions) kann auch praktisch alle Systeme zur Verarbeitung natürlicher Sprache mit Texteingabe (z. B. zur maschinellen Übersetzung) verbessern: Sie ermöglicht die Verarbeitung von Eingaben, für die diese Systeme ohne Korrektur wegen eigentlich unwesentlicher Schreibfehler keine Analyse liefern können.

Erste Systeme zur Korrektur von Eingabefehlern gab es bereits ein gutes Jahrzehnt nach der Konstruktion der ersten Computer (vgl. z. B. Blair 1960; Damerau 1964). Auch wenn damals noch Probleme wie Speicherplatzbedarf und Laufzeit im Mittelpunkt standen, die inzwischen durch leistungsfähigere Hardware gelöst sind, basieren schon diese Systeme auf einigen der auch heute noch verwendeten Grundlagen.

Bei der Korrektur von Fehlern werden zurzeit vier unterschiedliche Arten von Systemen verwendet: Systeme zur Korrektur von „Nichtwörtern“, Systeme zur kontextabhängigen Korrektur, Korrektursysteme für Suchmaschinen und Grammatikkorrektursysteme.

Durch Schreibfehler entstehen meistens „Nichtwörter“ (engl. *non-words*), also Zeichenketten, die in der Sprache nicht vorkommen. Eine **Nichtwort-Korrektur** korrigiert alle Zeichenketten, die nicht als Wörter in einem Systemlexikon gefunden werden können. Nach diesem grundlegenden Prinzip arbeiten die heute z. B. in kommerziellen Textverarbeitungen enthaltenen Rechtschreibkorrekturen.

Manchmal ergeben sich jedoch aus Schreibfehlern keine Nichtwörter, sondern zufällig andere gültige Wörter der Sprache, z. B. *mir der Hand* statt *mit der Hand*. Systeme zur kontextabhängigen Korrektur versuchen, diese Fehler aufzuspüren. Obwohl diese Art von Fehlern bereits in den ersten Arbeiten zur Rechtschreibkorrektur (vgl. Blair 1960) erwähnt wird, ist das Problem aufgrund seiner erheblich größeren Komplexität zunächst nicht weiter verfolgt worden. Die Schwierigkeit ist hier, dass man den Kontext, also die umgebenden Wörter, auswerten muss, um den Fehler überhaupt zu erkennen.

Eine interessante neue Herausforderung ist die Entwicklung von Korrektursystemen, die Benutzereingaben in Suchmaschinen (besonders Internetsuchma-

schinen) korrigieren: Insbesondere hat sich hier der Einsatz eines statischen Systemlexikons als unzureichend erwiesen. Eine wichtige Möglichkeit, dieses und andere spezifische Probleme zu lösen, eröffnet die Auswertung großer Mengen gespeicherter Suchanfragen (*Query Logs*).

Grammatikkorrektursysteme erfassen Grammatikfehler, bei denen die Wörter in der Eingabe zwar wie bei der kontextabhängigen Korrektur orthografisch korrekt sind, die Eingabe aber andere Fehler wie beispielsweise Kongruenz- oder Wortstellungsfehler enthält. Diese Fehler lassen sich erst mit einer syntaktischen Analyse der Eingabe entdecken, z. B. **einen selten Fehler* statt *einen seltenen Fehler*. Grammatikfehler und „Wort-Fehler“ lassen sich nicht in allen Fällen eindeutig voneinander abgrenzen; die zur Korrektur verwendeten Systeme unterscheiden sich allerdings deutlich.

Bisher fehlen noch Systeme, die diese verschiedenen Aufgaben vollständig verbinden. Systeme zur Nichtwort- und Grammatikkorrektur sind inzwischen oft in Anwendungen wie Textverarbeitungen, E-Mail-Programme oder Internet-Browser integriert; es gibt allerdings auch eigenständige Korrekturprogramme. Im Folgenden werden für die verschiedenen Arten von Systemen jeweils der Stand der Technik und die verwendeten Methoden beschrieben.

5.1.1 Korrektur von Nichtwörtern

Systeme zur Korrektur von Nichtwörtern ähneln sich in ihrer Funktion: Sie erlauben es, einen Text – meist interaktiv – nach Wörtern zu durchsuchen, die dem System unbekannt sind. Wird ein solches Wort gefunden, dann macht das System Vorschläge zur Korrektur, von denen der Benutzer den passenden auswählt. Die meisten Systeme bieten darüber hinaus die Möglichkeit, eigene Wörterbücher anzulegen und so schrittweise den Wortschatz des Systems an den eigenen anzupassen. Typische Beispiele hierfür sind das UNIX-Programm `ispell` und die Rechtschreibprüfungen, die inzwischen zum Funktionsumfang praktisch aller Office-, Layout- und E-Mail-Programme sowie von Internet-Browsern gehören (Microsoft Office, Apple iWork, Sun StarOffice, OpenOffice.org, QuarkXPress, Mozilla Thunderbird und FireFox usw.) und jeweils für bis zu 30 Sprachen vorliegen. Die meisten integrierten Rechtschreibprüfungen verfügen über eine Option „Prüfung während der Eingabe“, bei der bereits während des Tipps unbekannte Wörter durch Unterstreichungen markiert werden und so auch ohne einen eigenen Korrekturlauf korrigiert werden können. Korrekturvorschläge können dabei meistens über ein Kontextmenü abgerufen werden.

Zur Erkennung von Wörtern wird ein **Lexikon** verwendet; bei nicht im Lexikon gefundenen Zeichenketten geht das System davon aus, dass es sich um Nichtwörter handelt. Es gibt verschiedene Algorithmen, die das effiziente Suchen von Wörtern in Lexika erlauben. Dazu zählen unter anderem Hashtables, Buchstabenbäume (Tries), balancierte Bäume (alle ausführlich in Knuth 1998) und endliche Automaten (Aho und Corasick 1975). Mit diesen Methoden und auch durch die rasante Entwicklung immer leistungsfähigerer Computerhardware sind die Suche im Lexikon und das Identifizieren von Nichtwörtern heute nicht mehr zeitkritisch.

Die bloße Verwendung von Lexika hat sich jedoch als unzureichend herausgestellt, um natürliche Sprachen voll zu erfassen, in denen Wörter systematisch verändert oder neu gebildet werden. Das Problem tritt besonders deutlich bei Sprachen mit reicher Flexion oder Agglutination auf, in denen von einem Stamm eine Vielzahl – in einigen Sprachen bis zu Tausende – von Formen abgeleitet werden können. Auch durch Komposition und Derivation werden in vielen Sprachen spontan neue Wörter gebildet, die ebenfalls korrekt erkannt werden sollten (im Deutschen Wörter wie *Sicherheitslücke* oder *hobbymäßig*).

Einige Programme (z. B. das UNIX-Tool *ispell*) arbeiten mit einer Liste von möglichen Flexions- und Derivationsaffixen, die bei der Suche im Lexikon zusätzlich berücksichtigt werden. Außerdem erlauben die Programme, dass mehrere Wörter zusammengefügt werden, um so Komposition nachzubilden. Dieser Ansatz lässt sich für Sprachen mit relativ wenigen und überwiegend regelmäßigen Flexionsformen wie Englisch gut anwenden. Je komplexer jedoch die Morphologie der Sprache ist, umso aufwendiger wird es zu verhindern, dass das System auch falsch gebildete Worte und Wortformen akzeptiert (z. B. **die Museums* statt *die Museen*).

Um diese Probleme zu lösen, werden heute meistens **Morphologien** (vgl. Unterkapitel 3.3) verwendet, die (je nach Sprache) Agglutination/Flexion, Derivation und Komposition beschreiben. Ein weit verbreiteter Ansatz ist die zuerst von Koskenniemi beschriebene **Zwei-Ebenen-Morphologie** (Koskenniemi 1983) auf Basis von **Finite-State-Maschinen** (vgl. auch Unterkapitel 2.2). Solche Morphologien können besonders effizient verarbeitet werden, wenn das zugrunde liegende Lexikon ebenfalls als endlicher Automat kodiert wird, da sich dann Morphologie und Lexikon zu einer großen Finite-State-Maschine zusammenkompilieren lassen.

Zwei-Ebenen-Morphologien sind erfolgreich zur Beschreibung von sehr verschiedenen Sprachen verwendet worden, z. B. Arabisch, Deutsch und Finnisch. Die ersten entsprechenden Rechtschreibkorrektursysteme sind seit einigen Jahren auf dem Markt und entwickeln sich zunehmend zum Standard.

Mit den bisher beschriebenen Methoden können Rechtschreibfehler erkannt werden. Um einen Fehler zu korrigieren, benötigt man zusätzliche Verfahren, die zu einem nicht im Lexikon gefundenen Wort die wahrscheinlichste Korrektur im Lexikon finden.

Die älteste, immer noch häufig verwendete Möglichkeit, die wahrscheinlichste Korrektur auszuwählen, ist die **Suche nach dem ähnlichsten String** (*Approximate String Matching*).

Viele Verfahren zur Suche nach dem ähnlichsten String basieren auf einer Funktion, die den Abstand zwischen zwei Strings angibt (**Stringabstandsfunktion**, *String distance function*): Je kleiner der Abstand, umso ähnlicher sind die Strings. Normalerweise wird als Stringabstandsfunktion eine **Minimal Edit Distance** genannte Funktion (Wagner und Fischer 1974; Lowrance und Wagner 1975) verwendet. Die Funktion heißt *Minimal Edit Distance*, da sie die minimale Zahl von Änderungsoperationen angibt, die nötig ist, um ein Wort in ein anderes umzuwandeln (hier: das Nichtwort in ein im Lexikon vorhandenes Wort). Als Änderungsoperationen sind dabei das Auslassen, Einfügen oder Än-

dern eines einzelnen Zeichens, in vielen Formulierungen auch das Vertauschen zweier benachbarter Zeichen (Transposition), definiert. Verschiedene Formulierungen erlauben es, diesen Operationen unterschiedliche Kosten zuzuweisen; in der Praxis werden die Kosten allerdings oft einheitlich auf 1 gesetzt.

Dieser Ansatz basiert auf der schon in Damerau (1964) beschriebenen Beobachtung, dass die meisten Schreibfehler (Damerau geht von 80 % aus) sich auf genau eine solche Änderung zurückführen lassen: Um zum richtigen Wort zu gelangen, müssen also genau alle Möglichkeiten in Betracht gezogen werden, die diese Änderung rückgängig machen. Beispielsweise ist in **Fehler* (statt *Fehler*) das Zeichen *j* fälschlich eingefügt worden. Um den korrekten Lexikoneintrag zu finden, muss es daher entfernt werden. Die Suche nach der wahrscheinlichsten Korrektur über die *Minimal Edit Distance* generalisiert diese Idee auch für Korrekturen, die mehr als eine Änderung erfordern. Teilweise wird die Funktion nach dem Autor einer anderen frühen Arbeit auch als **Levenshtein-Distanz** bezeichnet.

Bei Verwendung von Buchstabentümmlern und Automaten zur Lexikonkodierung gibt es effiziente Verfahren, das Erzeugen von möglichen Korrekturen direkt in die Suche nach einem Wort zu integrieren (vgl. Wagner 1974; Mohri 1996; Oflazer 1996; Schulz und Mihov 2001). Dabei werden systematisch zusätzliche Grafkanten im Baum bzw. Automaten eingefügt, die jeweils genau eine Korrekturoperation beschreiben. Die Kanten sind jeweils mit der Korrekturoperation annotiert, so dass nach einer einzigen Traversierung sowohl das korrekte Wort („normale“ Kanten) als auch die Korrekturen, die nötig sind, um das vorliegende Nichtwort in das korrekte Wort umzuwandeln (Zusatzkanten), bekannt sind.

Eine andere Möglichkeit, den Abstand zwischen zwei Strings zu ermitteln, ist die Verwendung von ***n*-Grammen** von Buchstaben (typischerweise Trigrammen). Als Ähnlichkeitsmaß wird ermittelt, wie viele Buchstabenfolgen der Länge *n* zwei Strings gemeinsam haben, normalisiert über die Stringlänge. So haben beispielsweise *Wort* und *orten* ein Trigramm – *ort* – gemeinsam. Diese Funktion kann für einen schnellen Stringvergleich verwendet werden (Ukkonen 1992). Auch zur Suche in Lexika kann dieses Verfahren verwendet werden: Für die Wörter im Lexikon wird anhand der ermittelten *n*-Gramme ein Index gebildet, der zu jedem *n*-Gramm die Wörter auflistet, die dieses *n*-Gramm enthalten. Um ein Wort nachzuschlagen, wird das gesuchte Wort ebenfalls in *n*-Gramme zerlegt. Dann werden zu allen *n*-Grammen jeweils alle Wörter ermittelt, die dieses *n*-Gramm enthalten. Unter diesen Kandidaten wird dann ermittelt, für welche Wörter die meisten *n*-Gramme mit dem Suchwort übereinstimmen: Dies sind die ähnlichsten Wörter im Lexikon (vgl. Angell et al. 1983, Zobel und Dart 1995).

Eine Korrektursuche mithilfe von Verfahren, die phonetisch ähnlichen Wörtern aufgrund von phonetischer „Normalisierung“ gleiche oder ähnliche Schlüssel zuweisen, haben sich in der Praxis nicht bewährt (vgl. Zobel und Dart 1995).

Neuere Arbeiten versuchen, die Suche nach Korrekturen weiter zu verbessern, indem sie statistische Daten heranziehen. Als informationstheoretische Grundlage dient dabei die Datenübertragung über einen verrauschten Kanal (engl. *Noisy Channel*): Bei jeder Datenübertragung können Fehler auftreten. Man kann aus einem empfangenen (eventuell fehlerhaften) Datum allerdings auf das wahr-

scheinlichste ursprüngliche, korrekte Datum rückschließen, wenn man Informationen darüber hat, welche Störungen der verwendete Übertragungskanal verursacht. Diese Informationen werden zu einem Modell des verrauschten Kanals (*Noisy Channel Model*) zusammengefasst. Bei der Korrektur von Rechtschreibfehlern betrachtet man den gesamten Prozess der Texterfassung als verrauschten Kanal: Man geht davon aus, dass die Person, die den Text verfasst, die korrekten Wörter tippen will, diese Wörter aber auf dem Weg in den Computer verrauschen (u. a. durch falsch angeschlagene Tasten auf der Tastatur).

Damit entspricht das zugrunde liegende Problem, nämlich das wahrscheinlichste intendierte Wort zu einem beobachteten Nichtwort zu finden, dem in Unterkapitel 5.4 beschriebenen Spracherkennungsproblem: Das wahrscheinlichste Wort zu einem gegebenen Nichtwort ist dasjenige, das $P(\text{Wort}|\text{Nichtwort})$ maximiert. Nach der **Bayes-Formel** muss $P(\text{Wort}) \cdot P(\text{Nichtwort}|\text{Wort})$ für das gesuchte Wort maximal sein. Die relative Wahrscheinlichkeit eines Wortes $P(\text{Wort})$ kann mithilfe von Korpora ermittelt werden (Sprachmodell, vgl. Unterkapitel 4.1). Die Wahrscheinlichkeit $P(\text{Nichtwort}|\text{Wort})$, also die Wahrscheinlichkeit, dass statt dem intendierten *Wort* fälschlich *Nichtwort* geschrieben wird, wird anhand eines Modells von Schreibfehlern ermittelt. Dieses Fehlermodell kann grundsätzlich beliebiges Wissen über Schreibfehler enthalten. Je genauer es ist, umso größer ist die Wahrscheinlichkeit, dass sich das intendierte Wort unter den besten Kandidaten befindet.

In Kernigham et al. (1990) und Church und Gale (1991), einer ersten Arbeit in diesem Bereich, wird z. B. das Modell anhand einer von Hand zusammengestellten Sammlung von Fehlern mit den entsprechenden Korrekturen trainiert. Für das Modell wird die Häufigkeit der schon bekannten Fehlerarten Auslassung, Einfügung, Änderung und Transposition, bezogen auf die beteiligten Buchstaben, ermittelt. Mit diesem Modell wurde in einer Evaluation in 87 % der Fälle die richtige Korrektur vorgeschlagen.

In Brill und Moore (2000) wird das Modell dadurch weiter verbessert, dass die Ermittlung von Fehlerarten generalisiert und automatisiert wird, indem statt der genannten vier Grundoperationen beliebige Änderungen betrachtet und jeweils die Kosten für die Änderung ermittelt werden. So können Fehler automatisch genauer klassifiziert werden (z. B. *el* statt *le* am Wortende). Dieses Verfahren erhöht die Wahrscheinlichkeit, dass das richtige Wort als Korrektur vorgeschlagen wird, weiter auf über 98 %.

5.1.2 Kontextabhängige Korrektur

Nicht jeder Eingabefehler führt zu einem Nichtwort. Obwohl es schwierig ist, zuverlässige Zahlen zu ermitteln, deuten Studien darauf hin, dass – zumindest in englischen Texten – möglicherweise bis zu 40 % der Tippfehler gültige Wörter ergeben (vgl. Kukich 1992, S. 413). Peterson (1986) zeigt, dass die Wahrscheinlichkeit, zufällig ein gültiges Lexikonwort zu erzeugen, in Abhängigkeit von der Lexikongröße deutlich steigt. Bei Verwendung eines größeren Lexikons werden insgesamt zwar seltener Wörter zur Korrektur vorgeschlagen. Man erkauft das aber mit einer höheren Wahrscheinlichkeit, Fälle zu übersehen, in denen ein

Tippfehler zufällig ein gültiges Wort ergibt. Durch die Korrektur solcher Fehler ließe sich die Qualität von Rechtschreibkorrektursystemen erheblich verbessern. Bisher fehlen aber noch allgemeine Verfahren, um sie zuverlässig zu erkennen.

Eine Möglichkeit, die bereits in mehreren Grammatikprüfsystemen – z. B. im von IBM entwickelten Critique-System (vgl. Ravin 1993) und bei der Grammatikkorrektur von Microsoft Word – verwendet wird, basiert darauf, ähnlich geschriebene oder ausgesprochene und daher häufig verwechselte Wörter (z. B. engl. *there*, *their* und *they're*) zu **Verwechselungsmengen** (*Confusion Sets*) zusammenzufassen. Sobald in einem Text eines dieser Wörter verwendet wird, werden auf dieses Wort zugeschnittene Heuristiken angewandt, um mögliche Schreibfehler zu entdecken und eine Warnung auszugeben. Dieses Verfahren kann allerdings nur Fehler korrigieren, bei denen genau ein Wort aus einer Verwechselungsmenge anstelle eines anderen Wortes aus dieser Verwechselungsmenge steht, andere Fehler bleiben unbemerkt. Außerdem müssen für jedes Wort passende Heuristiken entwickelt werden.

Ein statistikbasierter Ansatz, der ohne explizite Verwechselungsmengen auskommt und auf *n*-Gramm-Wahrscheinlichkeiten von Wörtern, also der Wahrscheinlichkeit basiert, dass jeweils *n* benachbarte Wörter gemeinsam auftreten, wird in Mays et al. (1991) beschrieben. Solche Wort-*n*-Gramm-Wahrscheinlichkeiten werden durch Analysen sehr großer Korpora ermittelt (vgl. Unterkapitel 4.1) und vor allem in der Spracherkennung häufig verwendet (vgl. Unterkapitel 5.4). Im beschriebenen System werden aus einem gegebenen Satz durch systematisches Verändern von Wörtern durch Anwendung der schon beschriebenen vier Operationen Löschen, Einfügen, Ändern und Transposition von Buchstaben alternative Sätze generiert. Die Wahrscheinlichkeiten der Sätze, also das Produkt der Trigramm-Wahrscheinlichkeiten aller Wörter in diesem Satz, werden ermittelt. Wird dabei für einen vom System erzeugten Satz eine höhere Wahrscheinlichkeit als für den eingegebenen Satz errechnet, so wird er als Korrektur vorgeschlagen. In der Studie erreichen die Autoren für Sätze mit künstlich erzeugten Fehlern Korrekturraten von über 70 %.

In Golding und Schabes (1996) wird ein Verfahren beschrieben, das verschiedene statistische Methoden kombiniert und so bei Tests auf Texten mit künstlich eingestreuten Fehlern Korrekturergebnisse von bis zu 98 % erreicht. Hier müssen allerdings wieder explizit Verwechselungsmengen aufgestellt werden.

Zum Trainieren statistischer Verfahren benötigt man im Allgemeinen sehr große Korpora, die oft schwierig zu beschaffen sind. Aber selbst in sehr großen Korpora finden sich nicht alle möglichen korrekten Wort-*n*-Gramme. Dieses Sparse-Data-Problem lässt sich – u. a. mithilfe von Glättung (Smoothing) – zwar teilweise lösen; es lässt sich jedoch nicht vermeiden, dass korrekte, aber seltene Wortkombinationen als mögliche Fehler markiert werden. Außerdem sinkt die Leistung meistens deutlich, wenn die zu korrigierenden Texte stark von den verwendeten Trainingskorpora abweichen.

5.1.3 Rechtschreibkorrektur für Suchmaschinen

Neben der Rechtschreibkorrektur bei der Texterfassung etabliert sich zunehmend ein weiteres Gebiet als Schwerpunkt von Forschung und Anwendung, nämlich die Korrektur von Suchanfragen an Suchmaschinen. Von besonderem Interesse sind dabei Internetsuchmaschinen wie z. B. Google.

Eine gute Rechtschreibkorrektur für Suchmaschinen ist besonders wichtig, da Benutzern auf Anfragen mit Schreibfehlern entweder gar keine Suchergebnisse angezeigt werden oder nur solche Dokumente, in denen der gleiche Schreibfehler wie in der Anfrage vorliegt. Die Ergebnisliste ist in diesem Fall typischerweise deutlich weniger relevant als diejenige für die korrigierte Suchanfrage. Untersuchungen von gespeicherten Benutzeranfragen (*Query Logs*) zeigen, dass gerade Suchanfragen häufig Schreibfehler enthalten. Cucerzan und Brill (2004) fanden beispielsweise in einer Untersuchung, dass ca. 10 % aller Anfragen an eine Internetsuchmaschine Fehler enthielten.

Eine besondere Herausforderung für die Korrektur von Suchanfragen ist, dass es nicht ausreichend ist, ein festes Systemlexikon als Referenz zu verwenden, wie in Abschnitt 5.1.1 beschrieben. Für große und heterogene Dokumentsammlungen (und natürlich insbesondere für das Internet) ist es nicht möglich, eine ausreichende Abdeckung der verschiedenen Spezialwortschätze zu erreichen. In vielen Gebieten kommen außerdem ständig Neuprägungen dazu. Über manuelle Lexikon-Updates lässt sich dieses Wachstum nicht adäquat erfassen – anders als z. B. bei der Rechtschreibkorrektur in Textverarbeitungen, wo für jeden neuen Text meistens nur wenige unbekannte Wörter ins Benutzerwörterbuch aufgenommen werden müssen. Auch die unüberschaubare Zahl von Eigennamen (Personennamen, geografische Bezeichnungen, Markennamen usw.) in großen Textsammlungen lässt sich praktisch nicht systematisch erfassen.

Der grundsätzliche Ansatz, jedes überhaupt in mindestens einem Dokument vorkommende Wort ins Systemlexikon aufzunehmen und als mögliche Korrektur zu verwenden, ist höchstens für kleine Dokumentsammlungen mit sehr niedrigen Fehlerraten Erfolg versprechend. Für größere Sammlungen oder gar das Internet ist er nicht geeignet, da in den Texten selber zu häufig Schreibfehler vorkommen. Auch die Verwendung einer festen Minimalfrequenz als Kriterium für die Aufnahme ins Systemlexikon ist nicht praktikabel: Häufige Falschschreibungen z. B. populärer Namen kommen durchaus öfter vor als seltene, aber korrekt geschriebene Fachausdrücke.

Um die genannten Probleme zu lösen, werten Korrektursysteme für Suchmaschinen verstärkt *Query Logs* aus. Die zugrunde liegende Annahme ist dabei, dass die korrekte Schreibweise einer Suchanfrage häufiger vorkommt als falsche Schreibvarianten der gleichen Anfrage.

Der Ansatz von Cucerzan und Brill (2004) ist es, zu einer gegebenen, möglicherweise falsch geschriebenen Suchanfrage systematisch häufigere, ähnliche Anfragen in den *Query Logs* zu suchen. Ähnlichkeit ist dabei über eine Stringabstandsfunction (vgl. Abschnitt 5.1.1) definiert. Um auch Anfragen mit mehreren Fehlern korrigieren zu können, kann dieser Prozess mehrfach wiederholt werden. Dabei ist es auch erlaubt, eine Anfrage mit mehreren Wörtern aus mehreren an-

deren beobachteten Anfragen zusammenzusetzen, um das Sparse-Data-Problem zu umgehen.

Li et al. (2006) verwenden zur Fehlererkennung und -korrektur die sogenannte Verwechslungswahrscheinlichkeit (engl. *confusion probability*): Wenn zwei ähnliche Wörter häufig in der gleichen Umgebung, also zusammen mit den gleichen anderen Wörtern, vorkommen, ist das häufigere mit hoher Wahrscheinlichkeit die korrekte Variante. So lassen sich aus *Query Logs* wahrscheinliche Korrekturen für falsch geschriebene Wörter auch in Fällen lernen, in denen die oben beschriebenen Korrekturmethoden einen anderen Kandidaten bevorzugen. Die Autoren beschreiben beispielsweise, dass die Auswertung der *Query Logs* zeigt, dass das falsch geschriebene Wort *adventura* in ähnlichen Kontexten verwendet wurde wie *aventura*. Damit ist *aventura* eine wahrscheinlichere Korrektur als das insgesamt deutlich häufigere und – gemessen am Stringabstand – genauso ähnliche Wort *adventure*. Die Autoren verwenden dieses und eine Reihe weiterer Merkmale (u. a. auch den Stringabstand), um verschiedene statistische Modelle zur Fehlerkorrektur zu trainieren. In einer Evaluation zeigen diese Modelle eine deutliche Verbesserung gegenüber der Baseline (ähnlich dem oben beschriebenen Ansatz von Brill und Moore 2000).

Ein Nachteil dieses Ansatzes ist, dass für Wörter, die in *Query Logs* so selten vorkommen, dass sich keine brauchbaren Verteilungswahrscheinlichkeiten ermitteln lassen, keine besseren Korrekturvorschläge als bei Verwendung des reinen Stringabstandsverfahrens gemacht werden können. In einer Folgearbeit (Chen et al. 2007) ergänzen die Autoren das Modell durch Ergebnisse von Internetsuchen nach den jeweiligen Kandidaten. Das basiert auf der Beobachtung, dass Internetseiten, in denen eine falsch geschriebene Variante eines Wortes vorkommt, häufig auf Seiten verlinken, die die korrekt geschriebene Variante enthalten. Über diese weitere Datenquelle lassen sich auch für in *Query Logs* selten vorkommende Wörter deutlich bessere Korrekturvorschläge machen.

5.1.4 Grammatikkorrektur

Neben den beschriebenen Fällen kommen auch solche vor, in denen die Erkennung und Korrektur eines Fehlers nicht nur einen lokalen Kontext, sondern auch die Analyse eines ganzen Satzes und eventuell sogar des gesamten Textes voraussetzen. Typische Fehler dieser Art sind Grammatikfehler wie z. B. Kongruenzfehler (Subjekt und Prädikat, Adjektiv und Substantiv oder Antezedens und Pronomen stimmen nicht überein), fehlende Wörter (insbesondere Artikel und Präpositionen), falsche Argumente bei Verben (z. B. im Deutschen falscher Kasus oder Infinitivkonstruktion statt Dass-Satz).

Grammatikkorrektursysteme finden und korrigieren zumindest einige dieser Fehler. Ein Problem dabei ist, dass die syntaktische Analysekomponente von Systemen zur Verarbeitung natürlicher Sprache normalerweise Sätze vollständig zurückweist, die Fehler enthalten. Da der Benutzer aber genauere Informationen zum Fehler benötigt, muss die Analyse so verändert werden, dass sie bei nicht-grammatischen Eingaben sinnvolle Vorschläge zur Verbesserung machen kann. Dazu werden im Wesentlichen zwei Verfahren verwendet: ***Constraint Relaxation*** und ***Fehlerantizipation*** (*Error Anticipation*).

Bei der *Constraint Relaxation* werden bestimmte *Constraints* (Grammatikalitätsbedingungen, z. B. Kongruenz) innerhalb der Grammatik als nicht absolut feststehend betrachtet: Wenn für eine Eingabe keine Analyse gefunden werden kann, in der alle Constraints erfüllt sind, werden diese systematisch gelockert (relaxiert). Sobald eine Analyse mit einem relaxierten Constraint gelingt, ist bekannt, welcher Constraint genau verletzt wurde, und damit auch, welcher Grammatikfehler in der Eingabe vorliegt.

Bei der Fehlerantizipation geht man davon aus, dass viele Fehler Mustern folgen und deswegen mithilfe von Fehlermustern durch **Musterabgleich** (*Pattern Matching*) gefunden werden können. Ein Fehler wird hier also nur gefunden, wenn er einem vorher definierten Fehlermuster entspricht.

In der Praxis werden diese beiden Verfahren oft kombiniert, insbesondere weil eine volle syntaktische Analyse mit *Constraint Relaxation* im Allgemeinen zwar aufgrund umfangreicherer Informationen genauere Ergebnisse liefert, dafür aber mehr Zeit in Anspruch nimmt als ein einfacher Musterabgleich.

Ein System, das beide Verfahren vereinigt, ist das bei IBM entwickelte *Critique-System* (Jensen et al. 1993, Kapitel 3–7). Dieses System verwendet im Wesentlichen *Constraint Relaxation* zur Erkennung und Korrektur von Grammatikfehlern und Fehlerantizipation zur Erkennung von Stilfehlern (z. B. überlange Sätze). Ähnliche Systeme werden heute zusammen mit kommerziellen Textverarbeitungen (z. B. Microsoft Word) verkauft.

Auch in der computerlinguistischen Forschung sind verschiedene solcher Systeme beschrieben worden, z. B. die Grammatikprüfung für das Deutsche im Multilint-System (Schmidt-Wigger 1998), Scarrie für skandinavische Sprachen (Povlsen et al. 1999) und GramCheck (Ramírez Bustamente und Sánchez León 1996) für Spanisch.

Der Hauptkritikpunkt an Grammatikkorrektursystemen ist die zurzeit für den alltäglichen Gebrauch oft noch zu niedrige **Präzision**. Präzision ist ein ursprünglich im Bereich Information Retrieval geprägter Begriff und bezeichnet hier das Verhältnis der Zahl der vom System gefundenen tatsächlichen Grammatikfehler zur Zahl der insgesamt gemeldeten Fehler. Heutige Systeme erreichen meistens eine Gesamtpräzision von maximal 50 %, d. h. pro tatsächlich gefundenem Fehler wird im Schnitt mindestens eine falsche Warnung ausgegeben, was von vielen Anwendern als zu lästig empfunden wird.

Die **Vollständigkeit** (also das Verhältnis von vom System gefundenen zu den tatsächlich vorhandenen Fehlern, engl. *Recall*) ist ebenfalls noch nicht ausreichend, um Computer zur Endkorrektur von Texten einzusetzen, d. h. es werden noch zu viele Fehler übersehen – zum Teil, weil eine bestimmte Fehlerklasse im System gar nicht vorgesehen ist.

5.1.5 Perspektiven

Systeme zur Rechtschreibkorrektur sind eine der ältesten Anwendungen der Computerlinguistik. Durch die lange Forschung und die Möglichkeit, auch mit Lösungen für Teilprobleme (z. B. Nichtwort-Korrektur) hilfreiche Werkzeuge zu entwickeln, handelt es sich um eines der Gebiete der Computerlinguistik mit den

meisten alltagstauglichen Systemen. Diese Systeme sind weit verbreitet und auch kommerziell erfolgreich. Doch auch auf diesem Gebiet sind noch lange nicht alle Probleme gelöst. Aktuelle Forschungen versprechen eine stetige Verbesserung der Systeme in verschiedenen Bereichen.

Die vorgeschlagenen Korrekturen könnten durch die stärkere Einbeziehung von Statistiken über häufige Tippfehler und Verfahren wie z. B. die Berücksichtigung typischer Fehlerquellen (z. B. auf der Tastatur nahe beieinander liegende Tasten, die oft vertauscht oder zusammen angeschlagen werden) noch verbessert werden. Aktuelle Arbeiten in diesem Bereich zeigen viel versprechende Ergebnisse (vgl. Abschnitt 5.1.1).

Statistische Verfahren, insbesondere für kontextabhängige Korrekturen, können wahrscheinlich ebenfalls noch weiter verfeinert und für unterschiedliche Textsorten verallgemeinert werden. Allerdings scheint sich hier abzuzeichnen, dass man bei der Anwendung von korpustrainierten statistischen Verfahren auf allgemeine Texte an bestimmte Obergrenzen für richtige Korrekturen stößt (vermutlich im Bereich zwischen 70 und 90 %).

Der Bereich der Grammatikkorrektur ist der komplexeste und sollte daher am meisten Raum für Verbesserungen bieten. Die Verwendung effizienter Parser und die ständige Weiterentwicklung der Hardware erlauben den Einsatz immer mächtigerer Grammatiken ohne unzumutbare Wartezeiten. Zunehmende Erfahrung mit der systematischen Entwicklung großer, aber dennoch wartbarer Grammatiken wird Schritt für Schritt auch die Entwicklung von Grammatikprüfsystemen mit besserer Abdeckung erlauben. Auch die Verwendung von Hybridsystemen, die z. B. manuell geschriebene Grammatiken mit statistischen Komponenten verbinden, könnte zum einen die Entwicklungszeit der Systeme verkürzen, zum anderen zu einer größeren Flexibilität führen.

Semantische Verfahren könnten zumindest in Teilbereichen der Grammatikprüfung eingesetzt werden. Mit ihrer Hilfe könnten auch Fehler entdeckt werden, die keine syntaktischen Regeln im engeren Sinne verletzen, z. B. *#einen Reset auslösen* statt *auslösen*. Damit könnten die meist noch getrennten Bereiche Grammatik- und kontextabhängige Korrektur enger zusammenrücken.

Heutige Rechtschreib- und Grammatikprüfungen werden von den Benutzern zunehmend akzeptiert und als wirkliche Hilfe empfunden: Mit diesen Systemen kann man einen Text interaktiv schneller und mit ähnlich guten oder sogar besseren Ergebnissen überprüfen und korrigieren als rein „manuell“. Auch die Möglichkeit, Tippfehler in Anfragen an Suchmaschinen mit einem Klick zu korrigieren, wissen viele Benutzer zu schätzen.

Bis der Computer Texte selbstständig korrigiert, werden, je nach Art der Texte, wohl noch Jahrzehnte vergehen. Aber schon während dieser Zeit werden Korrektursysteme Autoren und Korrektoren immer mehr Unterstützung bei ihren Aufgaben bieten.

5.1.6 Literaturhinweise

Die meisten Verfahren zur Rechtschreibkorrektur sind aufgrund der langen Beschäftigung mit dem Gebiet gut etabliert. Neuere Entwicklungen sind die stärke-

re Verbreitung von morphologiebasierten Verfahren und die Forschung im Gebiet der kontextabhängigen und der Grammatik-Korrektur sowie der Korrektur von Anfragen an Suchmaschinen.

Ein guter, in weiten Teilen immer noch aktueller Überblick über das Gebiet findet sich in Kukich (1992). Hier wird eine große Zahl verschiedener Verfahren und Systeme kurz vorgestellt; eine umfangreiche Literaturliste begleitet die Arbeit.

Das Problem der Rechtschreibkorrektur wird in dem Einführungsbuch Jurafsky und Martin (2009) an mehreren Stellen aufgegriffen. Verschiedene Verfahren zur Korrektur (*Minimal Edit Distance*, statistische Korrektur) werden anschaulich beschrieben.

Peterson (1980) beschreibt die Grundlagen eines Systems zur Korrektur von Wörtern, die nicht im Systemlexikon gefunden werden können. Die Beschreibung enthält unter anderem Verfahren zur Suche von Wörtern in einem Lexikon, zur Korrektur nicht im Lexikon gefundener Wörter und zur Analyse von Flexionsendungen. Die beschriebenen Techniken werden in aktuellen Rechtschreibkorrektursystemen eingesetzt.

In Mitton (1996) wird ebenfalls ein Korrektursystem entwickelt. Die Arbeit legt einen besonderen Schwerpunkt auf Fehleranalyse. Mittons System bietet nicht nur eine Nichtwortkorrektur, sondern stellt auch eine kontextabhängige Wortkorrektur zur Verfügung.

In Norvig (2007) wird ein einfaches Korrekturprogramm entwickelt (mit Quelltext in *Python*), das exemplarisch die Kombination von *Minimal Edit Distance* mit einem aus Korpora gelernten Sprach- und Fehlermodell zeigt. Das Dokument enthält außerdem Verweise auf andere Beispielimplementationen.

Verfahren zur effizienten Suche gehören zu den Grundlagen der Informatik. Viele davon lassen sich für die Suche nach Wörtern in einem Lexikon verwenden. Knuth (1998) bietet eine umfassende und lesbare Grundlage dieses Gebiets, die auch theoretische Ergebnisse enthält. Ein ausführlicher Überblick über verschiedene Ansätze und Verfahren zur Suche nach dem ähnlichsten String findet sich in Navarro (2001).

Wie oben beschrieben, werden Morphologien zunehmend als Grundlage für die Rechtschreibkorrektur eingesetzt. Hier sei auf die Literaturhinweise des Morphologieunterkapitels (3.3) verwiesen.

Als Klassiker der Grammatikkorrektursysteme gilt das Critique-System (Jensen et al. 1993, Kapitel 3–7). Vieles in der Beschreibung dieses speziellen Systems lässt sich auf Grammatikkorrektursysteme allgemein übertragen.

Ein detaillierter Überblick über Grundlagen der Grammatikkorrektur und eine umfangreiche Liste von in der Literatur beschriebenen Systemen für verschiedene Sprachen findet sich in Fliedner (2001).

5.2 Computergestützte Lexikographie und Terminologie

Ulrich Heid

5.2.1 Lexikographie und Terminologie

Der Terminus Lexikographie bezeichnet im engeren Sinne die Praxis der Erstellung von Wörterbüchern, in einem weiteren Sinne jede wissenschaftliche Beschäftigung mit Wörterbüchern und mit den Prozessen ihrer Entstehung, also auch Wörterbuchanalyse und Wörterbuchkritik, Theorie und Geschichte der Lexikographie oder Forschungen zur Wörterbuchbenutzung und zu Arbeitsmethoden bei der Erstellung von Wörterbüchern. Diese wissenschaftliche Beschäftigung mit Wörterbüchern nennt man auch Metalexikographie (vgl. den Überblick in Hausmann 1985).

Ergebnis und Produkt der Wörterbucherstellung sind einsprachige, zweisprachige oder mehrsprachige Wörterbücher, gedruckte, interaktiv vom Menschen am Rechner benutzbare Wörterbücher und Wörterbücher sprachverarbeitender Systeme (vgl. Unterkapitel 4.4). Wörterbücher können unterschiedlich strukturiert sein, je nach Zugriffsbedarf: z. B. alphabetisch, nach semantischer Ähnlichkeit (vgl. z. B. WordNet etc.) oder nach anderen linguistischen Eigenschaften des Wortschatzes. Wörterbücher, die von NLP-Systemen benutzt werden können, gehören zu den Sprachressourcen, ebenso wie Grammatiken, Textkorpora oder Speech-Ressourcen.

Die Lexikographie befasst sich primär mit dem Wortschatz der Gemeinsprache. Die Beschreibung des Fachwortschatzes einzelner Wissensbereiche ist die Aufgabe der *Terminologie*. Sie beschreibt und systematisiert Fachtermini („Terme“), traditionellerweise durch eine konzeptuelle Beschreibung (z. B. als Domänen-Ontologie) und durch die linguistische Beschreibung der Fachwörter und -wendungen (Mehrwortausdrücke), die die Konzepte einer Domäne und ihre Interaktion benennen. Die Erfassung von Fachwortschatz in Terminologiedatenbanken (oft abgekürzt: Termbanken) oder gedruckten oder interaktiven Fachwörterbüchern wird oft *Terminographie* genannt.

Die Grenzen zwischen Fachwortschatz und Gemeinwortschatz sind fließend: z. B. hat das Wort *Kind* im Fachwortschatz der Kindergeldgesetzgebung selbstverständlich Termstatus. Wegen der unscharfen Grenzen zwischen Gemeinsprache und Fachsprache, zwischen Lexikographie und Terminographie, lassen sich viele computationelle Fragestellungen relativ parallel in beiden Bereichen behandeln. Für die verschiedenen Forschungsanliegen, Methoden und Werkzeuge, bei denen computerlinguistische Arbeit an lexikalischen Daten eine Rolle spielt, wurde das Begriffspaar *Computational Lexicography* vs. *Computational Terminology* geprägt.

5.2.2 Die Teilbereiche im Überblick

Computational Lexicography und Computational Terminology zielen auf die Erarbeitung, Verwaltung und Nutzung lexikalischen Wissen. Dazu gehören einzelne Wörter, aber auch Morpheme (z. B. Wortbildungssuffixe, wie im Deutschen *-bar*, *-ung*, *an-*, *über-* usw.) und Wortverbindungen, z. B. (ggf. idiomatische) Mehrwortausdrücke (z. B. *ab und zu*, *zur Rede stellen* usw.). Solche lexikalischen Objekte werden linguistisch beschrieben, und zwar auf allen Ebenen, z. B. phonetisch, morphologisch, syntaktisch, semantisch, pragmatisch.

Lexikalische Ressourcen dienen einerseits für NLP-Systeme: kein sprachverarbeitendes System kommt ohne lexikalisches Wissen aus. Je nach Zielanwendung, zu Grunde liegender linguistischer Theorie und Modellierungsansatz ergibt sich ein je unterschiedlicher Bedarf an lexikalischem Wissen. Ebenso brauchen unterschiedliche Benutzergruppen von interaktiven elektronischen Wörterbüchern (z. B. Lernende, Übersetzer, Gegenstandsbereichsexperten) unterschiedliches lexikalisches Wissen. In beiden Fällen betreffen die Unterschiede sowohl die Auswahl aus den verfügbaren Informationstypen, (oder, in der Sprechweise der Lexikographie: den verfügbaren Angabetypen), als auch die Präsentation, d.h. Ausführlichkeit, Anordnung und GUI-Gestaltung. Um die Bedürfnisse verschiedener NLP-Anwendungen durch eine gemeinsame Ressource abzudecken, wird daran gearbeitet, umfangreiche anwendungsneutrale, d.h. multifunktionale Ressourcen zu erstellen, aus denen jeweils bedarfsspezifische Teilwörterbücher abgeleitet werden können.

Computationelle Verfahren in Lexikographie und Terminologie betreffen alle Aspekte der Beschaffung, Verwaltung, Repräsentation und Nutzung lexikalischen Wissens:

- Akquisition von lexikalischem Wissen:
 - aus Textkorpora,
 - aus bestehenden Wörterbüchern;
- Verwaltung und Repräsentation von lexikalischem Wissen
 - innerhalb von NLP-Ressourcen und NLP-Systemen,
 - für den Austausch zwischen Entwicklern und/oder Anwendern (resource sharing),
 - für interaktiv benutzbare elektronische Wörterbücher;
- Nutzung von lexikalischem Wissen
 - in NLP-Systemen,
 - bei der interaktiven Verwendung.

In allen genannten Bereichen werden computerlinguistische Verfahren eingesetzt. Die einzelnen Bereiche werden im Folgenden genauer dargestellt.

5.2.3 Akquisition von lexikalischem Wissen

Lexikalische Akquisition (*lexical acquisition*) dient der Beschaffung von Daten, auf denen die lexikographische Beschreibungsarbeit aufsetzen kann. Wörterbuchverlage verwenden als Quellen Textmaterial, Belegkarteien (z.T. manuell erstellt), elektronische Versionen früherer Ausgaben eines Wörterbuchs, elektronische Datensammlungen mit Material aus verschiedenen bereits publizierten Wörterbüchern und die Intuition der Redakteure. Für den interaktiven Aufbau von Wörterbüchern für die Sprachverarbeitung werden in ähnlicher Weise Korpora, bestehende Wörterbücher und Intuitionen über die sprachlichen Phänomene benutzt.

Da die interaktiven Verfahren teuer, langsam und oft auch für Inkonsistenzen anfällig sind, weicht die Sprachverarbeitung zunehmend auf semi-automatische oder vollautomatische Akquisition aus Korpora aus. Viele automatische Verfahren beruhen auf Techniken des maschinellen Lernens (vgl. Mitchell 1997). Der Bereich der lexikalischen Akquisition ist somit am Schnittpunkt von Computational Lexicography/Terminology und Korpuslinguistik.

Akquisition lexikalischen Wissens aus Textkorpora

Die interaktive Suche nach lexikographisch relevanten Wortkontexten setzt auf *Konkordanzen* auf, die einen Keyword-in-Context-Output liefern (daher die Kurzform: KWIC): ein Wort oder eine Wortsequenz wird mit linkem und rechtem Kontext angegeben. Der Lexikograph kann die Belege sortieren oder nach weiteren Suchkriterien filtern, um sich daraus relevante Daten abzuleiten.

Automatische Verfahren zur lexikalischen Datenextraktion (oft in der Literatur: *lexical acquisition*, trotz der Namensgleichheit mit dem Wortschatzlernen bei Kleinkindern) decken meist zweierlei Aufgaben ab: die Suche nach Kandidatenwörtern oder -wendungen und deren linguistische Klassifikation. Dabei können die Klassifikationskriterien vorgegeben sein, oder im Rahmen allgemeiner Constraints vom System z. B. durch Clustering ermittelt werden. Verfahren zur Extraktion lexikalischer Daten sind zum Teil symbolisch, zum Teil rein statistisch, meist kombiniert. Immer werden im Kontext beobachtete Phänomene benutzt, um eine abstraktere Klassifikation zu erreichen.

Beispiele für Bereiche, in denen sehr viel Forschungsarbeit in lexikalische Datenextraktion investiert worden ist, sind die Subkategorisierung und der Aufbau syntaktischer Wörterbücher (vgl. Schulte im Walde 2009), Mehrwortausdrücke und ihre Klassifizierung, sowie die Identifikation von Synonymierelationen oder taxonomischen Relationen (Ober-/Unterbegriff, etc.) und der Aufbau von Ontologien und Thesauri. Aus syntaktischen Klassen und lexikalisch-semantischen Relationen lassen sich auch Anhaltspunkte für semantische Klassifizierungen des Wortschatzes gewinnen. Im Zusammenhang mit statistischer Maschineller Übersetzung, Wortalignment und Terminographie wird an der Extraktion von Äquivalentkandidaten aus Parallelkorpora und aus *comparable corpora* (d.h. muttersprachlichen Texten verschiedener Sprachen, zum selben Thema) gearbeitet.

Für alle Verfahren, die auf linguistischem Wissen aufsetzen, gilt natürlich, dass detailliertere und abstraktere Annotationen im Korpus spezifischere Suchverfahren und eine höhere Precision des Ergebnisses erlauben. Umgekehrt zielen Arbeiten zur Nutzung von maschinellem Lernen darauf, auch aus weniger detailliert vorverarbeiteten Korpora präzise Ergebnisse zu extrahieren, ohne den Recall zu beeinträchtigen.

Man will Beispielsätze in Korpora identifizieren, aus denen Wissen über Wörter abgeleitet werden kann. Die Sätze sollen klassifiziert und zusammen mit dem Phänomen und dem Wort, das sie illustrieren, verfügbar gemacht werden. Für Phänomene, die einzelne Wörter betreffen, kommt man mit Korpora von ca. 60 Millionen Wortformen aus (die Produktion einer Tageszeitung in 3 bis 4 Jahren). Für die Beschreibung von Wortgruppen ist mindestens drei- bis viermal so viel Material nötig. Das *British National Corpus*, BNC, das von Wörterbuchverlagen mitentwickelt wurde, umfasst rund 100 Millionen Wortformen.

Akquisition lexikalischen Wissens aus bestehenden Wörterbüchern

Die ersten als „Computational Lexicography“ bezeichneten Arbeiten, Anfang der 1980er Jahre, zielten auf die Nutzbarmachung der Inhalte gedruckter, z. B. als Satzdaten oder Textfiles vorliegender Wörterbücher für die Sprachverarbeitung.

Extrahiert wurden und werden bis heute neben morphologischen und syntaktischen Angaben vor allem Definitionen, als Grundlage für den Aufbau von Begriffshierarchien und Thesauri: Aus den Textmustern der Definitionen (Regel: genus proximum + differentia specifica, d.h. Oberbegriff + typische Merkmale) hat man Oberbegriffe extrahiert und daraus Begriffshierarchien aufgebaut. Dazu wurden die Definitionstexte zum Teil syntaktisch analysiert, zum Teil wurden auch hier musterbasierte Suchverfahren verwendet (Überblick und Beispiele in Boguraev und Briscoe 1989).

Bei der Nutzung von Angaben aus bestehenden Wörterbüchern für die Erstellung neuer Wörterbuchdaten ist folgendes zu leisten:

- Analyse des Quellwörterbuchs: Identifikation der verfügbaren Informati-onstypen (d.h. der Angabetypen), der Textstruktur der Wörterbuchartikel und der verwendeten Präsentationsmittel (z. B. Druck, Auszeichnung, Symbole).
- Definition einer Abbildung von den Beschreibungen des Quellwörterbuchs auf die für das Zielwörterbuch vorgesehenen Klassifizierungen.
- Extraktion der Angaben aus dem Quellwörterbuch, Reformierung und Einbindung ins Zielwörterbuch.

Neuere Wörterbücher liegen vielfach als XML-Daten vor; der technische Aufwand der Extraktion wird durch den Einsatz von XML-basierten Standardwerkzeugen, z. B. XML-Parsern, XML-Editoren, Stylesheets, geringer als früher; der lexikographische Analyseaufwand, d.h. der Aufwand für die ersten beiden oben

genannten Schritte, bleibt natürlich weitgehend derselbe. Oft ist es aber ausreichend, nur einzelne Angabetypen zu benutzen, und es ist keine vollständige Analyse aller Teile des Wörterbuchs nötig.

Neben elektronisch vorliegenden Versionen von gedruckten Wörterbüchern sind, insbesondere für die Philologie, auch Verfahren der Retrodigitalisierung gedruckter Wörterbücher relevant: Ziel ist es, computergestützt Wörterbücher, die nur auf Papier vorliegen, in ein maschinell verarbeitbares (z. B. XML-basiertes) Format zu bringen. Dadurch können auch solche Wörterbücher als Wissensquellen für interaktive Abfrage oder für automatische Verfahren verwendet werden.

Die Umsetzung vorhandener gedruckter Wörterbücher in eine interaktiv (auf CD-ROM oder über das Internet) abfragbare Form, bzw. die Erstellung von Wörterbüchern mit Blick auf beide Publikationsformen (gedruckt und elektronisch), ist seit Beginn des neuen Jahrtausends von sehr vielen Verlagen prioritär vorangetrieben worden. Je nach Quelldaten werden dabei Einträge des gedruckten Wörterbuchs am Bildschirm angezeigt oder nach Angaben oder als Volltext durchsuchbar gemacht. Solche interaktiven Wörterbücher bieten effizientere Suchmöglichkeiten als die gedruckte Version.

Termextraktion

Ziel der Termextraktion ist es, den Fachwortschatz eines Gegenstandsbereichs aus Textmaterial zu extrahieren. Diese Aufgabe wird dadurch erschwert, dass (i) Gegenstandsbereiche oft nicht genau abgegrenzt sind, (ii) viele Texte Themen zu mehr als einem Gegenstandsbereich beschreiben, also auch Fachwortschatz zu verschiedenen Bereichen enthalten, und (iii) die Definition dessen, was ein Fachterminus ist, von Anwendern und Anwendung Zielen abhängt. Trotzdem ist es für Übersetzungsanwendungen sinnvoll und nötig, Fachtermini aus Fachtexten zu extrahieren. Viele Fachtermini sind Mehrwortausdrücke.

Für die Termextraktion können im Prinzip dieselben Verfahren verwendet werden, wie für die Suche nach Mehrwortausdrücken. Daneben wurden verschiedene spezifische statistische Verfahren vorgeschlagen, die z. B. auf der Suche nach Häufungen von Termkandidaten an bestimmten Textstellen beruhen, oder auf der relativen Häufigkeit von Termkandidaten im Fachtext, verglichen mit der relativen Häufigkeit dieser Termkandidaten in gemeinsprachlichen Texten (die Annahme ist, dass Terme in Fachtexten häufig, in nicht-fachlichen oder fachlich ausgewogenen Texten aber unauffällig sind).

Aufsetzend auf den Verfahren zur Extraktion von Termkandidaten wird auch nach den sprachlichen Ausdrucksmöglichkeiten für Relationen zwischen Konzepten gesucht. Die Terminologielehre geht davon aus, dass Fachwörter sprachliche Ausdrücke für Konzepte sind. Wie die lexikalisch-semantischen Wortnetze (vgl. Unterkapitel 4.3) sind Fachterminologien oft als Konzepthierarchien organisiert. Dabei finden dieselben Relationen Verwendung wie z. B. in WordNet; zusätzlich mitunter noch weitere, die für den Gegenstandsbereich spezifisch sind. Viele solche Relationen werden durch relativ wenige und spezifische sprachliche Mittel ausgedrückt (z. B. Teil-von-Relationen, kausale Relationen).

Ziel von Extraktionsverfahren (musterbasiert, oder anhand von Prädikat-Argument-Strukturen) ist es, einerseits die sprachlichen Mittel zu identifizieren, mit denen die Relationen ausgedrückt werden, andererseits Paare von durch eine bestimmte Relation verbundenen Termen zu finden und für den Aufbau der elektronischen Terminologiesammlungen bereitzustellen. Einen Überblick über Termextraktionsysteme geben z. B. Cabré Castellví et al. (2001).

Die Hauptabnehmer von Terminologie sind Übersetzer, technische Autoren und Anwender von Information Retrieval und Informationsextraktion. Für diese Bereiche wird Multilingualität immer wichtiger, sodass das Interesse an mehrsprachigen Terminologien besonders groß ist. Außerdem wird angenommen, dass für einen relativ hohen Anteil des Fachwortschatzes eine einfache Äquivalenz (ohne allzu komplizierte Kontextbedingungen) gilt.

Anhand von parallelen Korpora (deren Texte Übersetzungen voneinander sind) wird zunächst die Zuordnung von äquivalenten Sätzen (Satzalignment) vorgenommen, dann versucht, innerhalb der Sätze Wortäquivalenzen oder Äquivalenzen auf der Ebene von Wortgruppen zu finden (Wortalignment, word alignment). Verfahren hierzu beruhen auf statistischen Modellen (vgl. die Diskussion der beispielbasierten Übersetzung und der statistischen Übersetzung in Unterkapitel 5.7). Die Suche nach Übersetzungäquivalenten kann mit der Identifikation von Termkandidaten kombiniert werden: z. B. zuerst Termkandidaten-identifikation im quellsprachlichen und im Zielsprachlichen Text, dann Markierung der Termkandidaten in beiden Texten und Nutzung von word alignment zur Feststellung der Term-Äquivalenzen.

5.2.4 Verwaltung und Repräsentation lexikalischen Wissens

Wie oben angedeutet (vgl. Unterkapitel 4.4), benötigen unterschiedliche NLP-Anwendungen bzw. unterschiedliche Benutzergruppen verschiedene Arten, Mengen und Formen von lexikalischem Wissen. Z. B. wird die morphosyntaktische Beschreibung für die meisten POS-Tagger weniger feinkörnig sein, als für einen Parser. Aus der Sicht der Verwaltung und Repräsentation lexikalischen Wissens stellt sich also das Problem, welche Angaben im Wörterbuch repräsentiert werden sollen, in welcher Granularität, und nach welchen linguistischen Ansätzen oder Theorien.

Daneben soll es möglich sein, Daten aus verschiedenen Wörterbüchern zu kombinieren, lexikalische Daten für verschiedene Anwendungen zu nutzen, oder sie zwischen verschiedenen Institutionen auszutauschen.

Diese Fragestellungen machen eine von vielen Anwendern anerkannte, idealerweise automatisch transformierbare Repräsentation der lexikalischen Daten nötig, ein Format für den Datenaustausch. Dabei geht es um die „Syntax“ der Repräsentation (das eigentliche Datenformat) und um die „Semantik“ der Datenkategorien (d.h. die Interpretation von Beschreibungsmitteln: z. B. was bedeutet „direktes Objekt“?). Für beides wurden Normungsvorschläge gemacht: für die inhaltliche Definition wurde ein Inventar gemeinsamer Datenkategorien vorgeschlagen, das zentral verwaltet und für Forscher und Entwickler weltweit zugänglich gemacht werden soll. (das ISO Data Category Repository, vgl. ISO

12 620). Daneben wurden Formate für die interne Repräsentation lexikalischer Daten in XML-basierten Datensammlungen entwickelt, z. B. das *Lexical Markup Framework*, LMF (Francopoulo et al. 2009).

Aus inhaltlicher Sicht ist festzuhalten, dass alle Arten von elektronischen Wörterbüchern erheblich stärker netzartige Wissenstrukturen enthalten (sollten) als gedruckte Wörterbücher. Der Wortschatz besteht aus lexikalischen Objekten (Wörtern, Morphemen, Mehrwortausdrücken), die vielerlei Relationen untereinander aufweisen: lexikalisch-semantische, wie sie in WordNet erfasst sind (z. B. Synonymie, Hyponomie, Hyperonymie usw.), aber auch morphologische Wortbildungsbeziehungen, oder Relationen, die sich aus der Zugehörigkeit zu bestimmten (syntaktischen oder semantischen) Klassen ergeben. In der Terminologie spielt die Zugehörigkeit zu einem Gegenstandsbereich eine wichtige Rolle. Repräsentationsformate müssen also mehr leisten, als nur lexikalische Listen abzubilden: sie sollen die Navigation in den lexikalischen Netzen, aber auch den Zugriff auf Daten mit beliebigen, ggf. unterspezifizierten Suchanfragen unterstützen.

Repräsentation lexikalischen Wissens für NLP-Systeme

Die Repräsentation, d.h. der Formalismus, die Notation lexikalischen Wissens und die darauf aufsetzenden Verfahren für die Wartung und Nutzung der Daten, ist immer von der oder den in Aussicht genommenen Anwendungen abhängig. Unifikationsgrammatiken beispielsweise benötigen Lexikoneinträge auf der Basis von Attribut-Wert-Paaren.

Die Ablage von lexikalischen Daten in multifunktionalen Wörterbüchern, d.h. für verschiedene Anwendungen, erfolgt oft in Datenbanken oder in XML-Dateien (oder XML-Datenbanken). Daneben werden auch Typsysteme von getypten Merkmalslogiken oder spezifische dedizierte Formalismen benutzt. Außerdem wurden verschiedene spezifische Repräsentationsformalismen erprobt, die von graph-basierten Formaten mit String-wertigen Attribut-Werten (z. B. Lexical Systems, vgl. (Polguère 2009), Lexical Markup Framework, LMF) bis zur Nutzung von Beschreibungslogik (z. B. OWL-DL) gehen. Die Aufgabenstellung bei der Entwicklung oder Wahl eines geeigneten Formalismus ist es, die Balance zwischen formaler Kontrolle und quantitativer Abdeckung (und damit effizienter Nutzbarkeit) zu halten, relativ zu den Zielanwendungen. Außerdem sollte ein multifunktionales Wörterbuch flexiblen Zugriff auf die enthaltenen Daten erlauben.

Repräsentationen und Formalismen für den Austausch lexikalischer und terminologischer Daten

Wegen der Entwicklungskosten für Wörterbücher tritt die Fragestellung des Austauschs und des Zusammenführens bestehender NLP-Wörterbücher, aber auch die Frage der verteilten Nutzbarkeit als Web-Service im Internet seit ca. 2005 in den Vordergrund des Interesses (vgl. z. B. das EU-Projekt CLARIN, dessen Ziel die Erstellung von linguistischen Web-Services für NLP-Werkzeuge und

Ressourcen ist). Ein Austausch oder eine Nutzung fremder Wörterbücher mit eigenen Werkzeugen setzt standardisierte Austauschformate voraus. Arbeiten zur Ressourcen-Interoperabilität zielen auf die Definition solcher Formate. Das Lexical Markup Framework (LMF) ist ein Metaformat dieser Art. Es definiert in Umrissen das Minimalinventar der Datentypen und ihre Zusammenhänge, als Grundgerüst von Modellen für NLP-Wörterbücher.

Die Entwicklung von Austauschformaten hat im Bereich der Terminologie eine längere Tradition als in der Lexikographie. Es gibt beispielsweise das *Terminology Markup Framework*, TMF, das, ähnlich wie LMF, die Prinzipien terminologischer Datenmodellierung beschreibt. Während für gemeinsprachliche NLP-Wörterbücher keine spezifischen Normen existieren, liegt mit dem Termbank-Austauschformat TBX ein spezifisches Austauschformat für Daten aus Terminologiedatenbanken vor, das von vielen Anbietern von Termbanksoftware unterstützt wird. Es gibt auch große Terminologiedatensammlungen (z. B. die EU-Datenbank IATE) in TBX-Kodierung.

Repräsentation lexikalischen Wissens für interaktiv benutzbare elektronische Wörterbücher

Bei interaktiv benutzbaren elektronischen Wörterbüchern stellen sich die meisten Repräsentationsfragen im Zusammenhang mit Fragen der Präsentation (wie sollen lexikalische Daten dem Benutzer angeboten werden?), des Zugriffs und der Gestaltung von graphischen Benutzerschnittstellen.

Auch hier gibt es datenbankbasierte, XML-basierte und spezifische dedizierte Systeme. Im Verlagswesen wird die Norm ISO-1951 (*Presentation/representation of entries in dictionaries – Requirements, recommendations and information*) diskutiert. Ein Fachverlag, der mehrere Wörterbücher komplett auf der Grundlage von ISO-1951 entwickelt hat, konnte z. B. durch parallele Nutzung von Suchsoftware und durch die Wiederverwendbarkeit von Teilen der lexikalischen Daten für andere Projekte oder Sprachpaare in erheblichem Umfang Entwicklungszeit und -kosten sparen.

ISO-1951 definiert XML-Modelle für inhaltlich definierte Angabetypen (z. B. grammatische Angaben, Bedeutungserläuterungen, Übersetzungsäquivalente usw.) und trennt diese von Auszeichnung, Layout und Formatierung. Damit wird auch das Publizieren für verschiedene Medien unterstützt (single source publishing).

5.2.5 Nutzung von lexikalischem Wissen

Die Nutzung lexikalischen Wissens in NLP-Werkzeugen ist in der Regel an die Bedingung der entsprechenden Verfahren und Werkzeugimplementierungen (Grammatiken, Parser, Tagger, etc.) geknüpft.

Bei interaktiven Wörterbüchern ist eine Abfrage über das Internet längst zum Standard geworden. Dabei gibt es zwei Trends: einerseits relativ einfache Wörterbücher mit einer großen Abdeckungsbreite, aber mit wenig lexikographischem Detail; andererseits sehr detaillierte elektronische Wörterbücher, die oft

als Bausteine von Sprachlernwerkzeugen konzipiert sind. Erstere zielen auf die gelegentliche zielgerichtete Suche, insbesondere nach Äquivalenzangaben (vgl. z. B. *LEO*), die zweiten auf das regelmäßige Browsing, zum Zweck von Sprach-training oder gezieltem Wortschatzlernen. Typische Beispiele der zweiten Sorte sind ELDIT für Deutsch/Italienisch oder BLF, die Base Lexicale du Français. In diesen Lern(er)wörterbüchern werden detaillierte Angaben zur Bedeutung, Syntax und Kombinatorik der Wörter gemacht. Interessanterweise scheint es schwer zu sein, die beiden Nutzungsarten (Browsing, im Gegensatz zu punktueller Suche, ggf. mit Constraints) über gemeinsame graphische Benutzerober-fächen zu bedienen. Die Lexikographie fängt erst an, Konzepte für interaktive Wörterbuch-Websites zu entwickeln. Die Möglichkeiten der Suche, der Daten-präsentation und der GUI-Gestaltung für interaktive Wörterbücher sind noch nicht ausgeschöpft. Kooperation zwischen Lexikographen, Computerlinguisten und Usability-Experten scheint hier notwendig.

5.2.6 Computerlinguistische Unterstützung lexikographischer Arbeit

Akquisition, Repräsentation und Präsentation von lexikalischen Daten können computerlinguistisch unterstützt werden. Zur Erleichterung der Arbeit von Wörterbuchentwicklern wird seit rund zehn Jahren an Lexikographie-Arbeitsplätzen gearbeitet, die Werkzeuge für die oben genannten Funktionen in einer gemein-samen Oberfläche vereinen.

Grundlage solcher Systeme ist eine (i.d.R. XML-basierte) Repräsentation von Wörterbuch-Inhalten. Diese wird durch einen spezifischen Editor zugänglich gemacht, der Funktionen für die Formatierung und für verschiedene Arten von Layout und Präsentation unterstützt. Solche Redaktionssysteme werden ggf. mit Akquisitions-Werkzeugen kombiniert, die z. B. Zugriff auf Konkordanzen oder auf Daten bestehender Wörterbücher geben.

Ein integriertes System dieser Art, das relativ bekannt geworden ist, ist *Sketch Engine* (Kilgarriff et al. 2004). Das System beruht auf regulären Grammatiken, die auf getaggte und lemmatisierte Korpora angewendet werden; die Suchverfahren sind mit Assoziationsmasken zur Identifikation von signifikanten Kook-kurrenzen kombiniert und liefern für ein gegebenes Wort die in einem gegebenen Korpus wichtigsten Kontextpartner, in der Form von nach Signifikanz sortierten Wortpaarlisten. Statt der manuellen Analyse von Konkordanzzeilen braucht der Lexikograph auf diese Weise nur die Wortpaardaten zu sichten, um typische Beispiele für Selektions- und Kollokationsphänomene zu identifizieren.

Die in Sketch Engine zugrundegelegte reguläre Grammatik ist für Englisch gut geeignet (z. B. genügt es, nach der ersten vollen Nominalphrase rechts des finiten Verbs zu suchen, um Verb+Objekt-Paare zu identifizieren). Sketch Engine wurde auf der Grundlage des BNC entwickelt und wird derzeit auf Web as Corpus (vgl. Baroni und Kilgarriff 2006) angewandt. Versionen von Sketch Engine exis-tieren auch für slawische Sprachen (Slowenisch, Tschechisch), d.h. flektierende Sprachen mit detaillierter, relativ wenig synkretistischer Kasusmorphologie. Für Deutsch müßte als Grundlage von Sketch Engine ein auf der Ebene von Depen-

denzstrukturen oder Prädikat-Argument-Strukturen geparses Korpus verwendet werden, damit trotz freier Wortstellung im Mittelfeld und synkretistischer Formen ordentliche Ergebnisse erzielen würden.

Sketch Engine kombiniert korpusbasierte Akquisition und Wörterbuch-Editierung. In Experimenten zum Deutschen wurden diese Funktionen auch mit Akquisitionswerkzeugen für die Nutzung vorhandener Wörterbücher kombiniert: für die Ergänzung von bestehenden Wörterbüchern wurden Funktionen entwickelt, die eine inhaltliche Ergänzung (um Daten zu Einzelwörtern, zu Mehrwortausdrücken, zur Subkategorisierung usw.) erlauben. Dabei werden die im bestehenden Wörterbuch vorhandenen Wörter und Wortgruppen mit dem Vokabular eines (oder verschiedener) Korpora verglichen. So werden automatisch Kandidaten zur Aufnahme ins Wörterbuch erzeugt.

Solche Werkzeuge erlauben auch den großflächigen Vergleich von Wörterbuchinhalten mit Korpusdaten (vgl. Evert et al. 2004).

5.2.7 Literaturhinweise

Es gibt kein Lehrbuch für Computational Lexicography. Im Rahmen der Handbuchreihe HSK, Handbücher zur Sprach- und Kommunikationswissenschaft, (Verlag Mouton de Gruyter, Berlin) ist aber ein Handbuchband in Vorbereitung, der die Computational Lexicography miterfasst (HSK-5/4, Wörterbücher, Supplementband, erscheint Ende 2010 oder 2011). Die aktuelle Forschung des Bereichs kann in den Konferenzen EURALEX und LREC (Linguistic Resources and Evaluation Conference), die beide in geraden Jahren stattfinden, verfolgt werden. Die Interessengruppe SIGLex (Special Interest Group on the Lexicon) der ACL hat seit ca. 1998 regelmäßig Workshops zu Mehrwortausdrücken, ihrer Extraktion aus Korpora, ihrer automatischen Klassifikation usw. veranstaltet (Proceedings im Netz). Das Einführungsbuch von Engelberg und Lemnitzer (2008) zur Lexikographie enthält auch kurze Beschreibungen von ausgewählten Aspekten der Computational Lexikography.

5.3 Text-basiertes Informationsmanagement

Günter Neumann

5.3.1 Überblick

Dank der Infrastruktur des Internets und des World Wide Webs (WWW) steht aktuell eine quasi schier unbegrenzte Menge an textuellen Quellen in Form von freien Texten und semistrukturierten Dokumenten oder Strömen solcher Quellen zur Verfügung. Dies umfasst nicht nur die mittlerweile klassischen Quellen, wie z. B. Webseiten, Emails, Nachrichtenartikel, wissenschaftliche Publikationen, Bücher, etc., sondern auch sehr dynamische Textquellen, wie Blogs, Diskussionsforen oder kollaborative Portale, wie Wikipedia und andere Wikis (siehe hierzu auch Abschnitt 4.7).

Der Einsatz von Suchmaschinen im WWW (zusammen mit Annotationen und Verknüpfungen in den Dokumenten) ist aktuell die gängigste Methode, diese textuellen Quellen für den eigenen Informationsbedarf inhaltlich zu erschließen. Dabei ist die Formulierung von Informationsbedürfnissen und die inhaltliche Erschließung immer noch recht beschränkt. Im Wesentlichen geschieht dies durch Angabe von Schlüsselwörtern, deren Bestimmung in Textquellen und durch Berechnung einer möglichst optimalen Anordnung der Treffer, d. h., der gefundenen Dokumente. Gerade im Kontext des globalen, multilingualen Webs sind die wissenschaftlichen und technischen Herausforderungen auch hierbei noch enorm (vgl. hierzu Belew 2000 und Manning, Raghavan und Schütze 2008).

Allerdings liegt der Schwerpunkt der inhaltlichen Erschließung und die damit verbundene kognitive Last immer noch beim Benutzer solcher Technologien. Er muss letztlich entscheiden, ob die gelieferten Dokumente tatsächlich relevant und hilfreich sind und die gesuchte Information enthält oder zumindest Hinweise darauf. Und er muss sich letztlich selber „Gedanken machen“, wie eine bessere Suchanfrage zu formulieren ist, damit eine möglichst exakte Beantwortung seiner Informationsanfrage möglich wird. Aufgrund der stetigen Zunahme an textuellen Dokumenten (insbesondere auch durch das Web 2.0) ist es aber ein zentrales Problem, dass die Informationssuche- und -sichtung einen erheblichen Zeitaufwand mit sich bringt und letztlich auch aus wirtschaftlicher Perspektive hohe Kosten erzeugt („Time is money“). Um die Informationsflut einigermaßen in den Griff zu bekommen, erscheint ein gezielter, fokussierter Zugang auf die inhaltlichen Informationen von Texten nicht nur hilfreich, sondern notwendig.

Im Bereich der Sprachtechnologie sind hierbei in den letzten Jahren eine Reihe von Konzepten und Technologien exploriert und entwickelt worden, insbesondere in den Bereichen des Text Minings, der Informationsextraktion und -integration, der semantischen Suche und Fragebeantwortung und der Informationspräsentation. Aktuell ist eine starke Konvergenz zwischen diesen Bereichen zu beobachten, insofern als viele Teillösungen gemeinsamen Ursprung haben, wie z. B. die linguistische Merkmalsextraktion oder die Identifikation von relevanten Entitäten und Relationen zwischen ihnen.

Diese Konvergenz unter der Bezeichnung **Text-basiertes Informationsmanagement (TIM)** aufzuzeigen und zu beschreiben ist Gegenstand dieses Unterkapitels.

Information

Um besser verstehen zu können, was „Text-basiertes Informationsmanagement“ meint, wollen wir kurz klären, wie wir im Kontext dieses Artikels den Begriff „Information“ verstehen. Man muss hierbei vorausschicken, dass dies in der Tat keine einfache Sache ist, da es noch keine eindeutige und zufriedenstellende Definition des Begriffes gibt (vgl. Meadow 1992 und Floridi 2005). Wir werden daher den Begriff hier nur soweit klären, wie er für das weitere Verständnis wichtig ist.

In den letzten Jahrzehnten wird meist eine Definition von **Information** als „Daten + Bedeutung“ bevorzugt, vgl. Floridi (2005). Vereinfacht betrachten wir „Information“ als eine Menge von diskreten, wohlstrukturierten und bedeutungsvollen Daten. Daten sind quasi die physikalischen Implementierungen von Symbolen (z. B. in einer Datenbank oder als Druckertinte auf Papier); **wohlgeformt** meint, dass die Daten in einer Struktur organisiert sind; **bedeutungsvoll** meint, dass diese organisierte Anordnung von Daten von einem Benutzer gelesen und verstanden werden kann. Unter **Lesen** wollen wir verstehen, dass die Datenquelle von einem Benutzer mit Hilfe seiner Sensoren empfangen, internalisiert und strukturiert werden kann, sodass auf deren Basis dann der eingebettete Inhalt bestimmt werden kann.

An einem einfachen Beispiel wollen wir das kurz klären. Betrachten wir das Symbol „28081749“. Ohne weitere „Information“ können wir dies als eine Zeichenfolge bestehend aus natürlichen Zahlen „verstehen“. (Aber aufgemerkt: Damit wir dies tun können, müssen wir natürlich „wissen“, was „natürliche Zahlen“ und auch „Zeichenfolgen“ sind.) Allerdings kann man diese Zeichenfolge auch interpretieren als „Goethes Geburtsdatum“ (Goethe wurde am 28. August 1749 geboren) oder auch als Breitengrad südlich von „New Delhi“. Das „28081749“ Goethes Geburtsdatum meint, mag man auch dadurch ableiten, dass das Symbol etwa in einem Satz wie „Goethe wurde am 28081749 geboren.“ vorkommt oder direkt als Element in einer Datenbank gepeichert ist, z. B. als Attribut-Wert-Paar $\langle \text{Geburtsdatum} = 28081749 \rangle$, das wiederum Teil eines komplexeren Eintrags über Wolfgang von Goethe sein kann.

Dieses Beispiel macht deutlich, dass Daten mehrere Interpretationen haben können und nur eindeutig interpretierbar sind, wenn Wissen und Kontext herangezogen wird, vgl. auch Abschnitt 4.6.2. Da Texte bekanntermaßen hochgradig mehrdeutig sind, ist das Informationspotential in Texten entsprechend hoch. Wir wollen daher als **relevante Information** den Anteil an der potentiellen Information bezeichnen, der von einem Benutzer als „nützlich“ betrachtet wird. Man beachte, dass die Bewertung, ob Information relevant ist oder nicht, benutzer- und situationsabhängig ist, wogegen „Information“ dies per se nicht ist, d. h., Information kann auch irrelevant sein.

Wichtig ist aber zu beachten, dass die Information stets eingebettet in einer physikalischen Implementation auftritt, sei es als Datenbank, als Bild, als Video oder eben als Text.

Text-basiertes Informationsmanagement – ein Walkthrough

Wesentliche Hauptaufgaben eines Informationsmanagementsystems sind:

- Die Information, die in digitalisierter Form als Daten repräsentiert ist, zu verwalten.
- Die zur Beantwortung einer Anfrage notwendige relevante Information zu bestimmen und einem Benutzer in verständlicher Form zur Verfügung zu stellen.

Im Falle eines TIM-Systems ist Information durch natürliche Sprache in Texten kodiert und muss entsprechend in Daten überführt werden. Je nachdem, wie tief und umfassend diese Strukturierung vorgenommen wird, muss auch eine Benutzeranfrage entsprechend interpretiert und strukturell analysiert werden.

Wir wollen an einem komplexeren Beispiel die wichtigsten Eigenschaften eines TIMs zeigen, die dann im späteren Verlauf des Artikels detaillierter beschrieben werden. Folgende Situation sei für das Beispiel angenommen: Ein Benutzer des TIMs – nennen wir sie im weiteren USER – möchte Informationen zu Umsatzmeldungen von Unternehmen aus der IT Branche sammeln, vielleicht, weil sie selber ein Unternehmen gründen möchte, oder weil sie diese Informationen nutzen möchte, um versteckte Beziehungen über mögliche Verflechtungen von Unternehmen zu bestimmen. Nehmen wir zusätzlich an, USER verfüge bereits über eine Datenbank mit Instanzen solcher Informationen (und gegebenenfalls Verweise auf die textuellen Quellen). Folgendes Beispiel zeigt einen Eintrag aus dieser Datenbanktabelle namens UMSATZMELDUNGEN:

Unternehmen	Jahr	Größe	Betrag	Tendenz	Differenz
Compaq	1998	Umsatz	31 Mrd. USD	+	27%

Hierbei repräsentiert das Attribut *Unternehmen* den Namen eines Unternehmens, *Jahr* das Jahr der Meldung und *Betrag* den mitgeteilten Umsatzwert; das Attribut *Größe* kodiert, ob es sich um einen Umsatz, Gewinn oder Verlust handelt und *Tendenz* entweder eine Zunahme oder Abnahme beschreibt. *Differenz* kodiert den prozentualen Unterschied zum Vorjahr.

Ziel ist es nun, durch Analyse von Webdokumenten automatisch diese Tabelle zu erweitern oder zu aktualisieren. Welche Schritte sind hierzu im Einzelnen notwendig? Die wohl wichtigsten sind:

1. Analyse der Informationsanfrage
2. Bestimmung von relevanten Dokumenten
3. Bestimmung der relevanten Textpassagen
4. Extraktion von Attributwerten und -beziehungen
5. Erzeugen neuer Tabelleneinträge

6. Integration in bestehende Tabellen

7. Präsentation der Ergebnisse

Wir werden nun die einzelnen Schritte an Hand unseres obigen Beispielszenarios durchspielen, wobei wir annehmen, dass es bereits ein System gibt (mit Namen **TIMPLE – Text-based Information Management Example**), dass die komplette Aufgabe automatisch löst, sobald USER eine Informationsanfrage an das System sendet. Nehmen wir an, USER formuliere ihre Anfrage direkt als natürlichsprachliche Frage „Welche Firmen aus der Computerbranche steigerten ihren Umsatz?“. Als ersten Schritt analysiert TIMPLE die Anfrage derart, dass eine Suche nach relevanten Dokumenten gestartet werden kann. Nehmen wir an, dass zur Dokumentenbestimmung TIMPLE eine oder mehrere Suchmaschinen einsetzt, etwa Ask (<http://de.ask.com/>), Google (<http://www.google.de>), MSN Live-Search (<http://www.live.com>) oder Yahoo (<http://de.yahoo.com>). Dann ist die einfachste Art der Analyse, die Frage direkt an eine Suchmaschine weiterzuleiten. Die Antwort ist dann in Form einer sortierten Liste von Treffern, etwa wie folgt (Es handelt sich hierbei tatsächlich um das Ergebnis der obigen Anfrage, die der Autor am 20.11.2008, ca. 15:00 Uhr durchgeführt hat. Es werden die ersten drei von 153 Treffern, in der Darstellung entsprechend angepasst, gelistet.):

1. Dell hängt Rest der **Computerbranche** ab – Unternehmen – IT ...

Die **US-Firma** **steigerte** im zweiten Quartal zwar den **Umsatz** um 8,1 % auf 1,55 ... hängt stark von der Entwicklung der Weltwirtschaft und **ihrem** Einfluss auf ...

<http://www.handelsblatt.com/unternehmen/it-medien/dell-\haengt-rest-der-computerbranche-ab;646067>

2. PC-Hersteller bleiben auf Wachstumskurs – Business | ZDNet.de News

Die Texaner **steigerten** **ihren** Gewinn im Vergleich zum Vorjahr um ein Viertel auf 846 Millionen Dollar (656 Millionen Euro). Der **Umsatz** kletterte um 18 ...

<http://www.zdnet.de/news/business/0,39023142,39127741,00.htm>

3. TAM – Energienews täglich aktuell – Energienachrichten per ...

In den ersten neun Monaten 2008 **steigerte** die schwedische Vattenfall wegen gestiegener Preise **ihren Umsatz** um knapp 12% ...

<http://www.tam.de/index.asp?ACTION=1>

Jeder Treffer besteht aus einer Textüberschrift, einem Textausschnitt und dem Link zur Webquelle. Wie bestimmt eine Suchmaschine die Dokumente? Nun, unter anderem auch dadurch, dass es einen wortbasierten Index für alle verfügbaren Webdokumente verwaltet. Damit ist es möglich, Dokumente dadurch auszuwählen (zu „erschließen“, engl. **document retrieval**), indem man im Index unter den Suchwörtern nachschlägt. Im Textausschnitt (auch: *Snippet*), werden die Wörter, die mit denen aus der Anfrage übereinstimmen, hervorgehoben. Man beachte, dass nicht alle Wörter aus der Frage auch in den Dokumenten vorkommen müssen. So kommt das Wort „Computerbranche“ nicht in den Treffern 2

und 3 vor. Man sieht aber auch, dass morphologische Varianten eines Wortes (z. B. „ihren“ in der Anfrage und „ihrem“ in Treffer 1) scheinbar keine Probleme bereiten.

Obwohl für dieses Beispiel nur eine kleine Dokumentenmenge bestimmt wurde (153 Dokumente), sind offensichtlich nicht alle Dokumente gleichermaßen relevant. So thematisiert Treffer 3 Umsatzmeldungen von Unternehmen aus dem Energiesektor. Daher **klassifiziert** Timple die Dokumente (nachdem es diese mittels des Weblinks heruntergeladen und in eine uniforme interne Dokumentenrepräsentation überführt hat) in solche, die zur Klasse „Computerbranche“ gehören und solche, die zu anderen Klassen gehören. Es ist offensichtlich, dass dies zwar ein semantischer Prozess ist, in der Regel aber auf einer sehr großen Textmenge durchzuführen ist und als Vorauswahl für die nachfolgenden Informationsextraktionprozesse dient. Daher muss diese Klassifikation sehr robust, schnell und dennoch möglichst genau durchgeführt werden. Timple ist zum Glück dazu in der Lage, da es auf der Basis von früheren Analysen automatisch gelernt hat, welche Wörter und wortbasierten Merkmale helfen, die Klassenzuordnung durchzuführen.

Nehmen wir an, dass auf diese Weise klar ist, welche Dokumente „Umsatzmeldungen“ enthalten und welche nicht. In der Regel wird ein solches Dokument aber nicht nur über Textpassagen verfügen, in denen die Informationen über Umsatzmeldungen gebündelt vorkommen, sondern der Text wird auch andere Informationen kodieren, die aus Sicht der Umsatzmeldung irrelevant sind. Daher ist es ein sinnvoller nächster Schritt für Timple die Textpassagen zu identifizieren, die den Umsatzmeldungen „unmittelbar“ entsprechen. Im Prinzip bedeutet dies, dass ein Text in eine Folge von Texteinheiten zu überführen ist, sodass für jede Texteinheit beurteilt werden kann, ob sie relevant oder nicht relevant zur Beantwortung der Informationsanfrage ist. Oder anders formuliert, es sollen genau die **Textpassagen bestimmt werden** (engl. **passage retrieval**), in denen potentielle Werte für Attribute von Umsatzmeldungen zu finden sind. Betrachten wir als Beispiel hierfür den Inhalt des Textes zum Treffer Nummer 1 (aus Platzgründen werden nur die ersten 3 der insgesamt 8 Paragraphen gezeigt, aus denen das Originaldokument besteht):

<http://www.handelsblatt.com/>, 18.07.2003

Größter Gewinner ist einmal mehr der Computerbauer Dell. Die Amerikaner sind weltweit die Nummer eins und wesentlich stärker gewachsen als der nächste Verfolger, Hewlett-Packard. „Die Kunden honorieren unser effizientes Geschäftsmodell, das auf Händler ganz verzichtet“, sagte der Deutschland-Chef von Dell, Mathias Schädel, dem Handelsblatt. Das US-Unternehmen vertreibt seine Produkte ausschließlich über eigene Mitarbeiter, zumeist per Internet.

Doch nicht nur Dell kann zufrieden sein. *Eine Mixture aus wachsendem Dienstleistungsgeschäft, Kostensenkungen und erfolgreichen Akquisitionen brachte Wettbewerber IBM im zweiten Quartal deutlich verbesserte Ergebnisse. Zwischen April und Juni stiegen der Umsatz*

um 10% auf 21,6 Mrd. \$ und der Reingewinn auf 1,7 Mrd. \$. Sonderlasten in Höhe von 1,4 Mrd. \$ hatten den Vorjahresgewinn auf 56 Mill. \$ gedrückt.

Der kultige Nischenanbieter Apple dagegen steht nicht ganz so gut da. Die US-Firma steigerte im zweiten Quartal zwar den Umsatz um 8,1% auf 1,55 Mrd. \$. Der Nettogewinn ging jedoch von 32 auf 19 Mill. \$ zurück. Apples Ergebnisse übertrafen damit die pessimistischen Erwartungen der Analysten. Apple gilt seit langem als innovativ und experimentierfreudig, der weltweite Marktanteil lag 2002 aber nur noch bei 2,3% gegenüber 8,3% vor zehn Jahren.

Die kursiv hervorgehobenen Stellen seien die relevanten Textpassagen. Wie könnte Timple diese Aufgabe leisten? Zum Einen scheint klar, dass Timple Texte in Einheiten, wie Zeichenfolge, Sätze, Paragrafen etc. strukturieren können muss. Zum Anderen scheint aber auch klar, dass Timple „wissen muss“, wie die Struktur der Datenbanktabelle UMSATZMELDUNGEN definiert ist, d. h., es muss den Typ der Attribute kennen und die Stelligkeit der Tabelle. Die Tabelle kann auch als eine n-äre Relation betrachtet werden, wobei die einzelnen Spalten den Argumenten entsprechen. Die konkrete Relation unserer Tabelle entspricht demnach der 6-stelligen Relation

UMSATZMELDUNGEN(UNTERNEHMEN, JAHR, GRÖSSE, BETRAG,
TENDENZ, DIFFERENZ)

Das bedeutet, dass Timple nicht nur diese Relation „wissen“ muss, sondern auch „operationalisieren“ können muss. Was heißt das? Nun, Timple muss bereits bei der Bestimmung von relevanten Textpassagen zumindest teilweise Wissen über diese Relation einsetzen, damit überhaupt relevante Textausschnitte bestimmt werden können. Beispielsweise sollte in einer relevanten Textpassage zumindest der Name eines Unternehmens (z. B. „Apple“) explizit oder zumindest implizit („die US-Firma“) genannt werden.

Daher betrachtet Timple eine **Relation** als eine Komposition von diskreten Elementen, die es versucht, von den einfachen Elementen bis zu deren komplexer Vernetzung zu identifizieren und zu organisieren. Dies umfasst zuerst die Bestimmung aller potentiellen Werte für die verschiedenen **Attribute** (Argumente) der Relation. Unserem Beispiel folgend müssen z. B. alle möglichen Firmennamen oder Geldbeträge im gesamten Text bestimmt werden, bevor in einem nächsten Schritt entschieden werden kann, welche der Kandidaten tatsächlich Teil welcher Relation sind. Im Beispieltext werden die Computerfirmen „Dell“, „Hewlett-Packard“, „IBM“ und „Apple“ genannt. Erst später wird sich zeigen, dass aber nur „IBM“ und „Apple“ von Relevanz sind, da „Dell“ und „Hewlett-Packard“ nicht als Wert in einer relevanten Relation genannt sind. Die Erkennung solcher spezifischen **Eigennamen** ist nicht trivial, d. h., nicht einfach durch einen direkten Zugriff auf eine eventuell vorhandene Liste aller Firmennamen zu realisieren. Zum Einen kann es sein, dass im Text verschiedene Schreibvarianten eines Firmennamens verwendet werden (z. B. „Apple“, „Apple Inc.“, „Apple Deutschland“ etc.). Zum Anderen könnten Firmennamen auch implizit genannt werden, z. B.

„Apple“ implizit durch die definite Nominalphrase „der US-Konzern“. Das bedeutet, dass zur Erkennung von Eigennamen nicht nur alle Benennungen von Firmennamen zu identifizieren sind, sondern diese auch korrekt den denotierten Entitäten zuzuordnen sind.

Es erscheint klar, dass zumindest für die **Bestimmung von Koreferenzen** zwischen Eigennamen und Nominalphrasen eine zumindest partielle syntaktische Analyse notwendig ist, die über die Behandlung von Phänomenen auf der Wortebene hinausgeht; z. B. müssen nicht nur die Nomengruppen erkannt werden, sondern auch ihre interne Struktur, wie etwa Spezifikator-Kopf-Kongruenz. Dies wird insbesondere dann deutlich, wenn die potentiellen Attributwerte zu komplexeren Einheiten als Relation verknüpft werden müssen. Eine mögliche Strategie hier ist anzunehmen, dass alle Attributwerte, die zu einer Relation gehören, in **struktureller Nähe** zueinander auftreten. Daher wollen wir annehmen, dass Timple zur weiteren Bestimmung von Relationen eine syntaktische Analyse aller relevanten Sätze durchführt, also aller Sätze, in denen mindestens ein Attributwert vorkommt. Darauf aufbauend können dann mit Hilfe von relationenspezifischen Mustern (etwa der Art „NP(FN) VP(STEIGEN(UMSATZ)) PP(UM, NP(B))“, wobei z. B. NP(FN) zu Lesen ist als „Nominalphrase deren Kopfelement vom Typ Firmename ist“) die Argumentwerte bestimmt und zugeordnet werden, die zu einer Relation gehören.

Allerdings kann im Allgemeinen nicht angenommen werden, dass stets immer alle Werte einer Relation in einem Satz auftreten. Daher muss Timple auch in der Lage sein, **Teilrelationen** zu identifizieren, wie z. B. für den Textabschnitt:

Eine Mixtur aus wachsendem Dienstleistungsgeschäft, Kostensenkungen und erfolgreichen Akquisitionen brachte Wettbewerber **IBM** im zweiten Quartal deutlich verbesserte Ergebnisse. Zwischen April und Juni **stiegen der Umsatz um 10% auf 21,6 Mrd. \$** und der Reingewinn auf 1,7 Mrd. \$.

Die entsprechenden **partiellen Instanzen** sehen wie folgt aus:

Unternehmen	Jahr	Größe	Betrag	Tendenz	Differenz
IBM					
		Umsatz	21,6 Mrd. \$	+	10 %

Nachdem Timple also alle potentiellen möglicherweise partiell instanzierten Relationen berechnet hat, muss in einem nächsten Schritt entschieden werden, welche Teilrelationen zusammen gehören und wie die identifizierten Teile **fusioniert** werden sollen. Für das obige Beispiel wählt Timple eine einfache, aber bewährte Heuristik: Es werden die Attributwerte von aufeinander folgenden Sätzen vereinigt, solange keine Inkonsistenzen entstehen. Für das obige Beispiel bedeutet dies, dass die zwei partiellen Einträge einfach verschmolzen werden können zu:

Unternehmen	Jahr	Größe	Betrag	Tendenz	Differenz
IBM	2003	Umsatz	21,6 Mrd. \$	+	10 %

Man beachte, dass TIMPLE hierbei auch den Wert für das Jahr eingefügt hat, obwohl dieser nicht explizit im Text genannt wird, wohl aber in der Kopfzeile des Artikels. Es ist klar, dass dies nur eine einfache (allerdings erprobte) Heuristik ist, da im Allgemeinen hier komplexere Inferenzprozesse nötig sind, um den richtigen zeitlichen Kontext einer Relation zu bestimmen. Da TIMPLE aber stets unter großem Druck steht, die relevanten Informationen möglichst zeitnah zu liefern, muss TIMPLE oft einen Kompromiss zwischen Aktualität und Genauigkeit finden.

Bevor diese neuen Informationen aber an USER weitergeleitet werden, integriert TIMPLE die gefundenen Instanzen als neue Einträge in USERS bestehende Datenbank UMSATZMELDUNGEN. Auch dies ist kein trivialer Prozess, da die Konsistenz der gesamten Datenbank nicht beschädigt werden darf, u.a., durch doppelte Einträge, durch sich widersprechende Einträge oder durch partielle Einträge.

Nehmen wir an, dass TIMPLE auch diesen Schritt meistert, dann wäre es so weit, USER die neuen Informationen mitzuteilen. Eine einfache Meldung könnte sein: „Lieber USER, es gibt 50 neue Einträge in der Datenbank UMSATZMELDUNGEN.“ Solch eine Meldung kann nützlich sein, ist in der Regel aber suboptimal. TIMPLE beschließt daher, automatisch eine verständliche Zusammenfassung der neuen Einträge als Ergebnis zu liefern, mit direkter Verbindung zu den Einträgen in der Datenbank, etwa durch folgende **Textzusammenfassung**:

Hallo USER! Ich habe 50 neue Einträge in die Datenbank UMSATZMELDUNGEN eingefügt. Für folgende Firmen liegt eine Umsatzsteigerung im Vergleich zum Vorjahr 2002 vor: Dell, IBM, Folgende Firmen verzeichnen dagegen einen Verlust: Sugar&Brothers, SaltInc., , ...

Dies ist natürlich nur eine mögliche Textzusammenfassung. Eine andere Möglichkeit wäre es, direkt die relevanten Textausschnitte heranzuziehen und kohärent zusammenzufügen, eventuell durch Verwendung von Paraphrasen. Auch sind wir im gesamten Beispiel davon ausgegangen, dass die relevanten Dokumente im gesamten Web gesucht werden. Hier wäre es auch möglich, direkt (zumindest für die konkrete Anfrage) Geschäftsberichte von Firmen zu analysieren oder spezielle Nachrichtenmagazine.

Des Weiteren gingen wir davon aus, dass die Informationsanfrage in Form einer vollständigen **natürlichsprachlichen Frage** formuliert ist und dass TIMPLE dies schon entsprechend interpretieren kann. Hierzu muss TIMPLE jedoch „verstehen“, dass mit der Frage das Auffüllen der Tabelle UMSATZMELDUNGEN „gemeint“ ist. Dies könnte TIMPLE tatsächlich auf verschiedene Weise leisten. Eine Möglichkeit ist es, zum Einen den **FrageTyp** (es soll eine Liste von Elementen eines bestimmten Typs bestimmt werden) und zum Anderen den **erwarteten Typ der Antwort** (hier Relation UMSATZMELDUNG) zu bestimmen. Letzteres könnte einfach dadurch realisiert werden, dass das Tabellschema direkt mit bestimmten Schlüsselwörtern assoziiert ist. Wenn die Schlüsselwörter (oder Varianten davon) in der Frage auftreten, wird das entsprechende Schema aktiviert.

In der Regel wird TIMPLE aber über viele verschiedene solcher Tabellenschemata verfügen, so dass es sinnvoll erscheint, diese systematisch in einem strukturierten Vokabular (**Ontologie**, vgl. auch Unterkapitel 4.6) zu verwalten. Daher könnte die Interaktion mit einem Benutzer auch so gestaltet werden, dass TIMPLE diese Ontologie dem Benutzer visualisiert, so dass der Benutzer einen relevanten Teilausschnitt eigenständig bestimmen kann und dann in diesem Kontext der Ontologie seine aktuelle Informationsanfrage formuliert, z. B. indem der Benutzer für einige Attribute Werte vorgibt (z. B. bestimmte Firmennamen oder Quartalsangaben etc.).

Eine weitere Möglichkeit, mit TIMPLE zu interagieren, wäre es, dem System eine Menge von Texten zu geben, die bereits mit Beispielen der gewünschten Informationen manuell annotiert sind, in unserem Beispiel also mit Relationen zu Umsatzmeldungen. TIMPLE kann dann dieses **Korpus** heranziehen, um mit Hilfe von **maschinellen Lernverfahren** automatisch zu erlernen, neue Texte als Umsatzmeldungen zu erkennen und die entsprechenden Attributwerte und Relationen zu finden und zu extrahieren.

All diesen verschiedenen Möglichkeiten ist gemein, dass sie die gleichen Basisaufgaben zu lösen haben, u. a. Volltextsuche, Klassifikation von Texten, Bestimmen von relevanten Textausschnitten, linguistische Merkmalsextraktion, Extraktion von Eigennamen und Relationen, sowie deren Fusion und Integration in bestehende Datenbestände. In den folgenden Abschnitten wird auf die verschiedenen Basistechnologien genauer eingegangen und es werden aktuelle Methoden und Verfahren kurz vorgestellt.

TIM-Grobstruktur

Die Grafik in Abb. 5.1 zeigt die grobe Struktur eines (ideellen) generischen TIM. Ein TIM vereinigt in sich die verschiedenen Perspektiven des Information-Retrieval (IR), der Informationsextraktion (IE) und der Fragebeantwortung (QA – Question Answering).

Die Hauptaufgaben im Bereich IR liegen in der Indizierung und Suche im gesamten Dokumentenbestand (**Volltextsuche**), in der Textklassifikation und im Textclustering (**Textgruppierung**). Der Schwerpunkt im Bereich QA liegt in einer genauen Analyse der Benutzeranfrage (**Frageanalyse**) und in der Extraktion von exakten Antworten (**Antwortextraktion**) aus relevanten Textausschnitten. Im Bereich der IE liegt der Forschungsschwerpunkt in der Extraktion von **Eigennamen** und **Relationen** und in der Gruppierung und Fusion gefunderner Entitäten.

Für alle drei Bereiche zentral ist natürlich eine textuelle Dokumentenbasis und vor allem die Extraktion von linguistischen Merkmalen.¹ Allerdings unterscheiden sich die Bereiche in der Größenordnung der Dokumentenmenge, die sie unmittelbar zu verarbeiten haben und damit auch in der Tiefe und Genauigkeit der linguistischen Analyse, die noch machbar ist. Für das IR gilt, dass hier sehr viel größere Datenmengen zu verwalten sind (z. B. Milliarden von Webseiten),

¹Vergleiche hierzu Kapitel 3, insbesondere die Unterkapitel 3.3, 3.4, 3.5 und Abschnitt 3.7.2 und aus Kapitel 4, Unterkapitel 4.3.

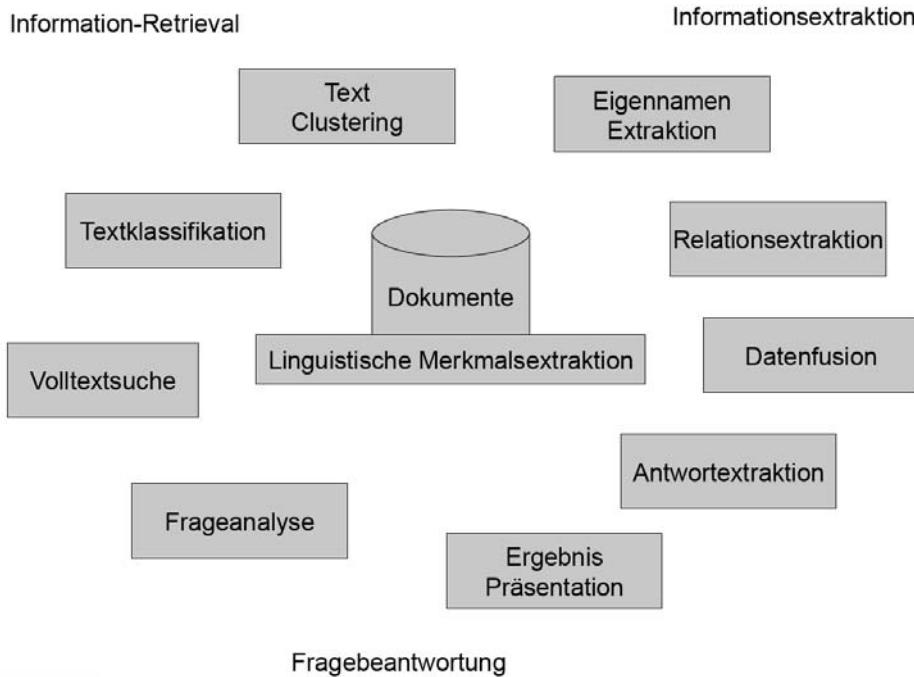


Abbildung 5.1: Text-basiertes Informationsmanagement: Technologieperspektive. Genaue Erläuterungen finden sich im Text.

wogegen die Dokumentenmenge für die IE und QA meist überschaubar ist (z. B. einige tausend Dokumente). Daher basieren die in der IR eingesetzten Komponenten in der Regel auf wortbasierten Merkmalen. In der QA und insbesondere in der IE ist jedoch eine umfangreichere linguistische Merkmalsextraktion notwendig, wie das obige Beispiel verdeutlicht.

Auf der anderen Seite bauen die verschiedenen Technologiezweige in immer stärkerem Maße aufeinander auf. So verwenden viele QA-Systeme direkt IR-Systeme zur Vorauswahl einer relevanten Textmenge. Dabei setzen sie auch unmittelbar Technologien aus dem Bereich der IE ein, wie z. B. die Extraktion von Eigennamen und Relationen. IE-Systeme setzen verstärkt Technologien aus dem Bereich des IR ein, insbesondere zur Bestimmung relevanter Textausschnitte und zur Gruppierungen von Daten.

Evaluationskriterien für TIM

Um die Güte der einzelnen Komponenten zu messen, werden oft zwei Maße, die **Präzision** und die **Vollständigkeit**, verwendet, die im Folgenden beschrieben werden:

- Die Präzision (engl. **Precision** (P)) bezeichnet den Anteil der korrekt gewonnenen Wissenseinheiten (WE) (Sätze oder einzelne Attribut-Wert-

Paare) im Vergleich zu den insgesamt gefundenen WE. Eine hohe Präzision bedeutet daher, dass fast alle gefundenen WE relevant sind, während eine geringe Präzision besagt, dass auch viele nicht relevante WE fälschlicherweise als relevant beurteilt wurden.

- Die Vollständigkeit (engl. **Recall (R)**) bezeichnet den Anteil der korrekt gewonnenen WE im Vergleich zu den insgesamt gewinnbaren WE. Bei einer geringen Vollständigkeit wurden viele relevante WE übersehen, während bei einer hohen Vollständigkeit fast alle relevanten WE extrahiert wurden.

Es ist schwierig, beide Parameter gleichzeitig zu optimieren: Wird eine Suche auf eine hohe Präzision hin optimiert, so steigt die Wahrscheinlichkeit, dass möglicherweise relevante Wissenseinheiten nicht erkannt werden. Optimiert man andererseits die Vollständigkeit, so steigt die Gefahr, dass Wissenseinheiten mit in das Ergebnis aufgenommen werden, die irrelevant sind. Um ein zusammenfassendes Maß für die Güte einer Methode zu schaffen, wurde das **F-Maß** definiert (in der Regel wird in folgender Gleichung $\beta=1$ gesetzt):

$$F = \frac{(\beta^2 + 1) * P * R}{\beta^2 P + R}$$

Die Güte des Ergebnisses hängt stark von der Schwierigkeit der Aufgabe ab. So berichten beispielsweise Grishman und Sundheim (1996) im Rahmen der MUC-7, dass bei einer einfachen Aufgabe, wie der Erkennung von Namen, die meisten Systeme sowohl bei der Vollständigkeit als auch bei der Präzision Werte von über 90% erreichten (das beste System erreichte $R=96\%$ und $P=97\%$). Allerdings war bei dieser Aufgabe die Domäne der Texte stark eingeschränkt, da es sich ausschließlich um Texte aus dem Wall Street Journal handelte.

Appelt und Israel (1997) weisen darauf hin, dass empirisch festgestellt wurde, dass bei komplexen Aufgaben im Bereich der Informationsextraktion die Güte des Ergebnisses in der Regel nicht besser als 60% bezüglich des F-Maßes ist. Eine solche Güte ist jedoch nicht so schlecht, wie es auf den ersten Blick erscheint, denn auch Menschen sind nicht in der Lage, bei der Analyse von komplexen Texten sowohl 100% Vollständigkeit als auch 100% Präzision zu erreichen. Ein Mensch erreicht bei der Erkennung von Eigennamen ein F-Maß von etwa 0,97%. Darüber hinaus arbeiten IE-Systeme im Gegensatz zu Menschen ermüdungsfrei, während die menschliche Leistung infolge Ermüdungserscheinungen nachlässt.

Die Maße Precision und Recall können auch aus der folgenden **Kontingenztafel** abgeleitet werden, die im Bereich des Information-Retrievals häufig herangezogen wird:

	D ist relevant	D ist nicht relevant
D ist gefunden	a	b
D ist nicht gefunden	c	d

Hierbei meint „D ist relevant“ die Anzahl der Dokumente (oder Eigennamen oder Relationen), die relevant sind, und „D ist gefunden“ die Anzahl der Dokumente, die ein System als Ergebnis geliefert hat (analoges für „nicht“). Dann ist z. B. a

die Anzahl der Dokumente die relevant sind und vom System geliefert wurden und c die Anzahl der relevanten Dokumente, die vom System nicht gefunden werden konnte. Die Summe von a, b, c und d ist die Gesamtanzahl aller Dokumente, die in einem Test ausgewertet werden. Damit entspricht Precision P der Anzahl $P=a/(a+b)$ und Recall R entspricht $R=a/(a+c)$. Desweiteren lässt sich die Akkuratheit A eines Systems berechnen durch $A=(a+d)/(a+b+c+d)$. Dies setzt natürlich voraus, dass man von jedem Dokument in einem Korpus weiß, ob es relevant ist oder nicht.

Ein weiteres häufig benutztes Maß speziell im Bereich der Fragebeantwortung ist der **Mean Reciprocal Rank** (MRR). Der **MRR** weist dort jeder Frage einen Wert zu, der gleich dem Kehrwert des Ranges der ersten korrekten Antwort der N besten Kandidaten ist (vgl. auch Abschnitt 5.3.4).

Im Gegensatz zur Akkuratheit betrachtet der MRR also mehrere Antwortkandidaten des Systems, wobei berücksichtigt wird, ob die richtige Antwort an erster, zweiter oder anderer Stelle steht. Wenn N=1 gewählt wird, entspricht das Ergebnis demnach der Akkuratheit des Systems. Der MRR ist zwar ein weicheres, allerdings aus Anwendungsperspektive praktikableres Maß als die Akkuratheit, z. B. in Fällen, wo ein Benutzer selbst die passende Antwort aus einer Menge von Kandidaten wählen kann.

5.3.2 Information Retrieval

Aus Sicht eines TIMs ist die wichtigste Aufgabe des IR, es einem Benutzer zu ermöglichen, *in sehr großen Textmengen sehr schnell zu suchen* (auch als **Volltextsuche** bekannt), um so rasch eine relevante Textmenge zu bestimmen. Man kann sich die Informationsverarbeitung eines TIMs auch als ein **interaktives Textzooming** vorstellen, wo ausgehend von einer initialen Suchanfrage Schritt für Schritt immer „tiefer“ in die Inhalte der Texte hineingezoomt wird, bis die „richtigen“ Antworten gefunden sind. Diese Interaktion kann direkt zwischen Mensch und Maschine stattfinden, indem die Maschine in jedem Schritt eine immer feinere Auswahl von Texten bestimmt, auf deren Basis der Mensch eventuell weitere Anfragen stellt. Man kann sich diese Interaktion aber auch direkt zwischen den Komponenten eines TIMs vorstellen. Z. B. operieren die meisten QA-Systeme auf Paragraphen von Texten, anstatt auf ganzen Texten. Daher wird meist die natürlichsprachliche Frage zuerst in eine Anfrage eines IR-Systems überführt, das damit eine relevante (meist sehr kleine) Menge von Dokumenten bestimmt. Das QA-System kann dann gezielter die relevanten Textpassagen bestimmen, in denen es die Antworten vermutet. Falls das IR-System keine Texte findet, kann das QA-System die Anfrage reformulieren und damit das IR-System erneut ansteuern. Ähnliche Interaktionsmuster lassen sich auch zwischen IE und IR modellieren, wie aktuelle Arbeiten im Bereich der domänenoffenen IE zeigen (vgl. Abschnitt 5.3.3).

Volltextsuche

Die zentrale Datenstruktur bei der Volltextsuche ist der **invertierte Index**, der eine Zuordnung zwischen Schlüsselwörtern und Dokumenten bereitstellt. Der invertierte Index fungiert als fundamentale Schnittstelle zwischen den Benutzern und den Texten, indem Anfragen als Schlüsselwörter interpretiert, auf den Index abgebildet und gefundene Dokumente geordnet nach ihrer Relevanz als Ergebnis geliefert werden. **Indexierung** nennt man den Prozess, der ein Vokabular aus Schlüsselwörtern mit allen Dokumenten eines Textkorpus (z. B. das gesamte Web) assoziiert. Meist wird eine **automatische Indexierung** durchgeführt, wobei beim Einlesen des Korpus aus jedem Dokument alle relevanten Wörter extrahiert werden. In der Regel sind dies die Inhaltswörter bzw. alle Wörter, die nicht in einer sogenannten **Stoppwortliste** vorkommen. Stoppwörter sind demnach solche Wörter, die bei einer Volltextsuche nicht beachtet werden sollen, da sie als inhaltlich irrelevant betrachtet werden und zu redundant sind, z. B. alle Wörter der geschlossenen Wortarten, wie Artikel oder Präposition. Das Sammeln (auch als **Crawling** bekannt) und **Einlesen sehr großer Korpora** (die Überführung von Folgen von Bytes in Folgen von Tokens) ist ein sehr komplexer Prozess an sich, da er nicht nur die verschiedenen Dokumentenformate und -kodierungen berücksichtigen muss, sondern auch die Granularität der Dokumenteneinheiten festlegt (z. B. ganze Dokumente, Paragraphen, Sätze, vgl. Manning et al. 2008 für mehr Details). Des Weiteren werden zur Verarbeitung sehr großer Korpora (insbesondere bei webbasierten Suchmaschinen) verteilte Indexierungsverfahren eingesetzt, wie z. B. das MapReduce Verfahren, das eine allgemeine Architektur für das verteilte Rechnen auf großen Clustern von Rechnern bereitstellt. Auf all diese Aspekte können wir leider nicht im Detail eingehen (vgl. aber Belew 2000 und Manning et al. 2008).

Jeder **Eintrag im invertierten Index** ist selbst eine komplexe Datenstruktur, die neben dem Schlüsselwort eine Liste aller Dokumente enthält (genauer aller *docIDs*, von denen jede einen Zeiger auf ein Dokument repräsentiert), aus denen das Wort extrahiert wurde. Jede *docID* kodiert auch jede Position der Vorkommen eines Wortes im Dokument. Jeder Indexeintrag speichert daneben noch zusätzliche Informationen, wie die Häufigkeit des Wortes *t* (oft auch als **Term** bezeichnet)² pro Dokument *d* (Termfrequenz $tf_{t,d}$) und im gesamten Korpus *c* (Korpusfrequenz cf_t), die Anzahl der Dokumente, in denen der Term *t* vorkommt (Dokumentenfrequenz df_t) oder auch eine Liste von Zeigern auf Synonyme des Termes. Mit Hilfe dieser statistischen Information kann die Trefferliste eines Schlüsselwortes (die mit dem Term assoziierten Dokumente) nach der jeweiligen Relevanz des Termes innerhalb der Trefferliste sortiert werden.³

Ein beliebtes aus diesen Zahlen berechnetes Kriterium ist der *tf-idf*-Wert, der das Verhältnis zwischen der **Termfrequenz** (tf_t) von *t* und seiner **inverse Dokumentenfrequenz** (idf_t , das dem skalierten df_t entspricht, genauer

²Wir werden im Weiteren daher beide Begriffe synonym verwenden.

³Zusätzlich kann auch noch andere Information zur Sortierung der Liste herangezogen werden, wie z. B. die Verlinkungsdichte eines Dokuments. Allerdings können wir auf diese Punkte nicht weiter eingehen (aber vgl. Manning et al. 2008).

$idf_t = \log \frac{|c|}{df_t}$) repräsentiert. Hierbei geht man davon aus, dass ein Wort umso charakteristischer für einen Text d ist, je weniger es in verschiedenen Texten vorkommt und je häufiger es in d vorkommt als in anderen. Anders ausgedrückt, unterschiedliche Wörter in einem Dokument d mit unterschiedlichem $tf\text{-}idf$ -Werten müssen unterschiedlich signifikant für d sein.

Man kann sich leicht vorstellen, dass ein invertierter Index ebenfalls sehr groß werden kann, insbesondere, wenn alle Wortformen als unabhängige Einträge betrachtet werden. Daher findet bei der Indexerstellung oftmals auch eine **Normalisierung** der Wörter auf kanonische Formen statt, z. B. durch eine **Stammformbildung**. Weit verbreitet ist der Einsatz von so genannten **Stemming**-Verfahren, die meist eine lexikonfreie Suffixanalyse vornehmen und einfach an viele Sprachen anzupassen sind. Man beachte, dass Stemming quasi eine Komprimierung mit Verlust darstellt, da ja durch das Stemming Wörter mit unterschiedlichen Suffixen auf den selben Teilstring reduziert werden können, z. B. ‚heizung‘, ‚heizen‘, ‚heizer‘, ‚heizten‘ auf ‚heiz‘. Da auch die Schlüsselwörter aus der Anfrage entsprechend normalisiert werden müssen, verursacht Stemming im Allgemeinen eine erhöhte Ungenauigkeit in der Volltextsuche. Dies ist auch bekannt als **Overstemming** (semantisch unterschiedliche Wortformen werden auf denselben Stamm abgebildet, z. B. ‚Wand‘ und ‚Wandere‘ auf ‚Wand‘) und **Understemming** (unterschiedliche Wortformen eines Wortes werden auf unterschiedliche Stämme abgebildet, z. B. ‚absorbieren‘ und ‚Absorption‘ auf ‚absorb‘ und ‚absorp‘). Um diese Probleme zu vermeiden, werden auch vollständige morphologische Analysekomponenten eingesetzt, was allerdings die Sprachabhängigkeit erheblich erhöhen kann (vgl. Manning et al. 2008).

Der invertierte Index ist der zentrale Zugang zu den mit ihm assoziierten Dokumenten. Das bedeutet, dass **Suchanfragen** als Ausdrücke über Schlüsselwörtern interpretiert werden. Das hat u. a. den enormen Vorteil, dass die Formulierung von Suchanfragen sprichwörtlich kinderleicht ist. Im einfachsten Fall besteht eine Suchanfrage gerade aus einem Schlüsselwort, sehr oft nur aus einer Menge von zwei bis drei Schlüsselwörtern (das haben Analysen von Suchmaschinen-Logs gezeigt). Auf der anderen Seite bedeutet dies aber auch, dass ein Benutzer scheinbar eine sehr hohe Treffergenauigkeit von seinen gewählten Schlüsselwörtern erwartet, d. h. die seiner Meinung nach relevanten Dokumente sollen auch ganz oben in der Trefferliste erscheinen. Für die Entwickler von IR-Systemen hat dies zur Folge, dass die Vorteile einer einfachen Mensch-Maschine-Schnittstelle teilweise dadurch erkauft werden, dass die eigentlichen Suchprozesse entsprechend komplexer zu gestalten sind (die Entwicklung von modernen QA-Systemen ist auch darin motiviert, hier eine alternative Modellierung anzubieten, vgl. Abschnitt 5.3.4). Die Kunst besteht im Prinzip darin, einer kleinen Suchanfrage eine maximale Precision und möglichst hohen Recall zu entlocken.

Es ist hilfreich, für die weiteren Erläuterungen des Suchprozesses ein **Vektorraummodell** für den invertierten Index zu adaptieren. Im Vektorraummodell werden alle n Dokumente und alle m Schlüsselwörter in einer $n \times m$ Matrix angeordnet, wobei jedes Feld dt_{ij} entweder den Wert 0 hat (der Term t_j tritt nicht im Dokument d_i auf) oder einen Zahlenwert enthält, der der Relevanz des Terms im Dokument entspricht, z. B. gemessen durch dessen $tf\text{-}idf$ -Wert. Die entsprechende Index-Matrix sieht wie folgt aus:

$$\begin{pmatrix} dt_{11} & dt_{12} & dt_{13} & \cdots & dt_{1m} \\ dt_{21} & dt_{22} & dt_{23} & \cdots & dt_{2m} \\ dt_{31} & dt_{32} & dt_{33} & \cdots & dt_{3m} \\ \vdots & & & & \\ dt_{n1} & dt_{n2} & dt_{n3} & \cdots & dt_{nm} \end{pmatrix}$$

In dieser Darstellung wird also jedes Dokument als Vektor der Länge m kodiert, wobei jede Zelle kodiert, ob der Term (gewichtet) vorkommt oder nicht. Wenn man jeden Term als Merkmal betrachtet, dann entspricht jedes Dokument einem **Merkmalsvektor**. Ein Termvektor repräsentiert das (gewichtete) Vorkommen eines Termes in allen Dokumenten. Eine Anfrage (die ja auch aus Termen besteht) kann man ebenfalls in Form eines Dokumentenvektors kodieren, den wir als Q bezeichnen wollen und der obigen Matrix als $n + 1$ Zeile hinzufügen werden kann.

Mit dieser Darstellung kann die Frage, welche Dokumente am besten zu einer Anfrage passen, unmittelbar beantwortet werden: Es sind die Dokumente, die der Anfrage relativ zu einer gegebenen Metrik (der Gewichtung) am ähnlichsten sind. Das Vektorraummodell bietet daher den Vorteil, dass man eine uniforme Repräsentationsebene für Dokumente und Anfragen besitzt, mit der verschiedene **Ähnlichkeitsfunktionen** untersucht und verglichen werden können. Grundlage für viele Ähnlichkeitsfunktionen bildet der **Cosinus**. Hierbei wird die Matrix als m -dimensionaler Vektorraum interpretiert, wobei jedes Dokument einem Punkt entspricht. Da eine Anfrage Q auch als Dokument aufgefasst wird, entspricht Q ebenfalls einem Punkt. Nun ist es möglich, den Abstand zwischen Q und allen Dokumenten als Kriterium für Ähnlichkeit heranzuziehen: Je kleiner der Abstand ist, desto ähnlicher sind sich Anfrage und Dokument. Der Cosinus ist nun genau das Mittel, diese Abstände zu berechnen (vgl. auch Belew 2000).

Man beachte, dass eine Volltextsuche eine im Wesentlichen wortzentrierte Operation ist, auch wenn eine Anfrage aus mehreren Termen besteht, da die einzelnen Terme als unabhängig voneinander betrachtet werden. Das hat zwar den Vorteil, dass die Suche sehr flexibel und robust ist, hat aber auch den Nachteil, dass für kleine Anfragen die Suche im Prinzip sehr underspezifiziert und unscharf ist. Manning et al. (2008) beschreiben eine Reihe von Erweiterungen, wie der Suchprozess kontrollierter realisiert werden kann, z. B. **Query Expansion** (beispielsweise durch das Hinzufügen von synonymen Wörtern) oder **Relevance Feedback** (der Benutzer bewertet explizit die Suchergebnisse, auf deren Basis das IR-System seine Suchstrategien zu optimieren versucht). QA-Systeme, speziell die Frageanalysekomponenten, versuchen ebenfalls gezielt eine Volltextsuche zu integrieren, da ausgehend von der syntaktischen und semantischen Analyse einer W-Frage („Wer war in den 80iger Jahren Deutschlands Bundeskanzler?“) gezielt ein IR-System angesteuert werden muss, um die relevanten Dokumente und Textpassagen für die Antwortextraktion zu bestimmen (vgl. Abschnitt 5.3.4).

Verzeichnisbasierte Suche

Eine weitere Möglichkeit ist es, die Dokumentenmenge in Teilmengen zu zerlegen, um so eine Art begriffliche Strukturierung der Dokumente in thematisch ähnliche Verzeichnisse zu erhalten. Z. B. könnte man die Verzeichnisse „Umsatz von Firmen“, „Computer“, „Unterhaltung“ und „Sonstiges“ definieren und die Dokumente eines Korpus entsprechend zuordnen. Im Prinzip könnte man auch pro Verzeichnis weitere Unterverzeichnisse festlegen, etwa „Umsatz von Firmen in der Computerbranche“ oder „Umsatz von Firmen im Bankenbereich“. Dann kann man einen zweistufigen Suchprozess anbieten, wo in der ersten Stufe der Benutzer zuerst das Verzeichnis auswählt (dies entspricht im Prinzip einem Navigieren in einer Hierarchie). Je nach Thema bekommt man dann entweder alle dem gerade ausgewählten Verzeichnis zugeordneten Unterverzeichnisse oder eine Liste von aktuellen Dokumenten angeboten, in denen man dann wie gewohnt durch Angabe von Schlüsselwörtern suchen kann, wobei natürlich nur in der relevanten Teilmenge der Dokumente gesucht wird. Beispiele für solche verzeichnisbasierten Suchmaschinen sind das Google Verzeichnis (<http://www.google.de/dirhp?hl=de>) oder das Open Directory Project (<http://www.dmoz.de/>).

Theoretisch ist es möglich, die Verzeichnisstruktur vorzugeben (z. B. durch einen Thesaurus oder eine Ontologie, vgl. Unterkapitel 4.6) und Dokumente manuell zuzuordnen. Allerdings ist dies ein zeitaufwändiger und sehr kostenintensiver Prozess, insbesondere für sehr große Dokumentensammlungen. Daher werden gerade in diesem Bereich meist maschinelle Lernverfahren eingesetzt, die wir hier kurz vorstellen wollen.

Textklassifikation

Im Prinzip kann man den Bezeichner eines Verzeichnisses auch als eine Art „eingefrorene Suchanfrage“ (**standing query**) auffassen, im Gegensatz zu den bisher betrachteten „spontanen Suchanfragen“ (**ad hoc queries**). Bei den ad hoc queries werden die Suchergebnisse quasi vergessen, wenn der Benutzer eine neue Anfrage stellt. Dadurch wird es aber sehr schwierig, Veränderungen in der Dokumentenmenge zu registrieren, wenn die Frage zu einem späteren Zeitpunkt erneut gestellt wird (z. B. für die Anfrage „aktuelle Preise von Blue Ray Recordern“). Die Idee der standing queries ist es nun, diese Anfragen periodisch auszuführen und die gefundene relevante Dokumentenmenge inkrementell zu erweitern, d. h., die mit einer standing query gespeicherten (alten) Dokumentenmenge zu aktualisieren.

Dokumente, die als Ergebnis einer standing query Q bestimmt werden, sind Texte über Q . Daher können standing queries auch als Klassenbezeichner aufgefasst werden, die die Dokumentenmenge in solche zerlegen, die *über* Q sind und die *nicht über* Q sind. Damit kann die Beantwortung von standing queries auch als **Textklassifikation** betrachtet werden: Gegeben sei eine Menge von Klassen. Bestimme für jedes Dokument, zu welcher Klasse es gehört. Anstelle von **Klasse** wird auch oft der Term **Topik** verwendet und Textklassifikation dann auch als **Topik Klassifikation** oder **Topik Spotting** bezeichnet.

Weit verbreitet, weil sehr erfolgreich, sind Statistik-basierte Verfahren zur Textklassifikation. Diese Verfahren lernen automatisch die Entscheidungskriterien aus einer Menge von Trainingsbeispielen, d. s. gute Beispieltexte für jede Klasse. Sie werden daher auch als **überwachte** Lernverfahren bezeichnet. Das Klassifizieren der Beispiele wird zwar in der Regel manuell durchgeführt, ist aber relativ einfach (man muss kein Wirtschaftsexperte sein, um zu entscheiden ob ein Text eine Umsatzmeldung beschreibt oder nicht). Oft ist der Prozess der Annotation auch Teil des Informationsflusses, wenn z. B. neue Dokumente, die mit einer standing query bestimmt wurden, als relevant oder nicht relevant für eine Klasse zu klassifizieren sind. Aufgabe der Lernverfahren ist es, mit den Trainingsdaten eine Wahrscheinlichkeitsverteilung zu berechnen, mit deren Hilfe die beste Klasse für neue Dokumente bestimmt werden kann. Dabei wird für jede mögliche Klasse der sich aus der Verteilung ergebende Wert berechnet und die Klasse mit höchstem Wert gewählt. Die Verteilungsfunktionen operieren über Merkmalen, wie z. B. den Wörtern der Texte, Wortfolgen, Wortarten etc. Daher werden die Texte entsprechend vorbearbeitet und in einem **Merkmalsvektor** auf ähnliche Weise repräsentiert, wie wir dies bereits in Abschnitt 5.3.2 bei der Volltextsuche beschrieben haben.

Zu den besten Textklassifikatoren (F-Maß von mehr als 0.88)⁴ gehören die stützmengenbasierten Verfahren, die so genannten **Support Vector Machines** (SVM; vgl. Schölkopf und Smola 2002). Für gegebene positive und negative Beispiele (da dies Vektoren sind, also Punkte im Vektorraum) ist es das Ziel, eine lineare Funktion zu bestimmen, die die Punkte dieser Klassen optimal von einander trennt, wobei nur die Punkte (sog. **Stützpunkte**) betrachtet werden müssen, die zur Funktion den geringsten Abstand haben. Optimal bedeutet, dass in der direkten Umgebung der Funktion ein möglichst großer Bereich ist, in dem keine Punkte vorkommen. Diese Eigenschaft erlaubt, neue Texte recht zuverlässig zu klassifizieren. SVM-Verfahren gehören zu den Kernel-Verfahren, die wir in Abschnitt 5.3.3 im Kontext der Extraktion von Relationen kurz erläutern werden. Weitere Details zu SVM-basierten Textklassifikatoren und zu anderen Verfahren, wie etwa **Naive Bayes**, **Rocchio** oder **k-Nearest-Neighbor** (**k**NN) finden sich in Manning et al. (2008).

Ein mögliches Anwendungsfeld der Textklassifikation ist die Informationsextraktion (IE), wo auf diesem Wege relevante Texte periodisch bestimmt werden können, aus denen dann die wichtigen Entitäten und Relationen extrahiert werden (wie wir dies ja bereits im Beispiel in Abschnitt 5.3.1 angedeutet haben). Der Vorteil ist hierbei, dass die IE direkt auf die so gefilterten Dokumente angewendet werden kann und sich somit nicht um „falsche“ Texte kümmern muss (vgl. Abschnitt 5.3.3). Weitere Anwendungsfelder sind u. a. das automatische Filtern oder Weiterleiten von Dokumenten, das Bestimmen von Spam-Seiten, von Texten mit pornografischen Inhalten oder mit subjektiven positiven oder negativen Bewertungen von Produkten (auch als Sentiment Detection bekannt) oder zur Sortierung von Emails.

⁴vgl. Li und Yang (2003)

Textclustering

Clustering ist die Verteilung von Daten in Gruppen von ähnlichen Objekten und leistet damit auch ein automatisches Erstellen von Klassen. Jede Gruppe wird als Cluster bezeichnet und besteht aus den Objekten, die einerseits untereinander ähnlich sind und andererseits unähnlich zu den Objekten der anderen Gruppen. Aus Sicht eines TIMs meint Ähnlichkeit hier, dass sich in einer Gruppe möglichst nur die Objekte befinden sollen, die bezogen auf ein Informationsbedürfnis die gleiche Relevanz haben. Für ein IR-System sind dies z. B. die Dokumente, die für eine Suchanfrage relevant sind; für ein IE-System mögen dies Textpassagen sein, die die gleichen Eigennamen und Relationen enthalten und für ein QA-System könnten dies Sätze aus verschiedenen Dokumenten sein, die Formulierungsvarianten derselben Antwort enthalten.

Clustering-Verfahren sind **unüberwachte** Lernverfahren, da sie keine Trainingsbeispiele haben. Auch sind die Gruppenbezeichner nicht explizit bekannt. In diesem Sinne entspricht ein Cluster einem „verstecktem“ (**hidden**) Muster. Im Falle eines TIMs entsprechen die Objekte linguistisch Objekten, wie etwa Texten, Sätzen, Wörtern oder Wortpaaren, oder auch Relationen. Die Mehrzahl der Clustering-Verfahren modelliert Objekte ebenfalls als Punkte-zu-Merkmale Datenformate, die konzeptuell ebenfalls einer $n \times m$ -Matrix entsprechen (dem Merkmalsraum für n Objekte und m verschiedene Merkmale). Das Ziel des Clusterings ist es dann, die Punkte einer Menge X einem endlichen System von k Teilmengen, den Clustern C_i , zuzuweisen, sodass die einzelnen Cluster sich möglichst nicht schneiden, d. h. $X = C_1 \cup C_2 \dots C_k \cup C_{outliers}$, wobei $C_i \cap C_j = \emptyset$. Die Ähnlichkeit von Objekten kann dann z. B. mittels eines Distanzmaßes zwischen entsprechenden Punkten bestimmt werden, sodass der „Abstand“ von Texten innerhalb eines Clusters möglichst minimal und die Distanz zwischen verschiedenen Clustern möglichst maximal ist.

Methoden zum Clustering können grob aufgeteilt werden in hierarchische und flache Verfahren. Beim **hierarchischen Clustering** ergibt der Clusteringprozess eine Baumstruktur. Jeder Clusterknoten enthält Kindercluster, wogegen Geschwistercluster die Punkte des Elternclusters partitionieren (vgl. Berkhin 2006). Methoden zum hierarchischen Clustering können weiter unterteilt werden in bottom-up (**agglomerative**) und top-down (**divisive**). **Agglomeratives Clustering** beginnt mit Clustern, die jeweils nur einen Punkt (Text) enthalten, und vereinigen rekursiv die jeweils zwei aktuell geeignetsten Cluster zu neuen Clustern. **Divisives Clustering** startet mit einem Cluster, der alle Punkte enthält und spaltet rekursiv das aktuell beste Cluster ab, solange das Terminierungskriterium nicht erreicht ist (oft ist dies eine vorgegebenen Anzahl k von Clustern). Die wichtigsten Vorteile des hierarchischen Clusterings sind 1) die Granularität der Clusterhierarchie kann gesteuert werden, 2) verschiedenste Distanzmaße können integriert werden und 3) sind damit beliebige Merkmalstypen wählbar. Ein gravierender Nachteil ist das vage Terminierungskriterium, da die Vorgabe von k schon einen erheblichen Eingriff in das Clustering-Verfahren bedeutet und oftmals nur durch Schätzung oder mit Hilfe von Domänenwissen bestimmt werden kann.

Flaches Clustering erzeugt eine flache Clusteringstruktur, d. h. es gibt keine explizite strukturelle Beziehung zwischen den Clustern. Sehr populär in wissenschaftlichen und industriellen Anwendungen ist der **K-means** Algorithmus. Jedes der k Cluster C_i wird durch den Durchschnittswert c_i seiner Punkte definiert. c_i heißt dann auch **Zentroid**. Ziel ist es, die durchschnittliche Distanz zwischen Texten und ihren Zentroiden zu minimieren bzw. die Ähnlichkeit zwischen Texten und ihren Zentroiden zu maximieren. Zuerst werden zufällig k Punkte als initiale Zentroide bestimmt und von diesen ausgehend die anderen Punkte um die Zentroide angeordnet. Für die sich so ergebenen Cluster werden dann die Zentroiden neu bestimmt. Auf dieser Basis wird der Prozess rekursiv solange über den k Clustern fortgeführt, bis keine Veränderungen bzw. Bewegungen der Zentroiden mehr stattfinden. K-means Verfahren sind einfach und schnell, haben aber den Nachteil, dass sie nur numerische Merkmale für die Distanzberechnung verwenden können. Außerdem reagieren sie sehr fragil auf die initiale Auswahl von Zentroiden.

Wie erwähnt wird Clustering bereits aktuell zur dynamischen Erzeugung von Verzeichnissen in Suchmaschinen eingesetzt, vgl. z.B. <http://clusty.com/>. Manning, Raghavan und Schütze (2008) beschreiben weitere Einsatzgebiete des Clusterings im IR. In Forschungsprototypen von QA-Systemen wird Clustering oft zur Erkennung von Paraphrasen eingesetzt und zur Gruppierung von semantisch ähnlichen Textpassagen, womit eine gezielte Antwortextraktion möglich wird (vgl. Abschnitt 5.3.4). Aktuell werden Clustering Verfahren verstärkt im Bereich der IE exploriert, insbesondere im Bereich der Behandlung mehrdeutiger Eigennamen (vgl. Abschnitt 5.3.3) und der webbasierten, domänenoffenen IE (vgl. Abschnitt 5.3.3).

5.3.3 Informationsextraktion

Das Ziel der Informationsextraktion (IE) ist es, domänenspezifische Informationen aus freien Texten gezielt aufzuspüren und zu strukturieren, bei gleichzeitigem „Überlesen“ irrelevanter Information. Die IE-Technologien versuchen keine umfassende Analyse des gesamten Inhaltes aller Textdokumente, sondern sollen nur die Textpassagen analysieren bzw. „verstehen“, die relevante Information beinhalten. Was als relevant gilt, wird dabei durch vordefinierte domänenspezifische Lexikoneinträge oder Regeln dem System fest vorgegeben. Dieses Wissen muss dabei so detailliert und genau wie möglich festlegen, welche Typen von Information von einem IE-System extrahiert werden soll, damit eine umfangreiche und zugleich präzise Extraktion ermöglicht wird. Dieser letzte Aspekt betrifft zumindest die **traditionelle IE**, die wir in diesem Abschnitt genauer beleuchten werden. In aktuellen modernen IE-Ansätzen – insbesondere **domänenoffene IE** – wird versucht, alle möglichen relevanten Informationen in einer Textkollektion zu erkennen und zu klassifizieren. Diese Verfahren werden wir in Abschnitt 5.3.3 genauer erläutern.

Beiden IE-Ansätzen gemeinsam ist das Ziel, in freien Texten das textuelle Vorkommen (die Erwähnung) von Entitäten und ihre Beziehungen zueinander zu lokalisieren und in ein strukturiertes Format zu überführen. Damit streben

sie eine tiefere Verstehensleistung an, als die volltextbasierte IR. IE kann so betrachtet auch als eine Art „schärferes Textzooming“ betrachtet werden und damit als ein dem IR nachfolgender Analyseprozess. Die wichtigsten konkreten Lösungsschritte umfassen 1) die Erkennung von spezifischen Eigennamen (z. B. Namen von Personen, Firmen, Produkten, Lokationen), 2) die Referenzierung der erkannten Eigennamen (z. B. bestimmen, dass „Angela Merkel“, „A. Merkel“ und „die Bundeskanzlerin“ dieselbe Person referieren) und 3) die Erkennung und Klassifikation von Relationen unterschiedlicher Komplexitäten zwischen den identifizierten Eigennamen. Mit anderen Worten ist die zentrale Aufgabe der Informationsextraktion die Extraktion von **semantischen Relationen**. Beispielsweise könnte aus folgendem Text

Die Hauptgeschäftsstelle von Apple Inc. befindet sich in der Mitte von Silicon Valley, in 1 Infinite Loop, Cupertino, Kalifornien.

die Relation HATHAUPTGESCHÄFTSSTELLE(APPLE INC., 1 INFINITE LOOP-CUPERTINO-CALIFORNIA) extrahiert werden. Dabei ist die zentrale Herausforderung, diejenige Relation aus Sätzen zu extrahieren, die die selbe Information ausdrücken, unabhängig von der konkreten sprachlichen Formulierung. Des Weiteren sollten auch entsprechende Informationen über andere Firmen extrahiert werden können, wenn diese in ähnlich formulierten Texten vorkommen, also dieselbe Relation verbalisiert ist, aber mit anderen Argumenten, z. B. HATHAUPTGESCHÄFTSSTELLE(IBM CORP., 1 NEW ORCHARD ROAD-ARMONK-NEW YORK).

Die frühe Generation von IE-Systemen bestand im Wesentlichen aus manuell implementierten Systemen, in denen der notwendige domänenspezifische Regelapparat für alle Teilbereiche per Hand programmiert wurde (vgl. u. a. Israel et al. 1996 oder Neumann et al. 1997 und Carstensen 2009b für einen genaueren geschichtlichen Einblick in die Entwicklung von IE-Systemen). Unser Beispiel von oben aufgreifend bedeutet dies, dass für eine bekannte semantische Relation HATHAUPTGESCHÄFTSSTELLE eine Menge von Regeln oder Muster der Art (hier sehr vereinfacht):

* HAUPTGESCHÄFTSSTELLE VON X BEFINDET SICH * IN Y

manuell zu erstellen ist, wobei X und Y getypte Variablen der Argumente (in der Regel Eigennamen) repräsentieren. Dieses Muster passt auf alle Textstellen, wo auf eine beliebig lange Folge von Termen (was durch den Kleene-Stern „*“ kodiert ist) die Wörter „Hauptgeschäftsstelle von“ genau in dieser Form und Anordnung folgen, denen sich unmittelbar ein Eigename vom Typ X anschließt, dem wiederum die Sequenz „befindet sich“ folgt und dieser schließlich ein Text folgt, der auf analoge Weise mit dem Teilausdruck * IN Y passend ist.

In manuellen IE-Systemen wird die Spezifikation solcher Muster auf der Basis einer manuellen Analyse relevanter Textkorpora geleistet, wobei diese jedoch oft durch eine automatische linguistische Merkmalsextraktion aktiv unterstützt wird. Die manuelle Durchsicht eines Korpus hat zwar den Vorteil, dass zum Einen sehr genaue Muster bestimmt werden können und zum Anderen auch die relevanten linguistischen Merkmale eindeutig identifiziert werden können. Ein gravierender Nachteil ist, dass dieser Prozess sehr zeit- und kostenintensiv ist

und dadurch oft nur eine geringe Abdeckung mit manuellen Systemen erreicht wird, insbesondere dann, wenn eine IE-Anwendung zeitnah bereitzustellen ist.

Daher basieren die aktuell am weitesten fortgeschrittenen Technologien algorithmisch auf Verfahren des Maschinellen Lernens, die unterschiedliche Granularitäten von linguistischer Merkmalsextraktion in Betracht ziehen, z. B. von POS-Tagging bis zu vollem Parsing. Ziel dieser Lernverfahren ist es, durch automatische Bestimmung optimaler Kombinationen von Merkmalen die Extraktionsmuster zu induzieren, die später in der Anwendungsphase zur automatischen Erkennung und Klassifikation von Entitäten und Relationen herangezogen werden. Damit kann dann eine sehr viel größere Abdeckung erreicht werden, wobei die aktuell besten Strategien bereits eine vielversprechende Genauigkeit demonstrieren. Die zugrunde liegenden Lernstrategien der Mehrzahl der neuen Ansätze reichen von überwachten über semi-überwachten bis zu unüberwachten Lernmethoden mit einer aktuell starken Tendenz zu semi-überwachten und unüberwachten Methoden. Wir werden nun in den nächsten zwei Abschnitten zuerst etwas genauer auf die Extraktion von Eigennamen und Relationen eingehen, bevor in Abschnitt 5.3.3 Strategien zu deren Dereferenzierung und Integration vorgestellt werden.

Extraktion von Eigennamen

Die Erkennung und Klassifikation von Eigennamen stellt einen zentralen Baustein für ein TIM dar. Unter Eigennamen versteht man in der Regel sprachliche Ausdrücke, die auf Individuen von Klassen oder Typen bestimmter Entitäten referieren, wie z. B. wie Personen-, Firmen-, Produktnamen, komplexe Datums-, Zeit-, und Maßausdrücke.

Die einfachste Art der Eigennamenerkennung (**Named Entity (NE) Recognition, NER**) wäre sicherlich die vollständige Aufzählung aller Eigennamen eines bestimmten Types in einer Liste (die dann auch als **Gazetteer** bezeichnet wird), z. B. eine Liste aller Personennamen. Dies ist sicherlich ein gangbarer erster Implementationsweg (und ist auch weitverbreitet), aber sicherlich kein ausreichender. Zum Einen gibt es für einen Eigennamen in der Regel viele Formulierungsmöglichkeiten (z. B. „Angela Merkel“, „A. Merkel“, „Merkel, Angela“, „Merkel, A.“, „Angie“, „Merkel, Dr. Angela“, „A. M.“, etc.), zum Anderen ist der Prozess der Generierung von Eigennamen sehr produktiv, z. B. bei Firmennamen oder Produktnamen. Hinzu kommt, dass durch die immer noch stattfindende Expansion von IE-Anwendungen und das Aufkommen von neuen verwandten Themen (wie etwa domänenoffene Fragebeantwortung, semantische Suche), auch eine drastische Erweiterung der NE-Kategorien stattgefunden hat. So stellt z. B. Sekine et al. (2002) ein System bestehend aus rund 150 verschiedenen Kategorien vor.

Auf Grund dieser Eigenschaften und Entwicklungen dominiert heute die Entwicklung von Daten-gesteuerten Verfahren zur NER mit einem Fokus auf überwachte und semi-überwachte Strategien. Die Idee **überwachter Lernverfahren** ist die automatische Berechnung von optimalen Merkmalskombinationen aus positiven und negativen Beispielen von NEs, die in einem großen Textkorpus

in der Regel manuell annotiert vorliegen. Diese Merkmalskombinationen dienen dann als Vorhersagemodelle zur **Erkennung und Klassifikation** von neuen Vorkommen von NEs. Die Merkmale beziehen sich dabei zum Einen auf die Schreibweise von Eigennamen (z. B. Großschreibung, Vorkommen von invarianten Designatoren, wie etwa AG oder Inc. oder die Länge der Buchstabenfolge) und zum Anderen auf Information im Kontext von NE, z. B. Wortform oder Wortart adjazenter linker und rechter Wörter oder aber auch komplexe syntaktische Merkmale, wie Vorkommen eines NEs in einer bestimmten syntaktischen Konstruktion einer Nominalphrase. Diese Lernverfahren sind oft Anpassungen von bekannten Lernverfahren, wie Hidden-Markov Modellen (HMM; Biket et al. 1999), Maximum Entropy Modellen (MEM; Borthwick et al. 1998), Support Vector Machines (SVM; Asahara und Matsumoto 2003) oder Conditional Random Fields (CRF; Li und McCallum 2003) oder auch Kombinationen von verschiedenen Verfahren, z. B. MEM und datengesteuertes Parsing (Neumann 2006).

Der zentrale Nachteil überwachter Lernverfahren ist, dass sie sehr große annotierte Korpora zum Trainieren benötigen, insbesondere dann, wenn sehr viele NE-Kategorien vorliegen. Daher werden auch alternativ sogenannte **semi-überwachte Verfahren** erforscht und sehr aktuell unüberwachte Lernverfahren (auf die wir in Abschnitt 5.3.3 genauer eingehen werden). Die zentrale Technologie von semi-überwachten Verfahren ist als **Bootstrapping** bekannt. Diese Verfahren verfügen über einen sehr kleinen Grad an Überwachung in Form einer kleinen Menge von bereits bekannten Eigennamen (auch als **Seeds** bezeichnet) pro zu erlernender NE-Kategorie, die zur Initialisierung des Lernvorgangs herangezogen werden. Wenn es beispielsweise das Ziel ist, Namen von Krankheiten in einem Textkorpus zu bestimmen, dann könnte der NE-Lerner mit einer Liste von fünf bekannten Namen von Krankheiten gestartet werden. Der NE-Lerner sucht dann das gesamte Textkorpus nach Vorkommen dieser Seeds ab und annotiert die entsprechenden Textstellen mit dem Namenstyp. In einem nächsten Schritt werden dann automatisch die besten oder sichersten Merkmalskombinationen bestimmt, die sich im Kontext (und der Schreibweise) der verschiedenen Vorkommen der Seeds bestimmen lassen. Dies ergibt eine erste Menge von Mustern. In einem nächsten Schritt werden diese Muster herangezogen, um nun ihrerseits neue Eigennamen zu bestimmen, die der Seedliste (nachdem die einzelnen Einträge vom NE-Lerner selbstständig bewertet wurden) hinzugefügt werden. Dieser Prozess wird dann solange wiederholt, bis keine neuen Einträge mehr gefunden werden können oder der Prozess von außen terminiert wird (siehe auch Nadeau und Sekine 2007). Beispiele für solche Lernverfahren finden sich u. a. in Collins und Singer (1999) und Yangarber et al. (2002), wobei sich die Verfahren auch in der Tiefe der linguistischen Merkmalsextraktion unterscheiden. So verlangt z. B. das Verfahren von Collins, dass das gesamte Korpus vollständig geparsst wird, bevor der Lernprozess gestartet werden kann, wohingegen Yangarberrs Methodik nur ein POS-Tagging benötigt.

Zentrale Nachteile der aktuellen Verfahren sind zum Einen, dass sie ein sehr viel größeres Korpus benötigen als überwachte Verfahren, und zum Anderen, dass die konkrete Auswahl der Seedelemente einen großen Einfluss auf die Performanz des NE-Lerners haben kann, z. B. wenn Namen gewählt werden, die z. B. nur sehr

spärlich und in sehr seltenen Kontexten vorkommen, was man aber im Vorfeld der Anwendung eines semi-überwachten Lerners in der Regel nicht wissen kann (eine manuelle Sichtung des Korpus soll ja vermieden werden).

Mehrdeutige Eigennamen

Die Extraktion von Eigennamen im engen Sinne erkennt und klassifiziert einzelne Ausdrücke, leistet aber noch keine Auflösung von mehrdeutigen Eigennamen. Im allgemeinen handelt es sich hierbei um eine viele-zu-viele Relation, die durch zwei lexikalisch-semantische Relationen verursacht wird: 1) Polysemie: Derselbe Name verweist auf mehrere Entitäten und 2) Synonymie: Dieselbe Entität kann mehrere Namen haben. Ziel der **Disambiguierung von Eigennamen** ist es, diese Relation für die extrahierten Namen explizit zu berechnen.

Eine aktuell sehr im Fokus stehende Anwendung ist die **Personensuche im Web**, wofür es auch bereits erste spezialisierte Suchmaschinen gibt, z. B. <http://www.spock.com/> oder <http://www.zoominfo.com/>. Meistens werden die zugrundeliegenden persönlichen Einträge und Profile zunächst von der jeweiligen Person selbst zur Verfügung gestellt. Es gibt aber bereits aktuelle Forschungsansätze, diese Information voll automatisch durch die Analyse von Webseiten zu bestimmen. Dabei werden als Informationsquelle Webseiten inspiziert und klassifiziert (vgl. Artiles et al. 2007). Der Informationsfluss ist in allen dort vorgestellten Systemen ähnlich. Zuerst wird der Textteil aus den HTML-Seiten extrahiert (basierend auf maschinellen Lernverfahren oder Regeln). Danach werden die Texte mit einer Kombination aus Web und LT-Tools weiter analysiert, u.a. mittels Stemmer, POS-Tagger, Chunker, NER, Koreferenzierer⁵, sowie durch Extraktion von Fakten, Email und URL und Analyse der Links. Alle diese extrahierten Information werden dann als Merkmale für ein nachfolgendes Clustering der Webseiten eingesetzt, sodass ein Cluster ein Individuum repräsentiert. Dabei hat sich gezeigt, dass eine fehlerfreie Erkennung von Eigennamen und die Extraktion von Relationen auf der Basis von vollem Parsing von steigender Wichtigkeit ist, um feinkörnige Informationseinheiten lokalisieren und klassifizieren zu können, vgl. <http://nlp.uned.es/weps/>.

Verfahren zur Disambiguierung von Eigennamen verknüpfen Namen mit Entitäten, die in **externen Wissensquellen** vordefiniert sind, z. B. in Form von externen strukturierten Lexika, in denen die verschiedenen Entitäten mittels getypten Attributen unterschieden werden. Aktuell im Fokus der Forschung steht die Entwicklung von Verfahren, die diese externen Wissensquellen automatisch erstellen können. Von besonderem Interesse ist hierbei die Verwendung von Wikipedia (vgl. auch 4.7), da **Wikipedia** tatsächlich als eine der größten verfügbaren Ressourcen dieser Art betrachtet werden kann. Man kann Wikipedia-Artikel über Entitäten (also Artikel, deren Titel im Wesentlichen aus einem Eigennamen besteht, z. B. „Jim Clark“) als deren Definition betrachten. Dann ist es das Ziel, für Eigennamen, die aus beliebigen anderen Textquellen extrahiert wurden, die

⁵Das ist eine Komponente, die bestimmt, welche sprachlichen Ausdrücke (z. B. Eigennamen und Nominalphrasen) auf dieselben Entitäten verweisen und diese entsprechend verknüpft, vgl. auch Unterkapitel 3.7.

entsprechenden korrekten Wikipedia-Artikel zu bestimmen. Dies ist keine triviale Aufgabe, da die Eigennamen in der Regel normalisiert werden müssen (z. B. „J. Clark“, „James Clark“ auf „Jim Clark“) und die Ähnlichkeit des Quelltextes mit dem Wikipedia-Artikel bestimmt werden muss (vgl. u. a. Bunescu und Pasca 2006 und Cucerzan 2007).

Extraktion von Relationen

Die zentrale Aufgabe im Bereich der **Relationsextraktion (RE)** ist das *Erkennen und Klassifizieren von Tupeln* von Entitäten, für die es textuelle Belegstellen gibt, dass diese Entitäten in einer bestimmten, in der Regel vordefinierten Relation zueinander stehen. Die textuellen Belegstellen entsprechen dabei Textausschnitten, die die Entitäten und die Relation direkt formulieren. Sucht man beispielsweise nach Tupeln (oder Instanzen) einer vordefinierten Relation vom Typ ÜBERNAHMEFIRMA(FIRMA1, FIRMA2), dann wäre der folgende Text eine positive Belegstelle für das Tupel **(Google, YouTube)**:

10. Okt. 2006 ... Die Gerüchteküche irrte sich nicht: **Google** hat die Videoplattform **YouTube** übernommen. Der Kaufpreis liegt mit 1,65 Milliarden US-Dollar in ...

wogegen der folgende Text kein positiver Beleg für diese Relation ist:

19. Nov. 2008 ... Das fragt Alex Iskold auf ReadWriteWeb. Er erzählt die Geschichte eines neunjährigen Jungen, welcher statt **Google** immer **YouTube** verwendet, ...

Diesem ersten Beispiel können wir bereits wichtige, häufig gemachte Annahmen entnehmen, die im Prinzip den meisten RE-Verfahren zu Grunde liegen. Erstens: Die Argumente der Zielrelation sind normalerweise Entitäten mit vordefiniertem Typ. In unserem Beispiel sind etwa nur Argumente vom Typ FIRMENNAME zugelassen. In den meisten Fällen sind die Argumente tatsächlich auf Eigennamen beschränkt (etwa PERSONEN, ORGANISATIONEN, ORTE, ZEITPUNKTE), da Relationen zwischen Eigennamen oft einen sehr hohen Informationswert (und Unterhaltungswert) besitzen. Zweitens: Die Relationsextraktion wird meist beschränkt auf das Auffinden einer vordefinierten Relation zwischen Entitäten, die im selben Satz auftreten. Drittens: Die RE-Aufgabe kann dann auch als Klassifikaitionsaufgabe für Sätze aufgefasst werden zu entscheiden, ob Sätze, die die gleichen (Typen von) Eigennamen nennen, auch die gleiche Relation benennen oder nicht.

Die ersten Maschinellen Lernverfahren für RE waren Varianten von **überwachten** induktiven Lernverfahren. Ausgehend von einer manuell erstellten Trainingsmenge von positiven und negativen Sätzen (in denen die Eigennamen natürlich auch markiert sind), ist es das Ziel, automatisch Regeln zu induzieren. Dazu werden schrittweise die durch die Trainingsmenge vorgegebenen Sätze verallgemeinert, sodass sie auch auf nichtannotierte, neue Dokumente möglichst fehlerfrei anwendbar sind. Auf diese Weise könnte z. B. aus einer Menge von

positiven Trainingssätzen, die Interaktionen zwischen Proteinen ausdrücken, folgende Regel abgeleitet werden (vgl. auch Bunescu et al. (2005)):

(7) interaction (0) [between | of] (5) PROT (9) PRO (17) .

wobei die geklammerten Zahlen (z. B. (7)) ausdrücken, wie viele beliebige Tokens oder Wörter maximal zwischen den explizit genannten Wörtern (z. B. interaction) oder den Eigennamen (z. B. Eigennamen vom Typ PROT, also Protein) vorkommen dürfen. Wörter in eckigen Klammern und durch | getrennt repräsentieren alternative Formulierungsmöglichkeiten an diesen Stellen in Sätzen.

Um zu diesen **domänenspezifischen Regeln** zu gelangen, werden die Sätze mit verschiedenen linguistischen Werkzeugen vorverarbeitet, von reiner Wortanalyse (z. B. Freitag 1998), über POS-Tagging (z. B. Califff und Mooney 1999) und Phrasenerkennung (Huffman 1996) bis zu vollem Parsing (z. B. Soderland 1997). Die verwendeten Verfahren verlaufen meist bottom-up und vergleichen Sätze und Regeln wortweise, wobei die zugrundeliegende Suchstrategie oft greedy (lokale Optima bevorzugend) ist. Die resultierenden Regeln können als kompaktifizierte Sequenzen von Wörtern (plus linguistischen Merkmalen) betrachtet werden, wobei die Wörter als explizite **lexikalische Anker** dienen. Das heißt, dass in neuen Sätzen genau an den gleichen Stellen die gleichen Wörter auftreten müssen, damit die Muster überhaupt angewendet werden können. Das bedeutet dann auch, dass Sätze, in denen keine solchen lexikalischen Anker vorkommen, direkt ausgespielt (quasi „überlesen“) werden können. Damit die erlernten Regeln einen möglichst hohen Abdeckungsgrad haben, ist die Anzahl solcher lexikalischen Anker in einer Regel meist klein. Das führt in der Anwendungsphase häufig dazu, dass die erlernten Regeln zwar eine hohe Präzision besitzen, aber nur eine geringe Vollständigkeit (sie „überlesen“ zu viel).

Neuere Lernverfahren zur RE versuchen dieses Problem dadurch zu beheben, dass sie nicht nur einen wortweisen Vergleich vornehmen, sondern dass sie alle möglichen Teilfolgen von Wörtern in Sätzen betrachten und es dem Trainingsalgorithmus überlassen, die besten Teilfolgen zu bestimmen. Prinzipiell lassen sich diese Teilfolgen automatisch vorberechnen und könnten somit als zusätzliche Merkmale benutzt werden. Allerdings führt solch eine explizite Erzeugung von Merkmalen zu einem enormen Anwachsen des expliziten Suchraums und damit zu nicht mehr machbaren Trainingszeiten. Aktuell erforschte RE-Lernverfahren versuchen daher, diese Berechnungen quasi implizit durch eine Funktion (genannt **Kernel**) zu bestimmen, die die Ähnlichkeit von Objekten (z. B. Sätzen) bestimmt. Je größer der Wert des Kernels, desto ähnlicher sind die verglichenen Objektpaare. Dabei liegt der Fokus nicht auf der expliziten Repräsentation von Objekten als Merkmalsvektor, sondern auf dem Datentyp der Objekte. Häufig verwendete Datentypen sind Zeichenketten, Sequenzen beliebiger Elemente, Baumstrukturen oder sogar Graphen. Ein Kernel kann dann als rekursive Funktion modelliert werden, in denen sehr effiziente elementare Vergleichsoperatoren effizient zusammengefügt werden. So lässt sich z. B. die Ähnlichkeit von zwei Sequenzen durch die Anzahl der gleichen Teilesequenzen bestimmen oder die Ähnlichkeit von zwei Bäumen durch die Anzahl der gleichen Teilbäume.

Ein zentraler Vorteil von Kernels ist, dass sie sehr elegant mit maschinellen Lernverfahren verknüpft werden können, wie z. B. Support Vector Machines (SVM) oder aber auch Latent Semantic Analysis. Hierbei ist das Ziel, ausgehend von einer Menge von Trainingsbeispielen einen problemspezifischen optimalen Kernel zu bestimmen (vgl. z. B. Zelenko et al. 2003, Culotta und Sorensen 2004, Bunescu und Mooney 2005, Zhao und Grishman 2005, Bunescu 2007, Wang und Neumann 2007 und Moschitti et al. 2008).⁶

Die bisher beschriebenen Verfahren zur Relationsextraktion erfordern in der Trainingsphase große Textmengen, die mit möglichst vielen Beispielen der zu extrahierenden Relation annotiert sind. Daher wird analog zur Entwicklung bei der Extraktion von Eigennamen auch an **semi-überwachten Lernverfahren zur Relationsextraktion** geforscht, die nur eine Handvoll von Trainingsbeispielen benötigen, zusammen mit sehr großen nicht-annotierten Textmengen. Agichtein und Gravano (2000) haben schon recht frühzeitig ein Verfahren zur RE auf Basis von Bootstrapping präsentiert, das den Seed-basierten Verfahren zur NER sehr ähnlich ist. Das Snowball genannte Verfahren startet mit einer kleinen Menge von Seed-Tupeln für eine zu extrahierende (binäre) Relation, z. B. korrekte Beispiele von BUCH/AUTOR oder FIRMA/HAUPTGESCHÄFTSSTELLE. Snowball sucht dann nach Sätzen, in denen eines dieser Tupel auftritt, und erzeugt entsprechende Muster, in denen die Tupellemente durch ihre entsprechenden Typen ersetzt und die anderen Elemente des Satzes entsprechend generalisiert werden. Die so bestimmten Muster werden nun selbst auf das gesamte Korpus angewendet und führen eventuell zu einer Erweiterung der Seed-Liste. Dieser Prozess wird so lange wiederholt, bis keine neuen Seed-Elemente mehr gefunden werden. Ähnliche Verfahren zum Bootstrapping finden sich auch in Brin (1998), Riloff und Jones (1999), Yangarber et al. (2000) und Xu et al. (2007).

Allgemein funktionieren diese Verfahren umso besser, je redundanter und vielseitiger eine Relation im Korpus paraphrasiert wird. Es erscheint auch klar, dass je stereotypischer eine Relation ausgedrückt wird, desto weniger Seed-Elemente anfänglich benötigt werden. Ein großer Nachteil dieser Verfahren ist auch hier (wie bereits oben im Kontext von NER erläutert) die hohe Sensibilität der Verfahren bezogen auf die initiale Seed-Liste, insbesondere, wenn Relationen auf sehr unterschiedliche Weise in Texten ausgedrückt sind.

Ein alternatives semi-überwachtes Lernverfahren, das in Bunescu (2007) vorgestellt wird, ist bzgl. dieses Problems weniger sensibel. Dieses Verfahren basiert auf dem Maschinellen Lernansatz des „Multiple Instance Learning“ (MIL; vgl. Dietterich et al. 1997). Die einzige Überwachung für das MIL-Lernverfahren ist eine kleine Menge von Paaren von Eigennamen, für die bekannt ist, dass sie entweder zur extrahierenden Relation gehören (positive Beispiele) oder nicht (negative Beispiele). Beispielsweise zeigt die folgende Tabelle 5.1 vier positive (mittels + gelabelt) und zwei negative (-) Beispiele.

Für jedes Paar wird nun eine Menge von Sätzen (auch Bag genannt) in einem Textkorpus bestimmt, in denen die zwei Argumente auftreten. Grundidee

⁶Hier kann nur stichwortartig auf diese aktuellen Entwicklungen eingegangen werden. Für ein vertiefendes Studium zum Thema „Maschinelles Lernen mit Kernels“ vgl. Schölkopf und Smola (2002) und Cristianini und Shawe-Taylor (2004).

$+/-$	Arg a_1	Arg a_2
+	Google	YouTube
+	Adobe Systems	Macromedia
+	Viacom	DreamWorks
+	Novartis	Eon Labs
-	Yahoo	Microsoft
-	Pfizer	Teva

Tabelle 5.1: Beispelpaare für positive und negative Firmenübernahmen.

des MIL Verfahrens ist nun die Annahme, dass ein Textkorpus existiert, das ausreichend groß und divers genug ist, sodass angenommen werden kann, dass in jedem Bag mindestens ein Satz enthalten sein wird, der explizit die angenommene Relation (oder ihre Negation) ausdrückt, ohne dass man das explizit manuell überprüfen muss. Ziel ist es nun, ausgehend von den Beispelpaaren und ihren assoziierten Bags automatisch ein RE-System zu induzieren, das akkurat für neue Sätze, in denen entsprechende Paare von Eigennamen genannt werden, entscheiden kann, ob die zugrundeliegende Relation gilt oder nicht. Bunescu (2007) verwendet nun MIL-Verfahren, in denen die zu lernende Entscheidungsfunktion mittels eines Kernels realisiert ist, ähnlich, wie er dies bereits für seine entsprechenden überwachten Lernverfahren zeigt, d. h. der Kernel betrachtet alle möglichen gemeinsamen Teilsequenzen zwischen zwei Sätzen. Auf diese Weise können die charakteristischen Merkmale (in Form von Teilsequenzen, die auch Lücken enthalten dürfen) jeden Bags berechnet werden. „ $\langle a_1 \rangle \dots \text{ bought } \dots \langle a_2 \rangle \dots \text{ in } \dots \text{ billion } \dots$ “ ist ein Beispiel für solch ein Merkmal. In seinen Experimenten benutzt Bunescu (2007) Sätze, die er mittels spezieller Suchanfragen an die Google-Suchmaschine bestimmt. Beispielsweise liefert die Suchanfrage „Google*****YouTube“ Dokumente, in denen Sätze vorkommen, wo zwischen den Termen „Google“ und „YouTube“ maximal sieben beliebig andere Tokens vorkommen (* fungiert demnach als Platzhalter für ein einzelnes Token). Auf diese Weise bestimmt er pro Paar Bags, die aus mehreren hundert Sätzen bestehen können, wobei auffällig ist, dass die Bags für positive Beispiele in der Regel größer sind als für negative (s. Bunescu 2007, S. 119). Für zwei untersuchte Relationen (FIRMENÜBERNAHME und PERSONGEBURTSORT) erreicht er sehr gute Werte, die das Potential dieser Methode demonstrieren. Das Verfahren von Bunescu (2007) scheint sehr fehlertolerant zu sein. Daher können die Bags vollautomatisch bestimmt werden und sind zumindest für die angegebenen Beispiele überschaubar in ihrer Größe.

Extraktion komplexer Relationen

Die bisherigen Verfahren für RE erlernen in der Regel binäre Relationen. Es gibt bereits erste maschinelle Lernverfahren zur **Extraktion von komplexen Relationen** (auch bekannt als **Ereignisextraktion** oder **Szenario-Template-Filling**) (vgl. u. a. McDonald et al. 2005, Melli et al. 2007 und Xu et al. 2007).

Als komplexe Relationen wollen wir beliebige mehrstellige Relationen betrachten, wobei einige der Argumente unspezifiziert sein können. Es ist klar, dass ein einzelner Satz nicht alle Argumente einer komplexen Relation ausdrücken muss, sondern es eher die Regel ist, dass die „Teile“ einer komplexen Relation über mehrere Sätzen verteilt sind (wie in den Beispielen in Abschnitt 5.3.1) oder aber dass die eine komplexe Relation ausdrückenden einzelnen Sätze zumindest sehr lang sind. Daher ist es nötig, potentielle Argumente aus unterschiedlichen Sätzen oder Teilsätzen zu vereinigen. In den älteren manuell erstellten IE-Systemen wurde dies meist mit einer einfachen Unifikationsstrategie behandelt: Wenn zwei Relationen identische Information in mindestens einem Argument haben und die übrigen Attribute konsistent sind, dann wird die gesamte Information beider Relationen mittels Unifikation vereinigt. Genauer wird für je zwei typkompatible Argumente überprüft, ob eine Koreferenzbeziehung besteht, ob sie semantisch kompatibel sind (via Domänenlexikon) oder ob sie in einem Subsumptionsverhältnis stehen. Darüber hinaus werden auch sehr anwendungsspezifische Heuristiken eingesetzt.

Ein elegantes Lernverfahren zur Extraktion von komplexen Relationen wird in McDonald et al. (2005) vorgestellt. Ausgangspunkt ist die Struktur der zu extrahierenden N-ären Relation in Form eines Tuples mit N Typen von Eigennamen, z. B. (PERSON, JOB, FIRMA). Kernidee ist nun, ein N-äres Tupel in die Menge aller möglichen konsistenten binären Tupel zu faktorisieren, z. B. {(PERSON, JOB), (PERSON, FIRMA), (JOB, FIRMA)}, sodass in einem ersten Schritt alle möglichen entsprechenden binären Relationen in einem Text bestimmt werden können. Hierfür können im Prinzip die oben erwähnten binären Lernverfahren herangezogen werden. Jedes gefundene Paar kann dann als eine Kante eines Graphen betrachtet werden, wobei die Eigennamen den Knoten entsprechen. Der Algorithmus versucht dann die komplexen Relationen dadurch zu rekonstruieren, indem er die **maximalen Cliquen** im Graph berechnet, wobei Knoten mit gleicher Marke (also gleichem Eigenname) die Verbindungspunkte darstellen.⁷ McDonald et al. (2005) betrachten auch gewichtete Cliquen, in denen die Kanten zwischen Knoten mit dem Wahrscheinlichkeitswert der zugehörigen binären Relation markiert ist. Dies erhöht die Fehlertoleranz, da nicht mehr eine perfekte binäre RE angenommen werden muss. Obwohl das Verfahren recht einfach ist, zeigt es eine vielversprechende Performanz, wie die Ergebnisse von Tests aus dem biomedizinischen Bereich zeigen (z. B. 0.6942 F-Maß für eine 4-stellige Relation). Ein weiterer Vorteil ist, dass das Verfahren nicht auf die Extraktion von komplexen Relationen aus einzelnen Sätzen beschränkt ist, wie dies etwa für die Verfahren von Melli et al. (2007) und Xu et al. (2007) gilt.

Domänenoffene Informationsextraktion

Die bisher betrachteten traditionellen Verfahren zur Informationsextraktion haben als Ausgangspunkt eine vordefinierte Relation und manuell überprüfte Bei-

⁷Cliquen sind Teilgraphen, in denen alle Knoten paarweise verbunden sind. Eine Clique C ist maximal, wenn es keine andere Clique gibt, die C enthält. Es gibt sehr effiziente Algorithmen zur Bestimmung aller maximalen Cliquen in einem Graphen.

spiele in Form von Seeds oder annotierten Texten. Solche IE-Systeme sind damit sehr darauf spezialisiert, einzelne Relationen so effektiv wie möglich zu extrahieren. In der Regel müssen sie aufwändig angepasst werden, wenn sie zur Extraktion von neuen Relationen eingesetzt werden sollen. Auch operieren die meisten implementierten IE-Systeme noch auf recht kleinen Textkorpora von wenigen tausend Dokumenten. Da die IE-Technologie aber eine immer stärkere Rolle in Nachbargebieten, wie z. B. Text Mining, Antwortextraktion oder Ontologien einzunehmen beginnt, erhöht sich auch damit der Bedarf an IE-Technologien, die schnell und einfach für neue Aufgaben angepasst werden können und beliebig große Textmengen beherrschen. **Wichtige Anforderungen an die zukünftige IE-Technologie** sind dabei, dass sie 1) alle sinnvollen Relationen aus Texten extrahieren können (**domänenoffen**), 2) dabei keine relationsspezifische manuell erstellte Eingabe benötigen (**unüberwacht**) und 3) prinzipiell das gesamte Web betrachten können (**webbasiert**).

Beispielsweise präsentieren Shinyama und Sekine (2006) einen Ansatz zum **unbeschränkten Entdecken von Relationen**, der anstrebt, alle möglichen Relationen aus Texten zu extrahieren und diese in Tabellen zu präsentieren. Sekine (2006) hat diesen Ansatz weiterentwickelt, den er als **on-demand information extraction** bezeichnet. Ausgangspunkt für die Entwicklung dieses Ansatzes ist ein mittels Schlüsselwörtern spezifizierter Topik, der zum Aufsammeln relevanter Webseiten verwendet wird. Das System setzt dann dependenzbasiertes Parsing als zentrales Werkzeug ein, mit dem die syntaktische Struktur möglicher relevanter Relationen bestimmt wird. Die Kandidatenrelationen werden dann mit spezialisierten Clustering-Algorithmen weiterverarbeitet. Ein ähnlicher Ansatz ist auch von Eichler et al. (2008) entwickelt worden, der diesen Ansatz mit fortgeschrittenen Benutzerinteraktionsmethoden verbindet.

Ein anderer Ansatz für das unüberwachte IE ist von Oren Etzioni und Kollegen entwickelt worden (vgl. Etzioni et al. 2005, Cafarella et al. 2005, Banko et al. 2007, Yates und Etzioni 2007 und Etzioni 2008). Sie haben eine Reihe von Systemen entwickelt (z. B. KnowItAll, Textrunner, Resolver), welche als Ziel haben, große Mengen von Fakten (z. B. Namen von Politikern oder Wissenschaftlern) aus dem Web zu extrahieren, und zwar domänenunabhängig und hoch skalierbar. Ausgangspunkt ist eine kleine Menge von generischen Mustern (nur acht Muster der Art „X such as Y“). Diese Muster werden eingesetzt, um automatisch domänenspezifische Extraktionsmuster zu erlernen. Die zwei Kernkomponenten in dieser Systemfamilie sind der Extraktor und der Bewerter. Der Extraktor generiert aus den Schlüsselwörtern jeden Musters eine Suchanfrage an eine Suchmaschine, die zur Sammlung relevanter Dokumente benutzt wird. Es werden dann dieselben Muster zur Extraktion von Kandidatenfakten verwendet. Der Bewerter berechnet einen Wahrscheinlichkeitswert zur Bewertung der Korrektheit jeden Faktums bevor diese in die Wissensbasis integriert werden. Der Bewerter basiert seine Berechnung der Wahrscheinlichkeiten auf dem Suchmaschinentrefferzähler. Dieser wird zur Berechnung einer gegenseitig abhängigen Information zwischen extrahierten Instanzen einer Klasse und der Menge der automatisch bestimmten Diskriminationsphrasen herangezogen. Diese Art der automatischen Bewertung ist eine Erweiterung des Algorithmus PMI-IR (vgl.

Turney 2001). Mit Hilfe der gesamten Methode ist es möglich, Millionen von Fakten mit einem relativen Recall von ca. 80% und einer Präzision von ca. 90% zu bestimmen. Um die Performanz weiter zu erhöhen, werden spezifische Wrapper auf der Basis von Maschinellen Lernverfahren eingesetzt, die Subklassen, Listen von Fakten und Definitionen extrahieren können. Ein ähnlicher Ansatz zur Extraktion von Definitionen und Listen aus Web-Snippets wird in Figueroa und Neumann (2008) vorgestellt (vgl. auch Abschnitt 5.3.4).

Der zentrale Flaschenhals der von Etzioni und Kollegen entwickelten Methoden ist die Fokussierung auf unäre Relationen, obwohl sie behaupten, dass ihr Ansatz zumindest theoretisch für n-äre Relationsextraktion einsetzbar ist. In einem aktuellen Papier haben Rosenfeld und Feldman (2006) das System URES vorgestellt, dass ebenfalls zur Klasse der unüberwachten IE gehört. URES ist in der Lage, große Mengen von binären Relationen zu extrahieren, wie z. B. CEO_of, InventorOf, MayorOf, wobei sie Präzisionswerte in den oberen 80% erreichen. Weiterhin haben Downey, Schoenmackers und Etzioni (2007) ein Verfahren vorgestellt, dass speziell zur Extraktion von spärlich vorkommenden Instanzen von Relationen eingesetzt werden kann. Sie zeigen insbesondere, wie unüberwachte Sprachmodelle zur Bewertung der Korrektheit verwendet werden können. Sie zeigen außerdem, dass mit Hilfe dieses neuen Ansatzes (den sie REALM nennen) eine Reduktion der Fehlerrate um 39% im Durchschnitt erreicht werden kann, wenn es mit dem Ergebnis des Systems Textrunner verglichen wird.

Das von Suchanek et al. (2007) entwickelte System YAGO ist ebenfalls ein interessanter Ansatz zur unüberwachten IE. Sie fokussieren auf die Extraktion einer sehr großen Ontologie aus Wikipedia und Wordnet, wobei sie ähnliche Verfahren wie die oben erwähnten einsetzen. Sie beschreiben Experimente, in denen sie automatisch eine Ontologie mit mehr als 17 Millionen Entitäten und Relationen mit einer geschätzten Präzision von 95% extrahieren.

Allgemeiner betrachtet ist es nun weit akzeptiert, dass mit steigender Komplexität der zu extrahierenden Information auch die Komplexität der zugrunde liegenden linguistischen Merkmalsextraktion einhergehen muss, um jeweils optimalen Recall und Präzision zu erreichen. Die meisten der aktuellen Systeme zur Relationsextraktion setzen hierzu existierende Dependenzparser, wie z. B. MontyLingua, Stanford-Parser oder Minipar ein. Allerdings mangelt es diesen Verfahren meist am nötigen Grad an Abdeckung und Geschwindigkeit, um sie realistisch in großskalierten domänenoffenen IE-Systemen einzusetzen. Glücklicherweise weisen aktuelle Forschungsergebnisse in der Computerlinguistik auf neue Richtungen für das volle Parsing von großen Korpora hin, insbesondere im Bereich der Daten-gesteuerten Dependenzanalyse. Beispielsweise präsentiert Nivre (2008) einen inkrementellen Dependenzparser, der ein Sprachmodell zur Auswertung lokaler Determinismusentscheidungen heranzieht.

5.3.4 Domänenoffene Fragebeantwortung

Text-basierte Fragebeantwortung

Eine **textbasierte Fragebeantwortung (textQA)** empfängt frei formulierte, geschriebene Fragen in natürlicher Sprache (natural language, NL) und extrahiert die entsprechenden Antworten aus sehr großen Beständen von unstrukturierten Dokumenten.⁸ Sie liefert präzise Antworten, also Textausschnitte, die genau der Antwort entsprechen (z. B. ein Personenname, ein Zeitausdruck oder eine definitorische Textpassage) und nicht nur Verweise auf Dokumente oder Paragraphen, die die Antworten (möglicherweise) enthalten, wie dies bei gängigen Suchmaschinen zur Zeit der Fall ist. TextQA-Systeme integrieren bereits eine Vielzahl von IR und IE Technologien. Ein typisches **domänenoffenes QA-System** umfasst dabei Kernkomponenten zur

1. **Frageanalyse:** Sie empfängt als Eingabe eine Frage Q in natürlicher Sprache und analysiert sie nach syntaktischen und semantischen Kriterien. Im Fall einer crosslingualen QA wird zusätzlich eine maschinelle Übersetzung der Frage durchgeführt (vgl. Abschnitt 5.3.6). Als Ergebnis liefert die Frageanalyse ein Datenobjekt O_Q , das u. a. den Fragetyp (z. B. Faktenfrage oder Definitionsfrage), den erwarteten Antworttyp (mittels Eigennamenerkennung, z. B. Person oder Lokation), eine semantische Repräsentation (oft als spezifische Relationsextraktion, vgl. Abschnitt 5.3.3), den Schlüsselwörtern und eventuell die Übersetzung beinhaltet.
2. **Dokumentensuche:** Die Schlüsselwörter werden in eine Anfrage eines IR-Systems überführt (vgl. Abschnitt 5.3.2), das damit die relevanten Dokumente bestimmt. Hierbei können auch weitere Prozesse involviert sein, wie z. B. eine Expansion der Terme, um so eine größere Trefferquote zu erreichen.
3. **Bestimmung relevanter Textpassagen:** Es werden die Textpassagen bestimmt, die mit hoher Wahrscheinlichkeit die Antworten enthalten. Eine weit verbreitete Strategie ist es, die Textpassagen zu bestimmen, in denen möglichst viele Terme der Frage in unmittelbarer Nähe zu einander auftreten (sehr häufig wird hierzu auch der $tf\text{-}idf$ -Wert herangezogen, vgl. Abschnitt 5.3.2). Die Annahme ist hier, dass Textpassagen oder Sätze, die die Antwort enthalten, oft eine hohe Ähnlichkeit mit der Frage haben („Wie hoch ist das Matterhorn?“, „Das Matterhorn ist exakt 4477,54 Meter hoch.“).
4. **Extraktion von Antworten:** Antworten sind in der Regel Instanzen des erwarteten Antworttyps (EAT). Daher werden alle Instanznamen (die in der Regel Eigennamen entsprechen) normalisiert (siehe auch Abschnitt 5.3.3), eventuell disambiguiert (vgl. Abschnitt 5.3.3) und gewichtet. Die

⁸Damit unterscheiden sie sich zentral von traditionellen QA-Systemen, die im Wesentlichen als Antwortquelle strukturierte Datenbanken heranziehen, vgl. auch Carstensen (2009b) für einen historischen Überblick.

Gewichtung eines Kandidaten a_i wird in der Regel seine Häufigkeit umfassen, was der Annahme zugrunde liegt, dass eine hochfrequente Instanz eher die gesuchte Antwort repräsentiert als eine andere, die weniger häufig genannt wird. Darüber hinaus wird die Anzahl und die jeweilige Distanz zu den Fragerterminen berücksichtigt, wobei auch syntaktische und semantische Ähnlichkeiten berechnet werden (z. B. mit dependenzbasierten Kernels, vgl. Abschnitt 5.3.3).

5. **Präsentation der Antworten:** Auf Basis der Gewichte werden die Antwortkandidaten sortiert und die N -besten Kandidaten dem Benutzer präsentiert. Um das Vertrauen in die Richtigkeit einer Antwort a_i zu bekräftigen, wird zusätzlich auch der Satz, in dem a_i auftritt, als Kontext dargestellt. Dabei muss allerdings darauf geachtet werden, dass der Satz auch semantisch konsistent ist. Dies ist keine triviale Forderung, da für a_i ja sehr viele Sätze in Frage kommen (in Abhängigkeit der Häufigkeit) und die semantische Konsistenz variieren kann. In Abhängigkeit der Frage ist auch möglich, dass Antworten im Prinzip Zusammenfassungen von Texten entsprechen können und damit die Antwortpräsentation quasi einem Textzusammenfassungssystem entspricht (vgl. auch Abschnitt 5.3.5).

Der aktuelle Erfolg in der Forschung von textQA-Systemen liegt zentral in der innovativen Kombination von Technologien aus verschiedenen Bereichen der Informatik, wie z. B. Information Retrieval, Informationsextraktion, Sprachtechnologie und Künstliche Intelligenz (s. Moldovan et al. 2004). Bedeutsam für den aktuellen Erfolg ist sicher auch die Tatsache, dass textQA-Systeme und Technologien seit 1999 im Rahmen der nordamerikanischen TREC Konferenzreihe (<http://trec.nist.gov/>), sowie seit 2003 im Rahmen der europäischen CLEF Initiative (<http://www.clef-campaign.org/>) systematisch evaluiert werden (vgl. auch Carstensen 2009b).

Webbasierte QA

Die Adaption und Skalierung dieser neuen textQA-Technologien für das Web und damit für die Weiterentwicklung aktueller Suchmaschinentechnologien stellt allerdings enorme Herausforderungen bereit, insbesondere auch unter der Vorgabe, dass die Systemreaktionszeit nicht signifikant steigen darf (s. Radev 2003). Man betrachte beispielsweise die riesige Menge an Webinhalten, die die gegenwärtig besten Suchmaschinen zu indizieren in der Lage sind (Milliarden von Webseiten). Während sich eine automatische linguistische und semantische Indizierung von TREC ähnlichen Datenmengen (ca. 1GB Nachrichtentext) zur Beantwortung einer eingeschränkten Menge von Fragen mittels sprachtechnologischer Vorverarbeitung als sehr fruchtbar für die Entwicklung der textQA-Technologien erwiesen hat (wobei Aspekte der Systemreaktionszeit meist noch keine Rolle spielen), erscheint die gleiche Vorgehensweise für das Web zumindest gegenwärtig außer Reichweite. Zudem sind die TREC und CLEF Korpora statisch, da sie auf eine kurze und fixe Zeitspanne beschränkt sind. Im Web werden neue Dokumente permanent hinzugefügt und bereits existierende Dokumente können jederzeit

verändert oder gelöscht werden. Systeme zur **webbasierten Fragebeantwortung (webQA)** sind damit verstärkt mit Problemen wie Redundanz, Aktualität und Zuverlässigkeit konfrontiert. Es muss dabei auch bedacht werden, dass ein Text in der Regel einen hohen Interpretationsspielraum umfasst, so dass eine statische, situationsunabhängige Analyse des Textes nur eingeschränkt möglich ist. Hinzu kommt die wachsende Mehrsprachigkeit der Webinhalte und der damit steigende Bedarf an einer crosslingualen Fragebeantwortung.

Es existieren bereits erste webQA-Systeme, die erfolgreich zeigen, wie durch Einsatz moderner Sprachtechnologie gepaart mit einer systematischen Ausnutzung der Redundanz der Webinhalte fortgeschrittene Technologien für zukünftige Antwortmaschinen erreichbar sind (vgl. Ankolekar et al. 2006, Clarke et al. 2001, Dumais et al. 2002, Kwok et al. 2001, Ravichandran und Hovy 2002, Neumann und Xu 2004, Wahlster 2007). Diese Systeme verfügen über eine vergleichbare Architektur und führen im Wesentlichen die folgenden drei Schritte durch:

1. Transformation einer NL-Frage in eine Anfrage der spezifischen Suchmaschine.
2. Bestimmung von relevanten Webseiten durch eine standardisierte Suchmaschine, z. B. Google, Live Search, Yahoo.
3. Extraktion der Antworten aus den Webinhalten.

Der erste Schritt ist notwendig, um optimal die Anfragesyntax der jeweiligen Suchmaschine zu bedienen und um die Zugriffswahrscheinlichkeit auf relevante Dokumente zu erhöhen. Dieser letzte Aspekt kann auch als die Bestimmung einer „Antwortkontext-Voraussage“ interpretiert werden. Beispielsweise wird hiermit eine NL-Frage *Wie heißt der deutsche Bundespräsident?* überführt in *der deutsche Bundespräsident heißt* oder *Deutschlands Bundespräsident ist*. Ohne Berücksichtigung einer vorherigen Korpusanalyse entspricht diese Art der Transformation prinzipiell einer Versuch-und-Irrtum Strategie. Daher verwenden fortgeschrittene webQA-Systeme eine statistische Datenanalyse zur Bestimmung einer optimalen Gewichtung der einzelnen Antwortkontakte (s. Dumais et al. 2002 und Ravichandran und Hovy 2002).

Was den zweiten Schritt betrifft, so liefern Suchmaschinen in der Regel nur Verweise auf relevante Dokumente zusammen mit automatisch erstellten Textausschnitten (**Snippets** genannt). Sie bestehen aus kleinsten Textfragmenten, in denen Terme der Suchanfrage vorkommen. Daher werden in der Regel für den dritten Schritt zuerst die N-besten Dokumente auf den lokalen Rechner geladen, bevor die eigentliche Antwortextraktion stattfindet (s. Kwok et al. 2001 und Ravichandran und Hovy 2002). Die **Antwortextraktion** wird dann analog der textQA-Technologie mit Hilfe von kombinierten Sprach- und Wissenstechnologie-Komponenten durchgeführt.

Die Notwendigkeit, relevante Dokumente dynamisch zu laden, hat allerdings einen negativen Effekt auf die Reaktionszeit des gesamten webQA-Systems. Dumais et al. zeigen, dass für die Beantwortung von faktenbasierten Fragen die **Snippets als Antwortquellen** ausreichen können, wenn zum Einen eine Vielzahl von Antwortkontexten systematisch berechnet und zum Anderen eine große

Menge von Snippets (einige hundert bis tausend) für die Antwortextraktion berücksichtigt wird (s. Dumais et al. 2002). Die Snippets enthalten Terme aus der Suchanfrage und heben diese entsprechend markiert hervor. Für die Antwortextraktion ist dies von Vorteil, da es oft der Fall ist, dass eine mögliche Antwort in unmittelbarer Nähe der Fragereme auftritt. Daraus kann aber nicht unmittelbar geschlossen werden, dass die im Kontext solcher Schlüsselwörter vorkommenden Terme automatisch sinnvolle Antwortkandidaten wären. Insbesondere kann daraus nicht abgeleitet werden, dass die Snippets der ranghöchsten Dokumente auch am wahrscheinlichsten die richtige Antwort enthalten.

Es ist ein zentrales Ziel der aktuellen Suchmaschinen, den besten Dokumenten einen hohen Rang zuzuordnen und nicht Dokumente zu präferieren, deren Snippets möglicherweise einen exakten Antwortstring enthalten (Suchmaschinen sind zur Zeit ja noch keine Antwortmaschinen). Daher kann die **Rangordnung**, wie sie durch aktuelle Suchmaschinen vorgegeben wird, nicht unmittelbar für die Bestimmung der Antwortkandidaten herangezogen werden. Denn es besteht ein relevanter Unterschied darin, ob die Suchmaschine aktuelle Webseiten oder Webseiten mit möglichst hoher Zugriffszahl präferiert. Dies ist z. B. für die Beantwortung von Fragen nach Personen des öffentlichen Interesses kritisch. Wurde beispielsweise gerade ein neuer Bundespräsident (oder Bundesligatrainer) gewählt, so ist die Zugriffszahl auf Webseiten mit diesem Inhalt zumindest vorübergehend niedrig im Vergleich zu Seiten, die auf vorherige Bundespräsidenten verweisen. Aus diesem Grunde kann man in der Antwortextraktion zwar die N-ersten Snippets betrachten, sollte danach aber deren Rangordnung ignorieren (vgl. Neumann 2008). Das heißt das in diesem Fall die N-Snippets als Menge betrachtet wird. Es ist dann ein Ziel, während der Antwortextraktion eine **antwortspezifische Rangordnung** von relevanten Textausschnitten zu bestimmen.

Aktuelle webQA-Systeme haben zur Zeit noch erhebliche Beschränkungen. Zum Einen betrifft das die Anfrage-gesteuerte Berechnung potentieller Antwortkontexte. Zum Anderen sind die Systeme in der Regel nur für die Verarbeitung bestimmter Arten von NL-Fragen (meist einfache faktenbasierte Fragen der Art „Wer/Wann/Wo“) optimiert. Hinzu kommt, dass die aktuell relevanten Systeme monolinguale (überwiegend englischsprachige) Systeme sind.

Datengesteuertes QA

Um auch im Bereich der QA eine der IE vergleichbaren Leistungssteigerung bezüglich Adaptivität und Skalierbarkeit zu erreichen, stehen gegenwärtig datengesteuerte Methoden für end-to-end QA-Systeme im Fokus der Forschung und Entwicklung. Die eingesetzten Verfahren sind denen der Relationsextraktion und domänenoffenen IE sehr ähnlich (vgl. Abschnitt 5.3.3).

Ramakrishnan et al. (2004) präsentieren einen Ansatz zum automatischen Erlernen von oberflächennahen Textmustern, die die Methode des Maximum Entropy Modellings zum Gewichten von Snippets heranzieht. Ihr QA-System lernt automatisch Strategien, um Antwortpassagen zu entdecken und zu gewichten. Als Trainingsmaterial verwenden sie Frage-Antwortpaare von TREC-Korpora.

Das sind in der Regel kurze Nachrichtentexte. Sie verwenden wissensintensive Methoden zur Merkmalsextraktion, u. a. volles Parsing und WordNet zur Disambiguierung.

Ein instanzbasierter Ansatz für QA ist von Lita und Carbonell (2004) entwickelt worden. Sie betrachten insbesondere temporalrestringierende Fragen aus den TREC-Korpora unter Verwendung eines sehr großen Fragekorpus. Sie stellen einen sehr interessanten Ansatz für das Clustern von Fragen vor, der auf dem strukturellen Vergleich von Fragen unter Verwendung von POS-Tagging und vollem Parsing basiert. Bei der Evaluation fokussieren sie auf webbasierte Snippets, die die korrekte Antwort enthalten, nicht jedoch auf die Extraktion der exakten Antworten.

In Figueroa et al. (2008) und Figueroa und Neumann (2008) werden Maschinelle Lernverfahren zur Beantwortung von Definitions- und Listenfragen aus Web-Snippets vorgestellt, die als gemeinsamen Kern eine Kombination von Clustering und Semantische Kernels realisieren. In einem ersten Schritt werden die Fragen automatisch so reformuliert, dass Standardsuchmaschinen optimal angesteuert werden können. Ähnlich dem Verfahren zur domänenoffenen IE (siehe Abschnitt 5.3.3) wird eine sehr kleine Menge von generischen Reformulierungsmustern verwendet (z. B. „ $\langle Q \rangle$ is a $\langle A \rangle$ “) mit denen konkrete Anfragen an eine Suchmaschine generiert werden. Die gleichen Muster werden ebenfalls zur Bestimmung der besten Textfragmente aus den gefundenen Snippets herangezogen, nachdem sie ebenfalls automatisch expandiert wurden. Zur Extraktion von Wörtern, die unterschiedliche Aspekte des Zielkonzeptes im Web repräsentieren, werden Korpus-basierte semantische Analysen (insbesondere Latent Semantic Analysis) und Disambiguierungsstrategien eingesetzt. Die Verfahren sind mit Frage/Antwort-Paaren der TREC und CLEF-Korpora evaluiert und die erzielten Ergebnisse sind vergleichbar mit denen der dort vorgestellten besten Systeme.

Basierend auf ihren eigenen Arbeiten an AskMSR erforschen Azari et al. (2003) die Verwendung von statistischen Modellen im Kontext von webbasierten QA-Systemen. Auch sie verfolgen eine strenge datengesteuerte Perspektive, die nur über sehr einfache sprachspezifische Wissensquellen verfügt. Im Wesentlichen führen sie einfache Reformulierungen und Mustervergleiche basierend auf N-Gramm-Technologien durch. Damit generieren sie sehr elegant (sehr viele) Paraphrasen zur aktuellen Frage, die unmittelbar zur Suche mit einer IR-Suchmaschine verwendet werden. Um zu optimalen Reformulierungsstrategien zu gelangen, wird anschließend eine Kostenanalyse auf Basis von erlernten Entscheidungsbäumen für alle erfolgreichen Frage-Antwort Paare durchgeführt.

Echihabi und Marcu (2003) präsentieren eine Methode, die auf dem probabilistischen Noisy-Channel Model zur Bestimmung der Ähnlichkeit zwischen Fragen und Antwortsätzen basiert. Ihr bestes Ergebnis von MRR=0.812 wird für 24 Fragen von der TREC-11 auf Basis einer Trainingsmenge von ca. 100.000 Frage-Faktenantwort Paaren erzielt. Unter Verwendung eines Korpus von ca. 90.000 allgemeineren Frage-Antwort Paaren erreichen sie einen MRR zwischen 0.343 und 0.365 für 500 TREC-11 Fragen. Im Unterschied zu den oben erwähnten Verfahren verwenden sie sehr wissensintensive Verarbeitungskomponenten und eine sehr viel größere Trainingsmenge.

5.3.5 Textzusammenfassung

Die Ergebnispräsentation hat zum Ziel, die gefundenen Informationen (z. B. Dokumente, Textpassagen, Antworten, gefüllte Tabellen) dem Benutzer in verständlicher Form so darzustellen, dass möglichst rasch bewertet werden kann, ob die extrahierten Informationen tatsächlich relevant sind oder zumindest so weit hilfreich sind, dass der Benutzer sie für seine weitere Informationsrecherche nutzen kann. Daher ist es auch ein Ziel der Ergebnispräsentation, die gefundenen Informationen kompakt so **zusammenzufassen**, dass die wichtigsten Informationen „auf einen Blick“ erkennbar sind. Dabei hängt es natürlich auch vom quantitativen Umfang der Information ab, wie einfach und vollständig dies geleistet werden kann.

Menschen produzieren beim Zusammenfassen meist kurze geschriebene oder gesprochene Texte (vgl. Endres-Niggemeyer 2004). Wahrscheinlich auch deshalb, weil kurze Texte vom Menschen effektiv und schnell konsumiert werden können. Die **automatische Textzusammenfassung** beschäftigt sich mit Methoden der automatischen Erzeugung von textuellen Zusammenfassungen, wobei das Medium der Informationsquellen im Prinzip beliebig sein kann, z. B. Videos, Bilder, gesprochene oder geschriebene Sprache oder auch Kombinationen von verschiedenen Medien. In diesem Abschnitt wollen wir uns auf die Verarbeitung von geschriebener Sprache konzentrieren, also die automatische Erstellung von Textzusammenfassungen aus geschriebenen Textquellen.

Suchmaschinen versuchen dies aktuell z. B. dadurch zu leisten, indem neben der Rangordnung der Dokumente zu jedem Dokument bereits ein Textausschnitt (**Snippet** genannt) berechnet wird, der die Terme der Anfrage im Kontext ihres Auftretens im Dokument zeigt, vgl. auch Abschnitt 5.3.4 und Manning, Ragavan und Schütze (2008). Die Snippets sollen dem Benutzer eine kurze und prägnante Erklärung liefern, warum das Dokument mit der Anfrage zusammenpasst. In diesem Sinne kann ein **Snippet** als kurze **Zusammenfassung** eines einzelnen Dokumentes betrachtet werden, bestehend aus dem Titel des Dokuments, dem Link zum Dokument und einem kleinen Textausschnitt. Im Prinzip kann man zwei verschiedene Methoden unterscheiden, wie diese kleinen Textausschnitte berechnet werden: **statisch** und **dynamisch**.

In einem statischen Verfahren wird – unabhängig von der konkreten Anfrage – stets ein bestimmter Ausschnitt aus den gefundenen Dokumenten bestimmt, z. B. bestehend aus den ersten N-Sätzen des Dokuments eventuell in Kombination mit vorhandener Meta-Information, wie etwa dem Typ des Dokumentes (PDF etc.) oder dessen Erzeugungsdatum. Eine statische Zusammenfassung wird oft zur Beantwortung von Definitionsfragen verwendet. Beispielsweise liefert die Anfrage „define:DFKI“ an die Google-Suchmaschine als Ergebnis (Snippet hier kursiv hervorgehoben):

Definitionen von **DFKI** im Web:

- Das Deutsche Forschungszentrum für Künstliche Intelligenz (DFKI) ist ein Public Private Partnership mit Großunternehmen, Mittelständlern, den ...
de.wikipedia.org/wiki/DFKI

Definitionen finden zu **DFKI** in: [Deutsch](#) [Englisch](#) [alle Sprachen](#)

wobei dieser Text übereinstimmt mit der ersten Zeile in der entsprechenden Seite in Wikipedia.

Dynamische Verfahren berechnen die Snippets in Abhängigkeit einer konkreten Anfrage (engl. **query-biased summarization**), so dass sichergestellt wird, dass mindestens ein Term der Anfrage im Snippet enthalten ist. Dies ist quasi automatisch der Fall, da ja die Terme über den invertierten Index mit den Dokumenten direkt in Verbindung stehen, u. a. auch dadurch, dass zu jedem Term die Position im jeweiligen Dokument gespeichert ist (vgl. Abschnitt 5.3.2). Damit kann sehr einfach und schnell zu jedem Term ein kleiner Textausschnitt bestimmt werden, z. B. in dem im Dokument die N-linken und rechten adjazenten Wörter eines Terms bestimmt werden. Für mehrere Terme, die im Dokument vorkommen, kann damit auch recht einfach bestimmt werden, ob sie in unmittelbarer Nähe zu einander stehen, z. B. im gleichen Satz. Wenn dies nicht der Fall ist und die jeweiligen Textausschnitte zudem nicht aufeinander folgen, dann kann man dies mittels ... anzeigen, wie dies beispielsweise für folgenden Snippet gilt, der von der Google-Suchmaschine für die Anfrage „aktuelle Blue-ray Player“ berechnet wird:

[Top 9: Die besten Blu-ray-Player im Test – CHIP Online](#)

1. Dez. 2008 ... Wir haben die **aktuellen** Stand-Alone-Geräte unter die Lupe genommen und sagen, ... Wer über die Anschaffung eines **Blu-ray-Players** nachdenkt, ...

Zwar können mit diesen einfachen Methoden *sehr schnell* dynamische Snippets erzeugt werden, die auch für einen ersten Überblick sehr hilfreich sind. Allerdings können so zum Einen in der Regel keine kohärenten Texte erzeugt werden, und zum Anderen ist jeder Snippet mit einem einzelnen Treffer (einem einzelnen Dokument) direkt verknüpft. Damit werden sie erst dann für den Benutzer quasi sichtbar, wenn der entsprechende Ausschnitt der Trefferliste angezeigt wird.

Eine andere Möglichkeit der **Textzusammenfassung** wäre es daher, einen einzelnen, sprachlich und inhaltlich abgestimmten Text für eine Menge von Textdokumenten zu erzeugen. Die zentralen Probleme für solche **Zusammenfassungen aus mehreren Dokumenten** (engl. **multidocument summarization**) sind a) das Erkennen und die Behandlung von redundanten Textstellen, b) die Identifikation von wichtigen Unterschieden in den verschiedenen Dokumenten und c) das Sicherstellen von kohärenten Zusammenfassungen, sogar dann, wenn die Dokumente aus verschiedenen Quellen (Autor, Stilrichtung etc.) stammen (vgl. Radev et al. 2002).

Zur Bestimmung der Redundanz werden oft alle Paragrafen oder Sätze der Dokumente auf Grund ihrer Ähnlichkeit mittels Clustering gruppiert, vgl. auch Abschnitt 5.3.2. Dann wird in einem nächsten Schritt für jedes relevante Cluster ein einzelner Text quasi als Stellvertreter bestimmt. In einem anschließenden Schritt werden dann die verschiedenen Textausschnitte zu einem Text zusammengefügt. Dies ist in der Regel ein sehr komplexer Prozess, da die Texte sprachlich und inhaltlich aufeinander abgestimmt werden müssen. Daher werden hierbei auch Verfahren der **Textgenerierung** eingesetzt (vgl. Abschnitt 5.6 und Endres-Niggemeyer 2004 für weitere Details).

5.3.6 Multilinguale und sprachübergreifendes TIM

Multilingualität spielt im Zeitalter des Webs eine immer größere Rolle (vgl. dazu auch <http://www.internetworldstats.com/stats.htm>). Von wachsender Bedeutung sind dabei auch sprachübergreifende (**crosslinguale**) Verfahren mit Hilfe derer Informationsquellen aus verschiedenen Sprachen gemeinsam genutzt werden können. Kerntechnologien für ein TIM sind hierbei:

- **Crosslinguale Information-Retrieval** (CLIR): erlaubt Benutzern Anfragen in einer Sprache zu formulieren, die sie fließend beherrschen und führt eine Volltextsuche in Dokumenten durch, die in einer anderen Sprache formuliert sind.
- **Crosslinguale Question Answering** (CLQA): findet exakte Antworten in Dokumenten einer Sprache für natürlichsprachliche Fragen, die in einer anderen Sprache formuliert ist.
- **Crosslinguale Informationsextraktion** (CLIE): extrahiert und integriert relevante Informationen (Entitäten und Relationen) aus Dokumenten von verschiedenen Sprachen.

In allen Bereichen werden Verfahren der **Maschinellen Übersetzung** (MÜ) als zentrales Mittel zur Realisierung der Crosslingualität eingesetzt. In den Bereichen CLIR und CLQA werden dabei ähnliche Verfahren eingesetzt. Die meisten Verfahren gehen dabei von bilingualen Verfahren aus, d. h. die **Zielsprache** (die Sprache der Anfragen) und die **Quellsprache** (die Sprache in denen die Dokumente verfasst sind) stammen jeweils aus einer einzigen aber jeweils unterschiedlichen Sprache. In CLQA wird häufig die Frage der Zielsprache zuerst mit MÜ-Verfahren in die Quellsprache der Dokumente übersetzt, bevor der eigentliche Suchprozess gestartet wird (vgl. Sacaleanu und Neumann 2006).

Ein großes Problem für CLIR und CLQA ist es, dass die Anfragen in der Regel sehr klein sind (verglichen mit der Größe von Dokumenten, siehe auch Abschnitt 5.3.2), sodass die Qualität der Übersetzung einen hohen Einfluss auf die Qualität des gesamten Systems hat. Beispielsweise kann ein Fehler in der Übersetzung zu weiteren Fehlern in der Frageexpansion und -umformulierung führen. Daher ist es auch eine häufig eingesetzte Strategie gerade in CLIR, die Dokumente der Quellsprache in die Zielsprache zu übersetzen und den Suchprozess in der Zielsprache durchzuführen (vgl. Hakkani-Tur et al. 2005). Ein Vorteil ist dann

auch, dass man die Ergebnisse aus verschiedenen Quellsprachen sehr einfach vermischen kann, da man ja nun nur einen einzigen Index zu berechnen und verwalten hat. Dabei kann man auch berücksichtigen, ob für Dokumente der Quellsprache bereits (manuelle) Übersetzungen vorliegen (so genannte parallele Texte). Damit kann man sich die Übersetzung ersparen und gleichzeitig den Redundanzwert (oder *tf-idf*-Wert, vgl. Abschnitt 5.3.2) der Terme erhöhen.

Zentrales Ziel bei der CLIE ist die Identifikation und Extraktion von relevanter Information aus Dokumenten mit unterschiedlichen Ziel- und Quellsprachen. Sudo et al. (2003) zeigen, wie ein CLIE auf Basis eines CLIR realisiert werden kann. Ihr Ansatz ist quasi eine crosslinguale Version einer on-demand IE (ODIE, vgl. Abschnitt 5.3.3). Die Kommunikation zwischen Benutzer und IE ist in englischer Sprache, d. h., die Anfrage in Form einer englischen Frage und die Ausgabe in Form von englischen Tabellen. Die eigentliche Extraktion findet in japanischen Texten mit Hilfe eines japanischen IE-Systems (JIE) statt. Daher wird zuerst die Anfrage ins Japanische übersetzt und damit die eigentliche IE angestoßen. JIE erzeugt Muster über Dependenzstrukturen dem Ansatz von ODIE folgend. Die erzeugten Tabelleneinträge werden dann Slot für Slot ins Englische übersetzt. Dieser CLIE-Ansatz kann auch als „anfragegesteuerte IE“ (query-driven IE) oder **zielsprachenorientierter Ansatz** bezeichnet werden. Sudo, Sekine und Grishman (2004) vergleichen diesen Ansatz mit **quellsprachenorientierten Ansätzen**, wo die Dokumente der Quellsprache in zielsprachliche Äquivalente übersetzt werden. Ihre Experimente und Evaluationen zeigen, dass die anfragegesteuerte IE einen 8–10% höheren Recall aufweisen. Einer der Gründe ist hierbei die Fehleranfälligkeit von MÜ-Systemen, insbesondere was die Übersetzung von Eigennamen betrifft. Große Probleme bewirkt ebenfalls die fehlerhafte Übersetzung von ganzen Sätzen, die relevante Relationen verbalisieren, da sie erhebliche Auswirkungen auf die nachfolgende Syntaxanalyse haben, sodass die Extraktion von Relationen in den übersetzten Dokumenten erschwert wird.

5.3.7 Perspektiven

Text-basiertes Informationsmanagement (TIM) integriert Technologien aus verschiedenen Bereichen der Sprachtechnologie, wie Information-Retrieval (IR), Informationsextraktion (IE), Fragebeantwortung (QA) und Textzusammenfassung (TZ). In diesem Artikel haben wir versucht zu zeigen, wie diese verschiedenen Perspektiven kohärent in einem Gesamtsystem zusammenspielen können. Dabei haben wir uns auf solche Aspekte konzentriert, die quasi als Basistechnologie betrachtet werden können. Viele wichtige Aspekte und Forschungsrichtungen konnten leider nur sehr knapp oder teilweise gar nicht angesprochen werden, wie z. B. Forschungs- und Entwicklungsrichtungen in den Bereichen des Semantic Webs, der Digitalen Büchereien oder spezifische Probleme und Lösungsvorschläge in bestimmten Bereichen, wie z. B. der Bioinformatik. Zusammenfassend für die zukünftige Forschung gilt, dass IR, IE, QA und TZ sich immer stärker aufeinander zu bewegen und verschmelzen werden.

5.3.8 Literaturhinweise

Die im Artikel angegebenen Literaturhinweise sollten bereits eine gute Basis für vertiefende Studien bieten. Darüberhinaus bieten die aktuellen Konferenzreihen TREC (Text Retrieval Conference, <http://trec.nist.gov/>), CLEF (Cross-Language Evaluation Forum, <http://clef-campaign.org/>) und TAC (Text Analysis Conference, <http://www.nist.gov/tac/>) einen sehr guten Einstiegspunkt in aktuelle Forschungsrichtungen und Entwicklungen. Da im Rahmen dieser Konferenzreihen auch regelmäßig vergleichende Systemevaluationen stattfinden, bieten sie auch einen sehr guten Einblick in die aktuelle Leistungsfähigkeit von IR, IE, QA und TZ Methoden. Weitere Hinweise zu Literatur und speziellen Webseiten kann man auch den Seiten von Wikipedia entnehmen, z. B. zum Thema „Information Retrieval“, „Information Extraction“ (deutsche und englische Seiteneinträge) und „Question Answering“ (nur englische Seite). Eine sehr informative Seite zum Thema „Text Summarization“ ist über den Link <http://www.summarization.com/> zu erreichen.

5.4 Sprachein- und -ausgabe

Elmar Nöth und Volker Fischer

Ebenso wie die in Unterkapitel 5.3 beschriebenen Anwendungen zur computer-gestützten Erstellung und Verarbeitung *klassischer* Textdaten ist auch das Spektrum an Systemen zur Verarbeitung gesprochener Sprache durch den strukturellen Wandel von Wirtschaft und Gesellschaft hin zu einer globalen Informationsgesellschaft geprägt. Manifestiert sich dieser Wandel einerseits in der stetig wachsenden Bedeutung des Internets als Informationsquelle, so zeigt andererseits die rasante Zunahme an (internetfähigen) Mobiltelefonen – gegenwärtig dürfen wir von weltweit etwa 3 Milliarden Geräten ausgehen – dass der Zugriff auf diese Informationen künftig vorrangig per Sprache und über das öffentliche Telefon- und Mobilfunknetz erfolgen wird. Die sprachgestützte Mensch-Maschine-Interaktion zeigt sich dabei nicht nur als *conditio sine qua non* für den ubiquitären Zugang zu Informationen, sondern kann sich im Zeichen einer zunehmenden Miniaturisierung von Endgeräten zugleich als wirkungsvolle Gegenmaßnahme gegen technisch komplexe und nur schwer zu beherrschende Benutzerschnittstellen erweisen.

Vor diesem Hintergrund – jedoch nicht ohne die militärische Nutzung und den Einsatz zur Kostensenkung bzw. Rationalisierung unerwähnt zu lassen – beschreiben die folgenden Abschnitte eine Reihe von Anwendungen, in denen Spracherkennung und -synthese zum Einsatz kommen.

5.4.1 Spracheingabe

Aus technisch-wissenschaftlicher Sicht sind kontinuierliche Fortschritte in der akustischen Modellierung mit Hidden-Markov-Modellen – zu nennen sind insbesondere Methoden zur Online-Adaptation (Sprecheranpassung zur Laufzeit) und zur Modellierung verrauschter Sprachdaten –, die Verfügbarkeit großer und repräsentativer Sprachdatenbanken in zahlreichen Sprachen (vgl. Unterkapitel 4.5), und die Etablierung geeigneter Standards zur Entwicklung von *Voice-User-Interfaces*, z. B. VoiceXML, hauptverantwortlich für ein beständig wachsendes Anwendungsportfolio.

Diente automatische Spracherkennung vor etwa 10 bis 15 Jahren noch nahezu ausschließlich der Informationseingabe, so hat sich der Schwerpunkt der am Markt verfügbaren Anwendungen seither zugunsten von Systemen zur Informationsabfrage und zur telefongestützten Durchführung von Transaktionen – also in Richtung der Mensch-Maschine-Interaktion – verschoben. Systeme zur Verschriftung spontansprachlicher Äußerungen und multimedialer Daten (Nachrichtensendungen etc.) betreten in jüngster Zeit zusätzlich die Szene; sie ermöglichen die inhaltliche Erschließung geschäftsrelevanter Fakten sowie die Archivierung und Suche in audio-visuellen Datenströmen. In loser Ordnung wird das Spektrum derartiger Anwendungen in den folgenden Abschnitten skizziert.

Einzelworterkenner: Einzel- und Verbundworterkenner mit einem Wortschatz von nicht mehr als 100 Wörtern werden hauptsächlich für *Kommando-*

systeme eingesetzt, die einerseits mit wenigen Befehlen auskommen, andererseits aber auch unter extrem schwierigen Bedingungen (Nebengeräusche) zuverlässig funktionieren müssen, und denen – zumeist aus Kostengründen – nur wenig Rechenleistung und Speicher zur Verfügung steht.

Typische Anwendungsbeispiele sind Maschinensteuerung, Not-Ausschalter, Sprach- und Namenswahl beim (Mobil-)Telefon und die Steuerung von Komfortfunktionen (Radio, Klimaanlage) im Fahrzeug. Neben Verfahren der statistischen Modellierung mittels Hidden-Markov-Modellen und *phonembasierten* Spracherkennern kommen dabei auch abstandsmessende Klassifikatoren auf der Basis des DTW (*dynamic time warping*, vgl. Unterkapitel 2.4) und Ganzwortmodelle zum Einsatz.

Diktiersysteme: Diktiersysteme sind PC-basierte, für den Einsatz in einer relativ ruhigen Büroumgebung konzipierte Spracherkennner mit großem Wortschatz, welche die gesprochene Sprache eines kooperativen Sprechers in geschriebenen Text umwandeln. Kommerziell erhältliche Programme stellen in der Regel auch zusätzliche Funktionen zur sprachgesteuerten Bedienung des Textverarbeitungsprogrammes und des PC-Desktops bereit.

Diktiersysteme verwenden stochastische Sprachmodelle, in der Regel Bigramm- oder Trigramm-Modelle, zur linguistischen Modellierung des Anwendungsbereichs (vgl. Unterkapitel 2.4). Erste Systeme mit einem Wortschatz von bis zu 50.000 Wörtern waren bereits vor etwa 15 Jahren erhältlich; sie verlangten jedoch sowohl eine isolierte Sprechweise mit kurzen Pausen zwischen den Wörtern als auch eine sprecherspezifische Anpassung der akustischen Modelle, die dem Benutzer das Aufsprechen eines phonetisch balancierten Trainingstextes aufbürdete. Heutige Diktiersysteme besitzen einen Wortschatz von bis zu 1.000.000 Wörtern, der durch Fachvokabulare aus zahlreichen Themengebieten (Wirtschaft, Jura, Medizin etc.) erweitert werden kann. Sie erlauben eine natürliche (kontinuierliche) Sprechweise mit einer Geschwindigkeit von bis zu 140 Wörtern pro Minute und erzielen Worterkennungsraten von deutlich über 95 Prozent. Eine Sprecher- bzw. Umgebungsanpassung ist optional und in der Regel nur noch bei starkem Dialekt oder minderwertigen Mikrofonen erforderlich.

Telefoniesysteme: *Interactive-Voice-Response*-Systeme (IVR) müssen mit einer auf die Bandbreite des Telefonkanals reduzierten Sprachqualität arbeiten, können sich zur *Online*-Sprecheranpassung bestenfalls auf wenige Sekunden Sprachmaterial verlassen und dürfen darüber hinaus keineswegs auf die Kooperationsbereitschaft eines geübten Benutzers (hier: des Anrufers) setzen, der sein Anliegen in grammatisch korrekter Rede vorträgt.

Erste Anwendungen zur Ablösung sogenannter *Touchtone*-Systeme (Zifferneingabe über die Tastatur des Telefons) gaben dem Anrufer daher eine stark eingeschränkte Syntax vor („*Sagen Sie ,eins*‘, wenn *Sie Fragen zu Ihrer Rechnung haben.*“), und besaßen eine strikt hierarchische Menüführung. Moderne Systeme erlauben zunehmend direktere, natürlichsprachliche Formulierungen („*Ich hab’ da ’mal ’ne Frage zu meiner Rechnung.*“). Dies wird oft durch die Modellierung der Benutzeräußerungen mittels sehr umfangreicher VoiceXML-Grammatiken erreicht. Solche Systeme lassen echte Benutzerfreundlichkeit durch die Vorgabe eines relativ starren Dialogverlaufs jedoch immer noch vermissen.

Telefonische Auskunftssysteme, beispielsweise zur Statusabfrage von Zug- oder Flugverbindungen, Buchungs- oder Transaktionssysteme (Flugreservierung, Telefon-Banking) und Systeme zur Qualitätssicherung – etwa im Call-Center-Bereich zur Erfassung der Kundenzufriedenheit – sind aktuell realisierte Anwendungen.

Transkriptionssysteme: Versehen mit einem ähnlich großen Wortschatz wie moderne Diktiersysteme und ebenfalls unter Verwendung stochastischer N-Gramm-Sprachmodelle, dienen Transkriptionssysteme der Verschriftung *sponsorer Sprache*, die in unterschiedlicher Qualität (Bandbreite, Umgebungsgeräusche) vorliegen kann.

Einsatzgebiete sind einerseits die automatische Verschriftung von Radio- und TV-Sendungen und andererseits die automatische Protokollierung von gesprochenen Dialogen zwischen Menschen, beispielsweise bei Geschäftstreffen und Telefongesprächen. Transkriptionssysteme dürfen somit als eine Schnittstellen-technologie verstanden werden, die gesprochene Sprache sowohl für sämtliche Textanalyse-Methoden (Archivierung, Suche, Klassifikation, Zusammenfassung etc.) als auch für die maschinelle Übersetzung zugänglich macht.

Telefonüberwachung: Die automatische Auswertung von aufgezeichneten Gesprächen (meistens Telefongespräche, seltener Sprechfunk) stellt einen Teilbereich des Protokollierens von Mensch-Mensch-Kommunikation dar. Ein Großteil der Anwendungen betrifft Sicherheitsbehörden (Polizei, Militär, Nachrichtendienste). Aber auch im Call-Center-Umfeld gibt es wichtige Anwendungen im Bereich Qualitätskontrolle, wobei hier die Äußerungen des Agenten untersucht werden: Für den Call-Center-Betreiber ist von Interesse, ob sich der Agent an vorgegebene Regeln hält und ob das Kundengespräch vorgegebenen Qualitätsansprüchen genügt: Ist der Agent stets höflich geblieben? Hat er sich mit seinem Namen gemeldet? Wurde das Gespräch häufig unterbrochen, weil der Agent eine Rückfrage bei einem Kollegen einholen musste (und somit eine innerbetriebliche Fortbildung besuchen sollte)?

Diese Art der Qualitätskontrolle wird bisher von Mitarbeitern durchgeführt, die im laufenden Betrieb $n\%$ der Gespräche mithören (n im Bereich $\approx 1\%$). Die Effektivität der Qualitätssicherung lässt sich jedoch durch die Verwendung von automatischen Transkriptionssystemen enorm verbessern. Durch den automatischen Ausschluss von eindeutig problemlosen Fällen (Anrufe, in denen sich der Agent korrekt verhalten hat) wird der Anteil der im Sinne der Qualitätssicherung kritischen Anrufe, die einer Kontrolle unterzogen werden, gesteigert. Auch die Suche etwa nach Telefonaten, die Rauschgifthandel betreffen oder nach Funkverkehr, der für einen Terroranschlag relevant sein könnte, entspricht der Suche nach der bekannten Nadel im Heuhaufen. Da nur ein kleiner Prozentsatz der Gespräche „von Hand“ kontrolliert werden kann, geht es auch hier darum, den Anteil der relevanten Gespräche zu erhöhen.

Sowohl im Call-Center-Umfeld als auch im Sicherheitsbereich kann es sinnvoll sein, statt einer vollständigen Transkription nur eine Schlüsselwortsuche durchzuführen. Dabei werden szenarioabhängig ca. 50 bis 100 Schlüsselwörter festgelegt, die eine Kategorisierung des Gesprächs erlauben (z. B. „*Herzlichen Dank*“ als Hinweis auf ein freundliches Gespräch), und nach denen dann im Gespräch

gesucht wird. Zwar reduziert sich hierdurch die extrahierbare Information deutlich, aber ein „Word Spotter“ ist schneller auf neue Gesprächsthemen übertragbar und mit weniger Sprachmaterial trainierbar als ein Transkriptionssystem.

Sprechererkennung, Landessprachenerkennung: Gerade im zuletzt genannten Anwendungsszenario sind weitere Gesprächsmerkmale von Interesse, also z. B. *wer* etwas in *welcher Sprache* gesprochen hat. Für viele Sprachen gibt es überhaupt keine Transkriptions- oder Schlüsselwortsysteme, so dass sich die Vorauswahl der Telefonate neben technischen Daten (Verbindungsnummern) nur auf solche Information gründen kann. Sprecherinformation ist von Interesse, da in der Regel Datenbanken mit älteren Daten der Zielpersonen vorliegen und man gerne wüsste, ob der Anrufer bekannt ist. Landesspracheninformation kann wichtig sein, da häufig nur Gespräche einer bestimmten Sprache von Interesse sind. Sprechererkennung spielt natürlich auch in zivilen Anwendungen eine Rolle, etwa bei biometrischen Zugangskontroll- oder Autorisierungsverfahren. Prinzipiell werden ähnliche Verfahren verwendet, allerdings liegt hier der Schwerpunkt auf Verifikationsaufgaben: der Sprecher behauptet, dass er die Person *XY* ist und das System überprüft diese Aussage. Im obigen Fall hingegen liegt eine Identifikationsaufgabe vor (die Sprachprobe muss einem von n Sprechern zugewiesen werden).

Sprechergruppenerkennung: Etwas allgemeiner ist die Aufgabe, nicht die Identität des Sprechers, sondern seine Zugehörigkeit zu einer Sprechergruppe, etwa *männlich/weiblich, älter als 50, Muttersprachler ja/nein* festzustellen. Auch hier gibt es kommerzielle und Sicherheitsanwendungen. Im kommerziellen Umfeld kann die Zugehörigkeit zu einer Sprechergruppe z. B. dazu benutzt werden, um den Anrufer dem nächsten „passenden“ (beispielsweise gleichaltrigen) Agenten zuzuordnen, die Prompts eines automatischen Systems sprechergruppenabhängig auszuwählen, oder um sprechergruppenabhängig Werbung einzuspielen.

Emotionserkennung, Sprecherzustandserkennung: Ein weiterer Aspekt des Sprachsignals ist, *wie* etwas gesprochen wurde. Neben den klassischen Emotionen (Freude, Ärger, Trauer, ...) spielen auch andere Zustände (unsicher, verwirrt, gelangweilt, ...) eine Rolle, weshalb häufig der allgemeinere Begriff *Sprecherzustand* verwendet wird. Informationen über den Sprecherzustand können dazu benutzt werden, die Dialogstrategie eines automatischen IVR-Systems an den Benutzer anzupassen (z. B. mehr oder weniger ausführliche Hilfetexte ausgeben) und um verärgerte Kunden mit einem Agenten zu verbinden. Im Bereich der Telefonüberwachung ist es beispielsweise von Interesse, ob ein Call-Center-Agent immer höflich blieb (hierfür ist sowohl der Sprecherzustand als auch die Wortwahl, s.o., wichtig) und ob ein abgehörter Sprecher aufgereggt war. Ein großes Problem ist es, an realistische, szenariotypische Trainingsdaten zu gelangen.

Medizinische Anwendungen: Viele Erkrankungen wirken sich auch auf die Kommunikationsfähigkeit eines Menschen aus. Der Bogen reicht hier von angeborenen Störungen wie *Lippe-Kiefer-Gaumenspalte* über verwascene, unverständliche Sprache aufgrund eines Schlaganfalls bis hin zu Einschränkungen aufgrund demenzieller Erkrankungen. Der Patient wird aufgefordert, einen Referenztext zu lesen oder gezeigte Gegenstände zu benennen. Durch den Vergleich

der erkannten Wörter mit dem Referenztext lässt sich dann feststellen, wie verständlich der Patient ist und ob durch sprachtherapeutische Maßnahmen ein Fortschritt erzielt wurde. Durch den Vergleich des akustischen Signals mit einem Referenzmodell aus möglichst vielen Kontrollsprechern und Sprechern mit der gleichen Sprechstörung lässt sich die Schwere der Beeinträchtigung abschätzen. Bei kontinuierlicher Sprache werden auch die prosodische Information sowie die Position und Dauer der Sprechpausen berücksichtigt. Sprechpausen können ein wichtiger Hinweis auf Wortfindungsstörungen sein; die Variation der Tonhöhe hat Auswirkung auf die Natürlichkeit und Verständlichkeit des Sprechers.

e-Learning: Im Bereich des e-Learning betrifft die Anwendung der Spracheingabe zwei Bereiche. Zum Einen wird im Bereich des Fremdsprachenlernens mit ähnlichen Methoden wie bei den medizinischen Anwendungen festgestellt, wie weit die Aussprache des Lernenden von der Aussprache eines oder mehrerer Referenzsprecher abweicht. Zum Anderen lässt sich der Verlauf eines computergestützten Unterrichts durch die Erfassung des Sprecherzustands anhand der gesprochenen Antworten steuern. Erkennt das System, dass der Benutzer unsicher ist, kann es als nächste Lektion ein Repetitoriumskapitel einfügen; ist der Benutzer dagegen unkonzentriert und müde, kann es ein Spiel einschieben.

Computer-Spiele: Bei Computer-Spielen, insbesondere bei Online-Multiplayersystemen, ist nicht nur die sprachliche Bedienung von Interesse, sondern auch der Sprecherzustand. Durch die Bestimmung des aktuellen Zustands und eine entsprechende Steuerung des Spiels (z.B. Auftreten bzw. Nicht-Auftreten einer neuen Gefahrensituation) lässt sich das Erfolgsergebnis des Spielers steigern. Da die hier angesprochene Zielgruppe sehr technikaffin ist, ist ein hoher Akzeptanzgrad zu erwarten.

Maschinelle Übersetzung gesprochener Sprache: Die Übersetzung von gesprochener Sprache hat in den letzten Jahren einen rasanten Aufschwung genommen. Die Gründe hierfür liegen zum Einen in großen Fortschritten insbesondere bei der statistischen Übersetzung und zum Anderen in dem stark angewachsenen Interesse an Simultanübersetzungen aufgrund der veränderten politischen und wirtschaftlichen Randbedingungen. Dies führte zu hohen finanziellen Anstrengungen und der Erstellung großer Korpora in den Zielsprachen (insbesondere für die Quellsprachen *Arabisch* und *Chinesisch* liegen verschriftete Sprachdaten in der Größenordnung von 2.000 Stunden vor).

Kombiniert mit den Anstrengungen bei den Transkriptionssystemen für Nachrichtensendungen und den (aufgrund großer paralleler Korpora) stark verbesserten Ergebnissen bei der statistischen Übersetzung von Texten werden bei der Übersetzung von gesprochenen Nachrichten bereits beachtliche Erfolge erzielt. Zwar führt der statistische Ansatz zu typischen Syntaxfehlern in der Zielsprache, doch kann die Bedeutung der Nachricht von Menschen in der Regel gut erfasst werden (die Ergebnisse sind vergleichbar mit Google-Translate, wobei zu bedenken ist, dass die Eingabe – die am besten passende Wortkette – sehr fehlerhaft sein kann). Für den mobilen Einsatz gibt es Systeme mit stark reduziertem Vokabular und eingeschränkten Szenarien. So wird von der US-Armee im Irak ein bidirektionales Sprach-Sprach-Übersetzungssystem für *US-Englisch* und *irakisches Arabisch* eingesetzt, um in typischen alltäglichen Situationen mit

der irakischen Armee und der Zivilbevölkerung kommunizieren zu können („Bitte zeigen sie Ihren Pass!“, „Was ist das Ziel Ihrer Reise?“, ...).

Perspektiven: Gegenwärtige Forschungstrends gehen hin zu noch alltags-tauglicheren Systemen. Dazu gehört insbesondere die Erkennung von Sprache, die mit Distanzmikrofonen (Raummikrofonen) erfasst wird, die Erkennung von Sprache, die mit starken, nichtstationären Hintergrundgeräuschen behaftet ist und die kontinuierliche automatische Erweiterung des Wortschatzes. Man geht davon aus, dass pro Jahr ca. 20.000 Wörter in einer Sprache entstehen bzw. verschwinden. Solche neuen, „Out-of-Vocabulary“-Wörter sind insbesondere für das Textverständnis von mit Transkriptionssystemen erfassten Nachrichten sehr wichtig.

5.4.2 Sprachausgabe

Frühe Arbeiten zur maschinellen Erzeugung menschlicher Sprache lassen sich bis weit vor die Entwicklung der ersten Computer zurückverfolgen. Sowohl Wolfgang von Kempelen um 1790 entstandene mechanische Sprechmaschine, die als Nachbau im Deutschen Museum in München zu besichtigen ist, als auch Homer Dudleys elektrischer Voder, eine manuell zu bedienende Sensation der Weltausstellung 1939 in New York, dienten zwar vorrangig der Unterhaltung ihres Publikums, benutzten jedoch beide Erkenntnisse über die menschliche Sprachproduktion (vgl. Unterkapitel 3.1), die zumindest teilweise auch in heutigen, computergestützten Sprachsynthesesystemen Verwendung finden.

Anwendungen: Sprachsynthese wird überall dort eingesetzt, wo Bildschirm oder Display als Ausgabegerät nicht zur Verfügung stehen oder aus anderen Gründen nicht genutzt werden können. Neben telefonischen Auskunfts- und Dialogsystemen (Reiseauskunft, Helpdesk, Telefonbanking etc.) stehen damit insbesondere mobile Szenarien im Fokus der Anwendungsentwicklung (z. B. Navigationsgeräte und Fahrzeugkomfortsteuerung im Auto); etablierte Anwendungsbiete wie die Unterstützung Sprechbehinderter oder etwa der Computerarbeitsplatz für Blinde und Sehbehinderte existieren weiterhin und dürften durch die Forderung nach einem barrierefreien Zugang zu Informationen – gerade auch für ältere Menschen – zukünftig eine Aufwertung erfahren.

Synthesequalität, Änderungsfreundlichkeit, und Ressourcenverbrauch der jeweiligen Applikation – neben Hauptspeicherbedarf und Rechenleistung gehören hierzu auch die Kosten für die Entwicklung einer synthetischen Stimme – hängen von der eingesetzten Synthesetechnik ab, weshalb sich die folgende *methodenzentrierte* Einordnung der verfügbaren Anwendungen anbietet.

Pre-recorded prompts: Bestmögliche Qualität erzielen Systeme, die sich auf die Wiedergabe vorgefertigter Audiodateien, sogenannter *pre-recorded prompts*, beschränken. Aus Sicht der Sprachsynthese sind derartige Systeme nicht weiter interessant, da sie weitestgehend ohne computerlinguistische Verarbeitung auskommen; sind jedoch Neuarrangement und Verkettung kurzer „Sprachbausteine“ vorgesehen, so stellt sich Entwicklern, Toningenieurinnen und Sprechern zumindest die Aufgabe, mehrere prosodisch verschiedene Varianten aufzunehmen, um hörbare Artefakte an den Nahtstellen zu vermeiden. Durch ihre äußerst geringe

Flexibilität – es kann nur wiedergegeben werden, was einmal aufgenommen wurde – sind diese Systeme nur in sehr eingeschränkten Szenarien, beispielsweise in Navigationssystemen („*Biegen Sie nach 100 Metern rechts ab.*“), einsetzbar. Zwar existieren auch einige größere *Voice-Portale*, insbesondere im Bereich des Telefon-Bankings, die sich vorgefertigter Prompts bedienen; die mit relativ hohen Kosten verbundenen Änderungen machen die Technik aber dennoch zu einer aussterbenden Spezies.

Formantsynthese: Bis vor etwa 15 Jahren definierte die computergestützte Simulation der menschlichen Sprachproduktion durch ein parametrisches *Source-Filter*-Modell den Stand der Technik. Der Luftstrom am Ausgang der Larynx wird als – je nach Phonationsart zufälliges oder periodisches – Anregungssignal (Quelle, *Source*) eines zeitlich variablen *Filters*, des Vokaltrakts, angesehen. Die Resonanzfrequenzen dieses Filters werden als Formanten bezeichnet; sie sind entscheidend für die Unterscheidung lautlicher Einheiten und daher namensgebend für das Verfahren. Formantsynthese-Systeme sind *Vollsynthese*-Systeme, die beliebigen Text in gesprochene Sprache konvertieren können. Da die Modellparameter in der Regel offenliegen und über spezielle Markup-Sprachen (*SSML, speech synthesis markup language*) manipuliert werden können, sind auch Sprechstil und Charakter der synthetischen Stimme leicht zu ändern. Die erzeugte Sprache wird von Testpersonen häufig als unnatürlich, aber auch als sehr gut verständlich eingestuft. Daher war die Formantsynthese über lange Zeit ein wichtiges Hilfsmittel bei der Unterstützung Seh- und Sprechgeschädigter. Ein weiterer Vorteil der Formantsynthese liegt in ihrem äußerst geringen Ressourcenverbrauch – komplexe Systeme lassen sich schon mit weniger als zwei Megabyte Speicherplatz realisieren –, der auch den Einsatz auf mobilen Geräten ermöglicht.

Konkatenative Synthese: Nahezu natürliche und sehr verständliche synthetische Sprache wird durch die Verwendung konkatenativer Verfahren erzielt, die im Gegensatz zur Formantsynthese nicht auf die Simulation menschlicher Spracherzeugung setzen, sondern vielmehr auf die geeignete Auswahl, Neuordnung und Verkettung (sehr) kurzer Segmente natürlicher Sprache. Konkatenative Syntheseverfahren erzeugen eine prosodisch-linguistische Spezifikation des Eingabetextes und wählen anschließend die Segmente eines akustischen Inventars aus, die optimal zu dieser Spezifikation passen; sie werden daher auch als *Unit-Selection*-Verfahren bezeichnet. Mit einem akustischen Inventar auf der Subwortebene – typischerweise werden Silben, Laute oder sogar subphonetische Einheiten verwendet – sind konkatenative Verfahren ebenfalls in der Lage, beliebigen Text zu synthetisieren.

Die Sprachqualität profitiert sowohl von einer guten Abdeckung des Anwendungsbereites im aufgenommenen Ausgangsmaterial als auch von reichhaltigen prosodischen Variationen der Basiseinheiten, und skaliert daher in der Regel mit Speicherbedarf und erforderlicher Rechenzeit. Konkatenative Systeme benötigen typischerweise 50 bis 1.000 Megabyte Hauptspeicher und werden daher insbesondere in Telefonieranwendungen und gelegentlich auch zusammen mit vorgefertigten Prompts eingesetzt. Ferner existieren PC-basierte Anwendungen, die das Vorlesen von Textdokumenten oder ganzen Web-Seiten ermöglichen, oder im Bereich des Erlernens von Fremdsprachen anzusiedeln sind. Mit reduziertem

Segmentinventar und/oder geeigneten Kodierungs- und Kompressionsverfahren lösen konkatenative Systeme mittlerweile aber auch in mobilen Szenarien, beispielsweise in handelsüblichen Navigationsgeräten, die Formantsynthese ab.

Perspektiven: Aktuelle Forschungsschwerpunkte betreffen insbesondere „emotionale“ und multilinguale Synthese. Nicht nur in Computerspielen erwartet der Hörer eine möglichst natürliche Interaktion, die an geeigneter Stelle von einer neutralen Intonation abweichen sollte (allerdings ist hier die zu erzeugende Emotion besser bestimmbar als etwa bei einem Vorleseautomaten für Nachrichten oder Bücher). Multilinguale Synthese ist insbesondere wichtig, um Fremdwörter korrekt wiedergeben zu können (man denke nur an den Gebrauch von Anglizismen).

5.4.3 Literaturhinweise

Breit angelegte Übersichten über die mathematisch-technischen Grundlagen der hier beschriebenen Anwendungen finden sich in den ausgezeichneten Lehrbüchern von Jurafsky und Martin (2009), Huang et al. (2001), und – für die Sprachsynthese – Sproat (1998) und Taylor (2009). Zahlreiche Aspekte des Entwurfs von telefonischen Auskunftssystemen beschreiben Cohen et al. (2004) und Kotelly (2003). Liegt der Schwerpunkt dort auf Beispielen aus dem nordamerikanischen Markt, so werden angebotene Lösungen, aktuelle Entwicklungen und zukünftige Trends aus Sicht der im deutschen Markt vertretenen Firmen in Feldt et al. (2008) skizziert.

5.5 (Multimodale) Dialogsysteme

Tilman Becker

Natürlichsprachliche Dialogsysteme ermöglichen die sprachliche Interaktion mit Computersystemen aller Art. Die Eingabe kann per Tastatur oder durch gesprochene Sprache erfolgen; die Komplexität reicht dabei von einfachen Kommandowörtern bis hin zu freier Rede. Die Systemausgaben können analog als Text auf dem Bildschirm oder als synthetische Sprache erfolgen. Telefonische Auskunfts- oder Bestellsysteme sind wohl die bekanntesten Anwendungen.

Multimodale Dialogsysteme (Wahlster 2006) erweitern Sprachdialogsysteme um weitere Modalitäten. Auf der Eingabeseite sind dies Gesten, Tastatur- oder Stifteingabe, Blick- oder Gesichtsausdruckerkennung; typische Ausgabemodalitäten sind neben dem Bildschirm Objekte in der realen Welt (z. B. ein Roboter) oder einer virtuellen Umgebung (z. B. ein animierter virtueller Agent).

Durch die stärkere Einbettung in eine (reale oder virtuelle) Welt entsteht eine wesentlich bessere situative Integration und damit ein größeres Bedürfnis nach natürlicher Interaktion. Dies stellt besondere Anforderungen an alle, aber insbesondere an die Sprachverarbeitungskomponenten. Dialoge in multimodalen Systemen sind typischerweise frei im Dialogablauf, erlauben *spontansprachliche*, unvollständige Eingaben (und Ausgaben) und freie Formulierungen.

Multimodale Interaktionssysteme können auch mit stark eingeschränkter sprachlicher Interaktion auskommen, und grundsätzlich auch ganz ohne. Durch leistungsfähigere (natürlichere) Interaktion in multimodalen Systemen steigt aber im Allgemeinen das Bedürfnis nach natürlichsprachlicher Interaktion, so dass in der Praxis die meisten Systeme *sprachzentrierte* multimodale Dialogsysteme sind.

Es gibt eine große Vielfalt von Anwendungsfeldern für multimodale Dialogsysteme, z. B. im Auto zur Steuerung von Radio, Navigation oder Mobiltelefon. Mit virtuellen Agenten werden sie als Informationskiosk oder interaktive Installation in Museen verwendet. Allgemein dienen multimodale Dialogsysteme als Auskunftssysteme, zur Administration, zur Steuerung und zur Unterhaltung.

5.5.1 Multimodale Kommunikation

Der hier verwendete Begriff der **Modalität** leitet sich aus der Physiologie ab und bezieht sich auf die Sinnesmodalitäten wie Hören, Sehen, Riechen, usw.⁹ In einem etwas allgemeineren Sinne bezeichnen wir jeden Kommunikationskanal zwischen Benutzer und Computersystem als eine Modalität.

Die wichtigste Eingabemodalität ist natürliche Sprache und tritt auf als gesprochene Sprache, aber auch als Tastatureingabe oder Handschrift. Eine weitere oft genutzte Eingabemodalität sind bildschirmbezogene Eingaben, die über eine Maus, einen Zeigestift oder einen Touchscreen erfolgen können und sich auf

⁹Dagegen bezeichnet der sprachwissenschaftliche Begriff Modalität die Modulation der logischen Satzaussage, etwa durch Modalverben.

die Bildschirmdarstellung beziehen, oder diese gar direkt manipulieren können. Über Kameras können aber auch Zeigegesten im Raum, die Position oder die Blickrichtung erfasst werden.

Darüber hinaus können aus dem Sprachsignal, aus der Körperhaltung und insbesondere aus dem Gesichtsausdruck Informationen über den emotionalen, affektiven Zustand des Benutzers erfasst werden.

Die neben der Sprachausgabe wichtigste Ausgabemodalität ist meist die Bildschirmdarstellung von Systemausgaben. Zusätzlich verwenden einige Systeme animierte Agenten, die als Avatar des Systems die Funktion des Dialogpartners übernehmen. Dabei können alle genannten Eingabemodalitäten bis hin zur Generierung von Emotionen auch auf der Ausgabeseite verwendet werden (Klesen und Gebhard 2007).

Wenn das Dialogsystem eine externe **Applikation** steuert, z. B. ein Fernsehgerät oder einen MP3-Player, müssen deren Ausgaben auch als Ausgabekanal berücksichtigt werden.

Ein Beispieldialog mit dem SmartKom-System (Wahlster 2006), Abbildung 5.2, verwendet neben Sprache auf Ein- und Ausgabeseite auch Zeigegesten, grafische Ausgabe und einen animierten Agenten („Smartakus“):

BENUTZER: *Dort möchte ich sitzen [Zeigegeste auf den Saalplan].*

SMARTAKUS: *Ist das so richtig
[Zeigegeste auf farbig markierte reservierte Plätze]?*

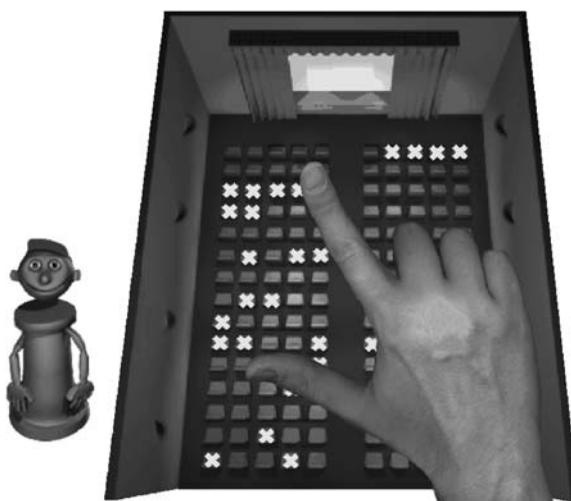


Abbildung 5.2: Multimodale Platzreservierung im Kino.

Integration der Modalitäten

Im einfachsten Fall werden zusätzliche Eingabekanäle (bzw. Ausgabekanäle) getrennt von der Spracheingabe betrachtet und liefern nach der Analyse Eingaben im gleichen internen Datenformat. Z. B. kann eine *STOPP*-Taste die gleiche Funktionalität wie das gesprochene Kommando „*STOPP!*“ haben.

Tatsächlich bestehen aber vielfältige, enge Beziehungen zwischen den Modalitäten. Zeigegesten etwa ergänzen oft referentielle Ausdrücke und unterliegen engen zeitlichen Beschränkungen in Relation zur sprachlichen Äußerung: „*Ich möchte dieses [Zeigegeste] Lied hören.*“ ist nur eindeutig, wenn die Zeigegeste in zeitlicher Nähe zum Wort „*dieses*“ erfolgt. Manche Benutzer zeigen allerdings ein sequentielles Verhalten, dann folgt die Zeigegeste unmittelbar vor oder nach der Äußerung (Oviatt et al. 2005). Eine korrekte Interpretation von ironischen Äußerungen ist nur durch zusätzliche Information, etwa aus Betonung, Gesichtsausdruck oder Körperhaltung möglich: „*Das ist ja toll.*“ mag wörtlich gemeint sein, aber kann auch das genaue Gegenteil aussagen.

Die integrierte Analyse der Eingaben in den verschiedenen Modalitäten wird **Fusion** genannt und hat ihr Äquivalent auf der Ausgabeseite, wenn die Systemausgabe auf verschiedene Modalitäten verteilt wird und sprachliche Referenzen auf andere Modalitäten generiert werden oder gar ein virtueller Agent animiert wird. In Analogie zur Fusion wird dieser Prozess auch **Fission** genannt.

In der Modellierung des Diskurses (aktueller Zustand, Geschichte, ...) müssen die Informationen auch bezüglicher ihrer Modalitäten repräsentiert werden. Dies betrifft Eingabe wie Ausgabe. Der folgende Dialog gibt ein Beispiel.

SYSTEM: *Diese Alben stehen zur Auswahl [Display zeigt die Albumcover].*

BENUTZER: *Das Album da rechts [Keine Geste].*

Die Benutzeräußerung kann nur verstanden werden, wenn entsprechende Eigenschaften in der Ausgabemodalität Grafik – in diesem Fall die räumliche Anordnung – im Kontext gespeichert sein. Diese Information ist weder in der Sprachausgabe, noch im inhaltlichen Kontext enthalten.

5.5.2 Sprachdialogsysteme

Ein wesentlicher Aspekt gesprochener Dialoge ist die **Robustheit**, also die sinnvolle Fortführung des Dialogs, auch wenn Eingaben unvollständig sind oder nur teilweise verstanden werden. Außerdem müssen gesprochene Dialoge in **Echtzeit** stattfinden. Reaktionszeiten, die einige Sekunden überschreiten, machen eine natürliche Interaktion unmöglich. Damit stellen sich hohe Anforderungen an die verwendeten Konzepte und Technologien in Dialogsystemen.

Eine wichtige Entscheidung, mit weitreichenden Konsequenzen für das Verständnis und das Design (multimodaler) Dialogsysteme, ist die Frage nach den Grenzen des Dialogsystems. Ein Dialogsystem kann als einfache **Benutzerschnittstelle** (User Interface) betrachtet werden, das lediglich der Ansteuerung

eines externen Geräts, z. B. eines Fernsehers oder Roboters, oder einer Softwareapplikation dient. Dann ist das Dialogsystem über eine **Programmierschnittstelle** (API, Application Programming Interface) an die Applikation angebunden und die Funktionsweise der Applikation muss im Dialogsystem gesondert modelliert werden. Eine integrierte Sichtweise hingegen begreift sämtliche Funktionalität als Teil des Dialogsystems und erlaubt eine natürlichere Interaktion. Da Dialogsysteme oft getrennt von der Applikation oder erst nachträglich entwickelt werden, ist die Sichtweise als Benutzerschnittstelle häufiger. Abbildung 5.3 in Abschnitt 5.5.3 zeigt die allgemeine Architektur eines solchen Systems.

Die Sichtweise als Benutzerschnittstelle wird bei den einfachsten Sprachdialogsystemen sehr deutlich. **Kommandowortsysteme** bilden jede Eingabe direkt in eine Funktionalität der Applikation ab und können daher nicht als echte Dialogsysteme betrachtet werden. Moderne Betriebssysteme, z. B. auch in Mobiltelefonen, integrieren solche Funktionalität bereits zur Steuerung einfacher Funktionen.

Dialogische Interaktion entsteht erst durch Eingaben (und Systemausgaben), die nicht direkt zur Programmierschnittstelle geleitet werden. Dies sind z. B.

- inkrementelles Zusammensetzen einer Eingabe durch einzelne Fragen, z. B. nach Start und Ziel einer Reise bei der Zugauskunft (diese Methode wird auch *slot-filling* genannt),
- Klärungsfragen, wenn Benutzereingaben nicht verstanden wurden oder nicht eindeutig sind,
- durch den Benutzer angestoßene Metadialoge, z. B. die Bitte um Wiederholung der Systemausgabe oder um Hilfe,
- Metafunktionalität, z. B. Zugang zu mehreren Applikationen, etwa die Wahl zwischen Radio und Navigation im Auto.

Einfache Kommandosysteme müssen sich dabei nicht zwangsläufig auf Kommandoworte beschränken, auch ganze Sätze oder gar freie Formulierungen einschließlich spontansprachlicher Freiheiten sind möglich; entscheidend ist die Fähigkeit bzw. Einschränkung, nur vollständige Kommandos zu verstehen. Ohne dialogische Interaktion muss der Benutzer die Funktionalität der Applikation dabei genau kennen.

Sobald dialogische Interaktion hinzukommt, gibt es ein weiteres Charakterisierungsmerkmal, die **Ablaufkontrolle**. In einfachen Systemen wird der Dialogablauf dabei vollständig durch das Dialogsystem bestimmt. Das System präsentiert alle möglichen Eingaben, oft in einer Menü-ähnlichen Struktur, aus denen der Benutzer auswählen muss. Zusätzlich werden vom System passende Fragen gestellt, die der Benutzer unmittelbar beantworten muss. So führt das System den Benutzer Schritt für Schritt durch die Interaktion mit der Applikation. Das wichtigste Beispiel für diese Art der Ablaufkontrolle ist **VoiceXML** (McGlashan 2004), ein W3C-Standard, der die Grundlage der meisten Telefonbasierten Sprachdialogsysteme bildet.

Toolkits zur Entwicklung von Sprachdialogsystemen basieren auf solchen zustandsbasierten Dialogmodellen, die auch grafisch als endliche Automaten (siehe Unterkapitel 2.2) dargestellt werden können. Dabei erweitern sie jedoch die Funktionalität beim Übergang von einem Zustand zum anderen z. B. durch Systemhilfe und Ausgabevariationen. Die entstehenden Dialogabläufe entsprechen der typischen Bedienung der Applikation und erlauben wenig Flexibilität.

Während bei Kommandosystemen der Benutzer die volle Kontrolle über die Interaktion hat, wechselt bei einfachen Sprachdialogsystemen die Kontrolle also vollständig zum System. Solange die dabei vorgegebenen Abläufe gut an die Applikation und das Benutzerverhalten angepasst sind, ist dies keine Einschränkung. Komplexere (multimodale) Dialogsysteme erlauben aber neben der Systemkontrolle jederzeit auch freie Eingaben des Benutzers. In diesem Fall spricht man von *mixed initiative* oder **gemischter Initiative**.

5.5.3 Struktur eines multimodalen Dialogsystems

Abbildung 5.3 zeigt die verschiedenen Module, aus denen ein multimodales Dialogsystem besteht.¹⁰ Dabei ist die Verarbeitungsreihenfolge nicht zwingend sequentiell, auch wenn die Verkettung der Module dies nahelegt. Insbesondere können spätere Module Rückmeldungen an frühere Module geben und eine Neubearbeitung auslösen.

Komponenten-Architektur

Der erste Schritt in der Verarbeitung gesprochener Sprache ist die **Spracherkennung**. Dabei können heutzutage kommerziell verfügbare Systeme eingesetzt werden, die sprecherunabhängig und in Echtzeit arbeiten und damit die oben genannten Anforderungen für eine natürliche Interaktion erfüllen. Zur Verbesserung der Spracherkennung kann das System durch Training auf einzelne Sprecher adaptiert werden und durch ein **Sprachmodell** oder durch eine **Spracherkennерgrammatik** auf die Domäne zugeschnitten werden, siehe Unterkapitel 3.2. Das vom W3C definierte Format SRGS (Speech Recognition Grammar Specification; Hunt und McGlashan 2003) wird von vielen Spracherkennern unterstützt. Als Ausgabe der Spracherkennung wird der erkannte Text weitergeleitet. Bei freier Eingabe entstehen durch *spontansprachliche* Effekte zusätzliche Schwierigkeiten, denn Verzögerungen, Wiederholungen, Stottern, Selbstkorrekturen usw. müssen im Spracherkennner berücksichtigt werden. Solche Versprecher können durch automatische Systeme, die auf großen Datenmengen trainiert wurden sind, entfernt werden (Germesin et al. 2008).

Im nächsten Schritt werden die erkannten Worte in der **Sprachinterpretation** analysiert, um die Intention des Benutzers zu erfassen. Dabei wird auf die verschiedenen Teile des Kontextes (in der Mitte der Abbildung 5.3) zugegriffen. Im ersten Schritt kommen robuste Parsingtechnologien zum Einsatz, die

¹⁰Die Darstellung lehnt sich an die „Referenzarchitektur“ von multimodalen Dialogsystemen aus Bunt et al. (2005) an.

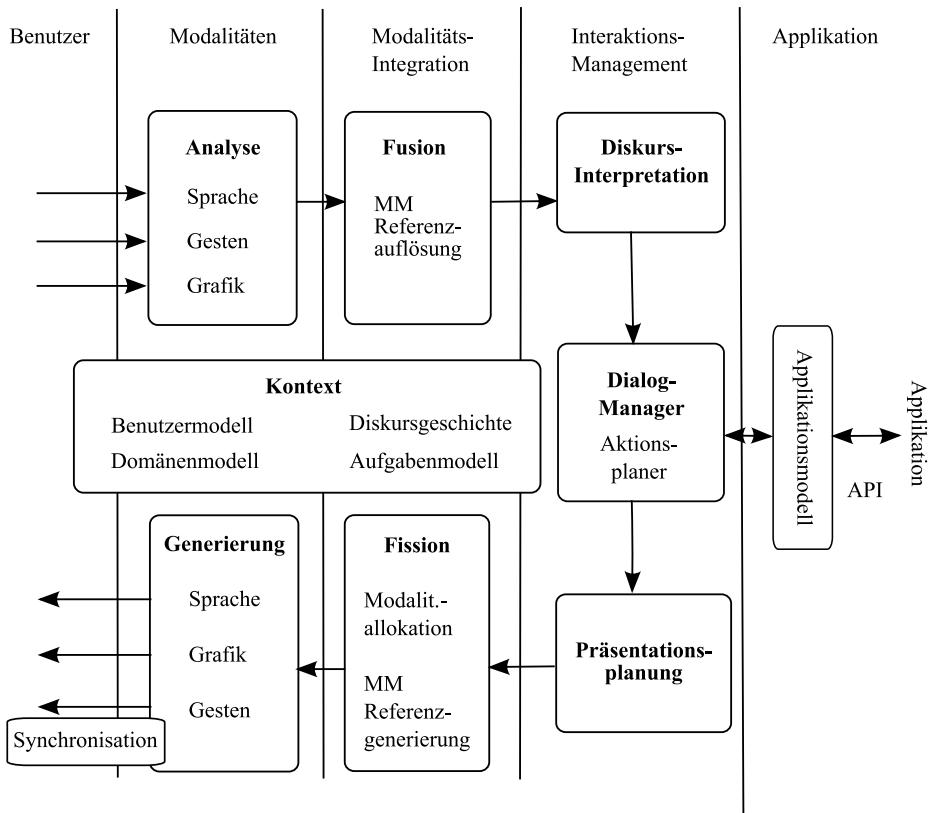


Abbildung 5.3: Die Architektur eines multimodalen Dialogsystems.

von reinen Schlüsselworterkennern bis zu semantischen Parsern reichen, siehe Unterkapitel 3.4 bis 3.6.

Auch hier müssen die Besonderheiten der spontansprachlichen Eingabe durch entsprechende Robustheit berücksichtigt werden. Insbesondere müssen auch partielle Analysen möglich sein, wenn Teile der Eingabe nicht verstanden werden. Entsprechende Verarbeitungsschritte werden parallel auch für die übrigen Modalitäten durchgeführt.

Auf die Modellierung der Intention geht der folgende Abschnitt 5.5.4 genauer ein. Verkürzt dargestellt werden die Intentionen, die in den verschiedenen Modalitäten erkannt und verstanden werden in der **multimodalen Fusion** zusammengesetzt. Die dabei auftretenden Phänomene wurden schon in Abschnitt 5.5.1 angesprochen. Das Ergebnis der Fusion muss nun in der **Diskursinterpretation** weiterverarbeitet werden um die Eingabe im Kontext des bisherigen Diskurses zu verstehen. Die Eingabe „*Das finde ich gut.*“ kann eine deiktische Referenz sein, oder aber die Bestätigung der vom System reservierten Sitze im Kino, also eine Anapher, siehe Unterkapitel 3.7. Eine weitere Aufgabe der Dis-

kursinterpretation ist das schrittweise Zusammensetzen der Benutzerintention über mehrere Interaktionen hinweg (*slot-filling*).

Zu diesem Zeitpunkt ist die Benutzereingabe vollständig analysiert und die Systemreaktion wird im **Dialogmanager** berechnet. Dabei liegt die oben besprochene Ablaufkontrolle zugrunde und der Dialogmanager steuert gegebenenfalls die Applikation an. Dabei wird in komplexeren Dialogsystemen eine Applikationsmodellierung genutzt, die Aktionen des Aufgabenmodells des Dialogsystems, etwa „eine Sendung aufzeichnen“ in die einzelnen Schritte umsetzt, die das API der Applikation zur Verfügung stellt.

Neben dem reinen Sprechakt, etwa zu informieren oder zu fragen, plant der Dialogmanager auch in einem ersten Schritt die Inhalte der Ausgabe. In der **Präsentationsplanung** werden die Inhalte genauer festgelegt. Unter Berücksichtigung der zur Verfügung stehenden Kontextinformation (Diskurskontext, Benutzermodell, Grounding, usw.) wird die Information angereichert bzw. reduziert.

In Analogie zu den Schritten bei der Verarbeitung der Eingabe plant die **multimodale Fission** die Verteilung der Inhalte auf verschiedene Modalitäten. Dieser Schritt wird auch **Medienallokation** genannt.¹¹ Dabei spielen zum einen die Art der Inhalte eine Rolle, z. B. lassen sich große Informationsmengen leichter auf dem Bildschirm präsentieren als mit gesprochener Sprache. Auch die modale Diskursgeschichte kann berücksichtigt werden, etwa um gesprochene Fragen auch per Sprache zu beantworten. Allerdings gibt es auch explizite Modalitätswünsche, beispielsweise „Zeige mir alle Alben von Elvis.“. Unter Umständen kann auch erst in diesem Schritt die Entscheidung fallen, dass die zu präsentierende Informationsmenge für die zur Verfügung stehenden Modalitäten zu groß ist und entsprechende Reduktionsprozesse angestoßen werden müssen.

Andererseits muss die Systemausgabe im geeigneten Umfang erkennen lassen, welche Teile der Eingabe verstanden worden sind und dies durch Wiederholung, also eine Augmentierung der Ausgabe verdeutlichen. Diesen Prozess nennt man **Grounding**.

Darüber hinaus muss eine adäquate Redundanz zwischen den Modalitäten geplant werden, um die Systemausgabe eindeutiger und natürlicher zu machen. Dafür müssen auch entsprechende multimodale Referenzen generiert werden.

Modalitätenspezifische Ausgabemodule steuern dann im letzten Schritt die einzelnen Ausgabekanäle an. Bei der Sprachausgabe ist dies zuerst die **Sprachgenerierung**, siehe auch Unterkapitel 3.8, und schließlich die **Sprachsynthese**. Ein allgemeines Phänomen bei dialogischer Interaktion ist das **Alignment**, also das Anpassen des Verhaltens an den Gesprächspartner. Linguistisches Alignment kann sich in der Sprachgenerierung in Wortwahl, Syntaxkonstruktionen oder Förmlichkeit (Duzen oder Siezen) widerspiegeln. Moderne Sprachsynthesen können durch zusätzliche Informationen in ihrer Prosodie stark kontrolliert werden, um kontrastive, emotionale und affektive Bedeutungen zu vermitteln (Burnett et al. 2004).

¹¹Passender wäre allerdings Modalitätenallokation.

Bei der abschließenden Ausgabe müssen die parallel erzeugten Ausgaben der anderen Modalitäten exakt synchronisiert werden. Ein prominentes Beispiel dafür ist die Synchronisation der Lippenbewegungen eines animierten Agenten mit der synthetisierten Sprache.

5.5.4 Modellierung und Repräsentation

Neben den Ablaufmodellen aus Abschnitt 5.5.2 ist insbesondere die Modellierung und Repräsentation des Inhalts der Interaktion von Bedeutung. Dies beginnt bei der Repräsentation der Ergebnisse des Sprachverständnisses und deckt alle Kontextmodelle, das Dialogmanagement und schließlich die Ausgabeplanung ab. Multimodale Dialogsysteme verwenden typischerweise symbolische, formale Beschreibungen; in heutigen Systemen meist auf der Basis von **Ontologien** (s. hierzu Unterkapitel 4.6).

Solche Repräsentationsformalismen unterstützen die Formulierung von Generalisierungen und die Komposition von Modellen durch multiple Vererbung. Beispielsweise kann die Repräsentation eines Liedes im MP3-Player zusammengesetzt werden aus Komponenten wie „Medienobjekt“ (hier wird das Abspielen repräsentiert), „Browserobjekt“ (kann als ein Element von vielen im Browser dargestellt werden), „Suchobjekt“ (eines der Benutzerziele ist das Finden solcher Objekte). Zu jeder Komponente können allgemeine Regeln in den Modulen des Interaktionsmanagements formuliert werden. Damit wird insbesondere die Erweiterbarkeit des Dialogsystems unterstützt.

Zur Verarbeitung der Informationen, die in den verschiedenen Modellen gespeichert sind, kommen verschiedene Mechanismen zum Einsatz. Neben Inferenzsystemen und regelbasierten Systemen können dies auch numerische oder algorithmische Verfahren sein. Eine wichtige Operation bei der Verarbeitung von Diskursinformationen ist z. B. **Overlay** (Alexandersson et al. 2006), eine Erweiterung der Unifikation, siehe dazu Unterkapitel 2.3. Damit lässt sich etwa die aktuelle Benutzerintention berechnen, indem vereinfacht gesagt die neue Benutzereingabe mit kompatibler Information aus der Diskursgeschichte (also den vorhergehenden Eingaben) angereichert wird, während nicht mehr passende Information wegfällt. Reine Unifikation führt in solchen Situationen zu keinem Ergebnis, während Overlay stets ein Ergebnis liefert und dabei die Priorität der aktuellen Eingabe berücksichtigt.

Dabei können einige wichtige Klassen von Modellen unterschieden werden:

- Das Aufgabenmodell enthält die Aufgaben, die mit dem System gelöst werden können. Diese korrespondieren nicht immer direkt zur Funktionalität der Applikation, wie sie durch das API beschrieben wird, sondern orientieren sich am kognitiven Modell, dass sich der Benutzer von der Applikation macht.
- Die Modellierung der Interaktion, also Dialogakte, Sprechakte, Problemlösungsverhalten.
- Das Situationsmodell: Dies sind alle Einflüsse aus der Umwelt, die für den Dialog wichtig sind. Ein Beispiel hierfür ist die Aufmerksamkeit des

Autofahrers, die in Gefahrensituationen nicht durch das Dialogsystem von der Straße abgelenkt werden darf.

- Das Modell der Diskursgeschichte repräsentiert neben den Inhalten auch Informationen über die verwendeten Modalitäten.
- Das Benutzermodell enthält statische und dynamische Präferenzen des Benutzers.

5.5.5 Literaturhinweise

Einen weiteren Überblick über Sprachdialogsysteme und (multimodale) Dialogsysteme liefert Carstensen (2009b). Weiterführende Informationen finden sich in den Publikationen zu einigen großen Forschungsprojekten wie SmartKom, TALK, Match oder WITAS. Wichtige Konferenzen für Sprachdialogsysteme sind Eurospeech, Interspeech, ACL (mit SIGDIAL) und ICMI(-MLMI).

Eine umfassende Darstellung aller Aspekte multimodaler Dialogsysteme bietet Wahlster (2006). Darin werden auch alle hier eingeführten Konzepte aufgegriffen und vertieft. Eine gute Einführung in natürlichsprachliche Dialogsysteme ist McTear (2004).

5.6 Angewandte natürlichsprachliche Generierungs- und Auskunftsysteme

John Bateman

Die meisten der heute in Betrieb befindlichen Computerprogramme kommunizieren mit ihren Benutzern in natürlicher Sprache. Wenn etwas schief geht, erstellen sie eine Fehlermeldung wie z. B. „Parenthesis error in line 345“ oder „Firefox ist momentan im Offline-Modus und kann nicht zum Browsen im Web verwendet werden“. Ein derartiger Gebrauch von natürlicher Sprache ist im Allgemeinen nicht flexibel. Die generierte Sprache ist direkt mit einem programminternen Zustand verknüpft und kann dargestellt werden als:

(5.1) `(5.1) Wenn <Bedingung>
dann drucke „Parenthesis error in line <n>“;`

Die Sprachkomponente in dieser Anweisung besitzt keine eigene interne Struktur; sie ist bloß eine Kette von Buchstaben, die gedruckt werden soll. Das Programm hat keine Kenntnis davon, dass diese Buchstabenkette bestimmten linguistischen Regeln folgen muss, um ein wohlgeformter sprachlicher Ausdruck zu sein. Wenn es z. B. mehrere Klammerfehler in Programmzeile $< n >$ gäbe, wüsste das Programm nicht, dass der zu ändernde Teil der Kette gerade die Subkette „error“ wäre, weil jetzt der Plural „errors“ ausgedruckt werden müsste.

Solange der herzustellende Text gleich bleibt, braucht das Programm nicht mehr darüber zu wissen. Heute gibt es jedoch eine wachsende Anzahl von Anwendungen, wo die Generierung von flexibleren Texten oder sogar von ganz neuen Texten auf der Basis von immer komplexeren programminternen Zuständen wünschenswert ist. Betrachten wir einmal Daten, die von meteorologischen Mess-Stationen gesammelt worden sind – oft in rein numerischer Form. Diese rohen Daten bilden die Basis für eine Fülle von verschiedenen Texten, unter anderem Wettervorhersageberichte. Obwohl es durchaus möglich ist, einen Wetterbericht durch eine Sequenz von oben geschilderten Druckanweisungen zu erstellen, wird dieses Verfahren nie in der Lage sein, einen fließenden Wetterbericht zu produzieren. Der Grund dafür ist, dass Wettervorhersagen keine einfache Aneinanderreihung von miteinander nicht verknüpften Aussagen über numerische Werte, sondern natürlichsprachliche Texte sind – und als solche verwenden sie verschiedene Mittel und Methoden, die einzelne Aussagen zu einem kohärenten Ganzen verbinden. Verschiedene numerische Daten werden zu Sätzen oder Satzgruppen zusammengestellt, referentielle Identität wird durch Pronominalisierung oder definite Nominalphrasen markiert, zeitliche Beziehungen werden durch passende Tempusformen ausgedrückt, und grundlegende grammatische und lexikalische Muster müssen eingehalten werden.

Natürlichsprachliche Generierung (engl. *Natural Language Generation*: NLG) ist das Gebiet, das solche linguistischen Merkmale herausarbeitet und zur automatischen Herstellung von natürlichsprachlichen Texten einsetzt. NLG

ist eine eigenständige Disziplin innerhalb der Computerlinguistik und hat, wie aus Unterkapitel 5.6 ersichtlich, ihre eigenen Methoden und Ziele, die sich häufig von den Methoden und Zielen der bekannteren Gebiete der natürlichsprachlichen Analyse oder des Textverständnisses unterscheiden. In dem vorliegenden Unterkapitel werden praxisnahe Anwendungen von NLG behandelt, sogenannte ‚angewandte natürlichsprachliche Generierung‘. Es wird definiert, welche Systeme unter dem Begriff ‚angewandtes NLG-System‘ zu verstehen sind, und erläutert, wie sich deren Problemstellungen von denen allgemeiner NLG-Systeme unterscheiden. Beispiele für aktuelle Anwendungsbereiche werden auch gegeben. Der Artikel zeigt die wichtigsten heute für angewandte Generierung verwendeten Methoden und schließt mit einer kurzen Darstellung von Schwachstellen und offenen Fragen, die eine breitere Anwendung der NLG-Technologie zur Zeit noch verhindern.

5.6.1 Was ist angewandte NLG?

Ein NLG-System ist ein Computerprogramm, dessen Eingabe ein oder mehrere kommunikative Ziele sowie ein bestimmter Wissensbereich sind und dessen Ausgabe ein natürlichsprachlicher Text ist, der das kommunikative Ziel erfüllt. Alle solchen Systeme – ob angewandt oder nicht – müssen sich mit denselben Aufgaben befassen: Information, die ausgedrückt werden soll, muss ausgewählt werden (Inhaltsfestlegung); eine Textstruktur muss festgelegt werden (Inhaltsorganisation), die dem Zweck des Texts gerecht wird (Textplanung und Linearisierung); der Inhalt der einzelnen Sätze muss determiniert werden (Satzplanung); grammatische und lexikalische Entscheidungen innerhalb dieser Sätze müssen getroffen werden (s. Unterkapitel 5.6).

Obwohl alle diese Aufgaben zu jeder Textherstellung gehören, unterscheiden sich konkrete NLG-Systeme beträchtlich voneinander, inwieweit sie für diese Aufgaben allgemeingültige, wiederverwendbare Lösungen anbieten. Nicht alle Kontexte, in denen ein NLG-System verwendet wird, erfordern volle Flexibilität. Wenn der Anwendungszweck bekannt ist, dann können entsprechende Vereinfachungen im Systemdesign vorgenommen werden. Es ist möglich, dass breit-abdeckende grammatische oder lexikalische Ressourcen (s. Unterkapitel 4.3) nicht notwendig sind, oder dass das Benutzerprofil (s. Abschnitt 3.7.4) von vornherein festliegt. So ein NLG-System kann dann oft nur Texte eines bestimmten Typs generieren (z. B. Wetterberichte, Börsenberichte, Wirtschaftskorrespondenz usw.). Betrachtet man die eingesetzten Generierungsmethoden, sind ähnliche Vereinfachungen zu sehen: eine formale Garantie z. B., dass *alle* Oberflächenformen für einen gegebenen semantischen Inhalt gefunden werden, ist zwar von theoretischem Interesse, aber weniger relevant in der Anwendung.

Für den Zweck dieses Artikels werden **angewandte NLG-Systeme** definiert als Systeme, die für eine bestimmte extern gegebene Anwendungssituation gebaut wurden. Solche Systeme versuchen, ein Informationspräsentationsbedürfnis zu erfüllen, das unabhängig vom Generierungssystem bereits in der Gesellschaft existiert oder existieren könnte.

5.6.2 Beispiele für angewandte NLG-Systeme

In diesem Abschnitt werden einige aktuelle oder signifikante angewandte NLG-Systeme beschrieben mit dem Ziel, drei Aspekte zu verdeutlichen: Erstens, welche Bereiche sind reif für die Anwendung von NLG-Technologie, zweitens welche Motivierung für ihren Einsatz gibt es, und drittens welche Rahmenbedingungen können zu einem erfolgreichen Einsatz führen? Zwei Bereiche von potenziellen und schon verwirklichten Anwendungen sind in anderen Abschnitten dieses Buches beschrieben: gesprochene Auskunftssysteme (s. Unterkapitel 5.5) und maschinelle Übersetzung (s. Unterkapitel 5.7); ein weiterer ist die automatische Textzusammenfassung. Der Platz reicht hier nicht aus, um die einzelnen Systeme mit Literaturnachweis zu zitieren; interessierte Leser seien auf die im Web veröffentlichte und regelmäßig aktualisierte Liste von NLG-Systemen unter <http://purl.org/net/nlg-list> verwiesen. Alle in diesem Artikel durch Namen identifizierten Systeme sind mit ausführlichen Beschreibungen und weiteren Hinweisen in dieser Liste zu finden.

Der etablierteste und bekannteste Anwendungsbereich für NLG-Systeme sind **Wettervorhersagen**. Wettervorhersagen sind aus drei Gründen als Anwendung besonders geeignet. Erstens kann die Sprache, die zu generieren ist, klar und deutlich definiert werden und bildet einen abgeschlossenen Rahmen von Möglichkeiten. Zweitens können die rohen Eingabedaten von den meisten potenziellen Benutzern nicht verstanden werden. Mit der Lieferung einer natürlichsprachlichen Einbettung der Daten steigert sich deren Wert und Wiederverwendbarkeit enorm. Drittens gibt es ein vorhersehbares und immer wiederkehrendes Bedürfnis für die Generierung von neuen Berichten. Besonders der letzte Grund gewährleistet, dass sich die ursprüngliche Entwicklungsinvestition rentiert. Zu Systemen dieser Art gehören das von der der NLG gewidmeten Firma CoGenTex entwickelte System FOG (Forecast Generator), das ein Jahrzehnt lang täglich Berichte in französischer und englischer Sprache für den Kanadischen Wetterdienst generierte, und mehrere Nachkömmlinge. Eine Reihe von Systemen beschäftigen sich jetzt mit dieser allgemeinen Problematik. Besonders relevant ist die an der Universität Aberdeen erreichte Verallgemeinerung solcher Ansätze, um die Versprachlichung von Reihen von zeitabhängigen Daten (*engl. "Time-Series Data"*) abzudecken, egal von welcher Quelle diese Daten stammen. Systeme dieser Art sind für medizinische, industrielle und anderen Bereichen fertiggestellt (Sripada et al. 2001; Yu et al. 2005).

Ein weiterer großer Bereich, in dem wesentliche Vorteile durch die Anwendung von NLG-Technologie zu erwarten sind, ist das **Gesundheitswesen**. Es ist heute bekannt, dass je individueller Information gestaltet wird, sie umso effektiver beim Leser ankommt. Leider ist eine solche intensive Individualisierung unter derzeitigen ärztlichen Arbeitsbedingungen kaum möglich. Customisierung, bei der Texte genau auf Gruppen oder Individuen zugeschnitten werden, ist aber gerade ein wesentlicher Grund für den Einsatz von NLG (s. Abschnitt 3.7.4). Deswegen ist es naheliegend, dass jetzt Experimente in diesem Bereich stattfinden. Das System PIGLET experimentierte z. B. mit der natürlichsprachlichen und für den Laien verständlichen Wiedergabe von medizinischen Akten. Es gibt

auch mehrere Versuche, automatisch generierte Benutzer-adaptierte Hypertexte für die medizinische Fachkommunikation einzusetzen: GeNet, OPADE, ARI-ANNA. Die Effektivität solcher Anwendungen muss allerdings noch evaluiert werden, um die Auswirkungen und möglichen Vorteile von NLG-Technologie genau zu bestimmen. Das an der Universität Aberdeen entwickelte System STOP, das ‚Mit dem Rauchen aufhören‘-Briefe (*engl.* ‚smoking cessation letters‘) generiert, wurde als erstes NLG-System einer breiten klinischen Evaluierung durch den Britischen ‚Health Service‘ unterzogen. Daraus wurde deutlich, dass es nicht einfach ist vorherzusagen, ob und wie Customisierung einen erwünschten Effekt bei den Lesern hervorrufen kann. Weitere empirische Studien sind hier dringend notwendig.

NLG-Technologie offeriert auch wichtige Teillösungen für die **technische Dokumentation**. Die notorischen Probleme bei der Erstellung von technischer Dokumentation sind allgemein bekannt: (a) Gewährleistung von terminologischer Konsistenz, insbesondere wenn Übersetzungen von größeren Textmengen notwendig sind, (b) Aktualisierung der Dokumentationen. Diese Probleme werden besonders relevant bei Dokumentationen für komplizierte Objekte wie Flugzeuge, Rechner, usw. (die in extremen Fällen mehrere Tonnen Papier in Anspruch nehmen können). Wenn die Dokumentation automatisch aus einer aktuellen Datenspezifikation erzeugt wird, dann wird sie zwangsläufig terminologisch konsistent und aktuell sein. Einige wesentliche Versuche, NLG diesbezüglich zu verwenden, sind: IDAS, Corect, und Ghostwriter. Ein weiteres Anwendungsgebiet findet sich dort, wo Information dynamisch aus Simulationen gewonnen wird. Hier können die Texte nicht alle im Voraus geschrieben werden, weil nicht genau bekannt ist, welche Situationen vorkommen werden. Ein System dieser Art ist das von der Columbia Universität in New York und Bellcore entwickelte System PlanDoc. PlanDoc berichtet über Auswirkungen von Änderungen in einem Telefonnetzwerk und wurde von Southwestern Bell verwendet.

Eine weitere Gruppe von Systemen versucht die Schnittstelle zwischen Mensch und **Datenbanken** oder anderen Informationsystemen durch die Verwendung von natürlicher Sprache für die Systemausgabe oder -eingabe zu vereinfachen. Einige Produkte setzen bereits natürliche Sprache ein, um dem Benutzer seine Modellierungentscheidungen klarzumachen (z. B., das KISS-‘Verbalizer‘-Produkt). Es scheint der Fall zu sein, dass es für viele Benutzer einfacher ist, einen normalen Text zu überprüfen, als das ursprüngliche Datenmodell. NLG kann sogar eine Rolle bei der *Eingabe* von Datenbankabfragen spielen. Dafür könnte die an der Universität Brighton entwickelte WYSIWYM-Technologie (*engl.* ‚What you see is what you meant‘) von Bedeutung sein (Power et al. 1998). Bei diesem Verfahren editiert ein Benutzer einen sogenannten ‚Feedback Text‘, der direkt mit einer Inhaltsspezifikation verknüpft ist. Nach jeder Änderung wird der Feedback-Text neu generiert und der Benutzer kann weitere Präzisierungen oder Änderungen vornehmen. Der Vorteil für den Benutzer liegt darin, dass stets natürliche Texte zu sehen sind und weder programminterne Darstellungen von Inhalten noch formale Abfragesprachen wie SQL gebraucht werden. Das WYSIWYM Konzept ist bereits exemplarisch in Anwendungssystemen wie dem CLIME-System (Evans et al. 2006) eingesetzt worden.

Je strukturierter und reichhaltiger die Wissensquellen werden, desto mehr Möglichkeiten für sinnvolle Anwendungen von NLG ergeben sich. Daher besteht ein weiterer Anwendungsbereich in der Entwicklung von Online-Diensten wie Enzyklopädien und virtuellen Museen. Solche Systeme können NLG-Technologien benutzen, um die Interaktion zwischen Mensch und System ganz individuell und dynamisch zu gestalten. Um den potenziellen Benutzerkreis zu maximieren, sind Systeme dieser Art normalerweise Web-basiert. Ein frühes Beispiel hierfür ist das an der Universität Edinburgh entwickelte ILEX-System, das ein virtuelles Museum zum Thema Schmuck anbietet. Der Ansatz ist später ausgebaut worden, um Museen im Allgemeinen abzudecken, und ist gut durch das aktuelle System M-PIRO (*engl. „multilingual personalised information objects“*) repräsentiert (Androulidakis et al. 2007).

Die neueste Entwicklung, die voll auf der Verfügbarkeit von gut strukturiertem Wissen aufbaut, ist in letzter Zeit durch die ‚Semantic Web‘-Initiative (Berners-Lee et al. 2001) mächtig vorangetrieben worden. Trotz kritischer Anmerkungen und Zweifel an der Realisierbarkeit dieser Vision steht heute eine rasant wachsende Anzahl von strukturierten Wissenbasen zur Verfügung und dies bietet eine hervorragende Gelegenheit, nicht-triviale Methoden aus der NLG-Technologie in die Praxis einzusetzen. Daher gibt es bereits einige Systeme, die ‚Semantic Web‘-taugliche Information, vor allem in der Web-Ontologie-Sprache OWL dargestellt, in natürlicher Sprache automatisch wiedergeben können (Galanis und Androulidakis 2007). Diese Anwendung wird sicherlich weiterhin von wachsender Relevanz sein. Sobald Information in standardisierter ontologischer Form (s. Unterkapitel 4.6) vorhanden ist, bietet es sich an, entsprechende NLG-Systeme daran zu setzen, anstatt aufgabenspezifische Schnittstellen zu bauen.

Bereiche, wo der Sprung zu kommerziellen Anwendungen des Semantic Web immer näher rückt, sind zur Zeit vielleicht am deutlichsten durch geographische Informationssysteme repräsentiert. Aus der Forschung zu räumlicher Sprache ist heute bekannt, dass räumliche Beschreibungen besonders aufgaben-, zuhörer- und kontextabhängig sind. Daher sind hier angemessene NLG-Methoden unabdingbar. Systeme werden jetzt für statische Information (Porzel et al. 2002; Turner et al. 2007) sowie für dynamische Information wie Wegbeschreibungen bei Navigationssystemen (Lizogat 2000; Cziferszky und Winter 2002, Dale et al. 2005) entwickelt. Dariüberhinaus wird sich das Wiederverwendbarkeitspotenzial besonders von Ontologie-basierten allgemeinen Generierungssystemen wie KPM (Penman) im Zusammenhang mit dem Semantic Web noch mehr erhöhen.

5.6.3 Mechanismen und Methoden

In diesem Abschnitt wird auf zwei Dimensionen eingegangen, hinsichtlich derer NLG-Systeme unterschieden werden:

- Generierungstiefe – der Unterschied zwischen flachen und tiefen NLG-Systemen, und
- Wiederverwendbarkeit – der Unterschied zwischen anwendungsspezifischen und allgemeingültigen, wiederverwendbaren NLG-Systemen.

Beide Unterscheidungen werden oft mit dem Unterschied zwischen angewandter und nicht-angewandter NLG gleichgesetzt. In Abschnitt ‚Wiederverwendbarkeit‘ wird gezeigt, warum dies heutzutage nicht mehr aufrechterhalten werden kann.

Generierungsverfahren: Flach–Tief

Traditionell (siehe Reiter 1995) werden zwei grundsätzliche methodische Ansätze zum Bau von NLG-Systemen unterschieden: Systeme, die fertige Textteile (*engl. „canned text“* oder *„templates“*) verwenden und Systeme, die ihre Texte auf der Basis tieferen grammatischen, lexikalischen, semantischen und rhetorischen Wissens erzeugt haben. Da diese Trennlinie heute nicht mehr so klar gezogen werden kann, hat Busemann eine passendere Charakterisierung des Unterschieds vorgeschlagen (Busemann 2000). Er unterscheidet zwischen ‚flachen‘ und ‚tiefen‘ Ansätzen. Tiefe Generierung verwendet wissensbasierte Text- und Satzplanungsverfahren. Flache Generierung ist die logische Fortsetzung von Template-basierten Ansätzen.

Heute gibt es einige sehr flexible und komplexe Template-Mechanismen, die Generierungsfunktionalitäten unterstützen, die früher nur in Systemen mit tiefen Generierungsverfahren zu finden waren. Zwei solche Systeme, die in angewandten Generierungssystemen verwendet werden, sind Busemanns TG/2 und das System EXEMPLARS. TG/2 basiert auf einem Produktionsregelansatz–d.h., wenn die Bedingung einer Regel erfüllt ist, wird die Regel ausgeführt (Busemann 2000). In EXEMPLARS sind die Templates in einer Vererbungshierarchie angeordnet. Im Generierungsprozess wird jeweils das spezifischste passende Template ausgewählt. Das Klassifikationsverfahren ist als eine Menge von objektorientierten Java-Methoden implementiert. Nachdem die notwendigen Templates ausgewählt worden sind, wird ein Java-Programm für den dynamischen Aufbau von HTML-Webseiten erstellt. TG/2 wird als Technologie immer noch in mehreren Systemen verwendet (Busemann 2005) und EXEMPLARS wird u.a. in den Anwendungssystemen ProjectReporter und CogentHelp verwendet.

Obwohl sie wenig mit den früheren Template-Ansätzen gemein haben, besitzen beide Systeme einige in tieferer Generierung nicht übliche Einschränkungen. Laut Busemann ist, z. B., der TG/2-Ansatz nicht zu empfehlen, wenn die Generierungsaufgabe komplexe Satzplanung, subtile Lexemauswahl oder flexible Verteilung von semantischen Informationen auf grammatische Einheiten erfordert. Außerdem können Template-Ansätze ein teures Redesign verlangen, wenn sie an eine neue Anwendung adaptiert werden sollen.

Wiederverwendbarkeit

Ein anderes methodologisches Unterscheidungsmerkmal ist der Aspekt der Wiederverwendbarkeit. Die anwendungsorientierten Systeme werden in einer bottom-up Vorgehensweise entwickelt. Das heißt, jede Anwendung ist eine neue Situation, deren Sprachgebrauch untersucht wird, und die linguistischen Ressourcen (Grammatik, Lexikon) werden – typischerweise – neu geschrieben.

Beim Bau der großen Generierungssysteme der 80er und 90er Jahre ist man ganz anders vorgegangen. Diese Projekte konzentrierten sich darauf, große und

allgemeingültige linguistische Ressourcen aufzubauen, um eine beliebige Wiederverwendbarkeit des Systems in den verschiedensten praktischen Anwendungsszenarien zu gewährleisten. Wiederverwendbarkeit wird auch in Reiter und Dale (2000) als ein zentrales Anliegen für angewandte NLG-Systeme dargestellt. Reiter und Dale zitieren drei Beispiele von wiederverwendbaren, frei verfügbaren Generierungssystemen: KPML (Penman), FUF/SURGE und RealPro. Unlängst ist ein weiteres System für syntaktische Realisierung (s. Unterkapitel 5.6) von der Universität Aberdeen entwickelt worden: SimpleNLG. Dieses System beinhaltet zwar keine großen Grammatiken aber dafür ist für kleinere Anwendungen besonders leicht erweiterbar.

Heutige angewandte Generierungssysteme haben jedoch die Methodologie von allgemeingültigen linguistischen Ressourcen nicht weiter verfolgt. Dafür gibt es zwei Gründe. Erstens, eine solche wiederverwendbare Grammatik ist eine große komplexe Angelegenheit, die in einer einfachen Anwendung schnell zum vermutlich langsamsten und platzaufwändigsten Modul werden würde. Und zweitens erfordert die Anpassung eines allgemeinen Generierungssystems an eine neue Anwendung oder auch nur eine kleine Änderung in der Anwendung einen Experten. Deshalb ist es für Anwender am Anfang oft günstiger, ein System zu benutzen, das sie selbst warten und anpassen können, wie ein Template-basiertes System. Wenn neue linguistische Konstruktionen notwendig werden, kann hier der Nutzer die notwendigen neuen Templates definieren.

Diese beiden Probleme habe dazu geführt, dass viele Systementwickler unterscheiden zwischen praktischen, angewandten NLG-Systemen einerseits, die klein und zweckspezifisch sind, und theoretischen Forschungssystemen andererseits, die groß und schlecht handhabbar sind und für Anwendungen nicht in Frage kommen. Ob man diese Unterscheidung aufrechterhalten kann, ist allerdings fraglich. Die großen NLG-Systeme bieten heute auch Template-Funktionalität an und bewegen sich auf eine effektive Implementierung zu, während Template-basierte Systeme zwangsläufig wiederverwendbare Template-Bibliotheken (z. B. Busemanns TG/2) entwickeln. Weitere Schritte zur Wiederverwendbarkeit sind auch durch die Verbindung von Generierungssystemen mit vorhandenen lexikalischen Wissensquellen wie Wortnetzen (s. Unterkapitel 4.3) unternommen worden.

Eine vielversprechende Möglichkeit, die Wiederverwendbarkeit von Systemen zu erhöhen, besteht darin, Methoden zu entwerfen, die in der Lage sind, Systeme für die Bedürfnisse von spezifischen Anwendungen automatisch zuzuschneiden. Wenn, z. B., die für eine Anwendung notwendige **Subsprache** erkannt werden könnte und ein System für diese Subsprache automatisch vorbereitet wird, wäre dies eine große Erleichterung für die Systemherstellung. Ein früher Versuch in dieser Richtung wurde in Henschel und Bateman (1997) vorgestellt, wobei kleine anwendungsbezogene Grammatiken aus allgemeinen Grammatiken automatisch erzeugt worden sind. In den letzten Jahren ist dieses Problem häufiger mit probabilistischen Verfahren angegangen worden. Das pCRU-System von Belz (2007) betrachtet z. B. das ganze Generierungsverfahren als eine Menge von gleichberechtigten Regeln. Um einen konkreten Ausgabestil zu erzeugen, werden diese Regeln mit Wahrscheinlichkeiten versehen, die durch die statistische Auswer-

tung eines Zielkorpus gewonnen worden sind. In diesem System entfallen auch einige der inhärenten Probleme der ‚Standardarchitektur‘ für Generierung, wobei Generierungsphasen in einer eher zu strikten Reihenfolge aufeinander folgen (s. Unterkapitel 3.8). Noch zu untersuchen ist jedoch, inwiefern und unter welchen Bedingungen die Ausgabequalität solcher automatisch erzeugten Systeme adäquat bleibt.

5.6.4 Perspektiven

Die wichtigste Dynamik, die man zur Zeit auf dem Gebiet der angewandten NLG beobachten kann, ist von zwei gegensätzlichen Strömungen gekennzeichnet: Einerseits werden die sprachlichen Fähigkeiten von Marktprodukten immer komplexer (obwohl noch unter Verwendung eher simpler Methoden), während andererseits die von der NLG-Forschung anvisierten Anwendungen sich immer mehr an wirtschaftlich relevanten Aufgaben orientieren. Es versteht sich von selbst, dass natürlichsprachlicher Output einen Marktvorteil für Produkte in den verschiedensten Gebieten darstellen könnte. Die sprachlichen Fähigkeiten von Marktprodukten bringen aber als Nebeneffekt auch eine steigende Erwartungshaltung der Nutzer mit sich. Die Zeit ist deshalb reif für eine umfassende Interaktion zwischen NLG-Forschung und wirtschaftlichen Anwendungen. Je mehr die einfachen Methoden der Textherstellung an ihre Grenzen kommen, umso mehr hat NLG-Forschung ökonomische Lösungen anzubieten. Angewandte NLG ist deshalb das zur Zeit am schnellsten wachsende Gebiet innerhalb der NLG.

Obwohl NLG vor dem Durchbruch zur wirtschaftlichen Anwendung steht, gibt es immer noch einige ernsthafte Hindernisse. Es ist zu oft noch der Fall, dass der Aufbau eines Generierungssystems von Null (bottom-up-Methodologie) weniger Aufwand mit sich zu bringen scheint als die Zuschneidung eines allgemeinen Systems auf eine spezifische Aufgabe (Wiederverwendbarkeits-Methodologie). Ohne klare Evaluierungsstrategien sind in der Tat wenige Entscheidungskriterien dafür vorhanden, welche Methoden angewendet werden sollen. Ein weiteres, genauso relevantes Problem ist, dass die Möglichkeiten der NLG-Technologie auf Industrieseite nicht genügend bekannt sind. Praktische Produkte, die bereits NLG-Synthese enthalten, verwenden relativ simple Methoden und haben den Kontakt mit der NLG-Forschung nicht erforderlich gemacht. Der Entwickler von marktreifen Anwendungen kommt mit den Grenzen simplerer Generierungsmethoden nur indirekt über steigende Entwicklungs- und Wartungskosten in Berührung, die entstehen, wenn die Komplexität der zu generierenden Sprache zunimmt. Das Bild von NLG als etwas einfach Machbarem wird leider unterstützt durch die voreilige Entgegensetzung von kleinen, praktisch anwendbaren Systemen auf der einen Seite und großen experimentellen zur Anwendung nicht geeigneten auf der anderen Seite.

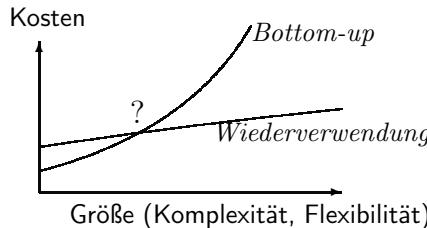


Abbildung 5.4: Generierungsmethoden: Kostenvergleich

Es ist daher dringend nötig, genauere Evaluationsvergleiche anzustellen. Mit steigender natürlichsprachlicher Funktionalität werden irgendwann die Kosten für Systeme, die von Anfang an neu gebaut werden, die Kosten für Systeme mit großer Abdeckung übersteigen. Dies ist in Abbildung 5.4 grafisch veranschaulicht. Jedoch ist weder bekannt, wie komplex und reichhaltig die Sprache, und vor allem, die *Sprachvariation*, werden muss, um den Schnittpunkt der Kurven zu erreichen, noch welchen Anstieg die beiden Größe-Kosten-Kurven haben. Diesen Fragen muss noch wesentlich mehr Aufmerksamkeit geschenkt werden, damit angewandte NLG aus ihren Kinderschuhen wachsen kann.

Dass dies passieren wird, ist allerdings klar. Mit einer zunehmenden Menge an elektronisch gespeicherter Information bekommt die Verfügbarkeit von effizienten Werkzeugen zur Darstellung dieser Information für potenzielle Interessenten heute enorme ökonomische Bedeutung. Die ökonomische Relevanz wird noch erhöht durch den wertsteigernden Effekt einer Mehrfachnutzung derselben Daten durch verschiedene Zielgruppen unter unterschiedlichen Gesichtspunkten: Gesundheitswesen: Arzt, Pflegepersonal, Patient; Fernunterricht: Anfänger, Fortgeschritten, Kursleiter; Softwarenutzer: Anfänger, Experte; und vielen anderen. Dies wird darüberhinaus durch die steigenden Anforderungen von Multilingualität und sogenanntem ‚Cross-Media Publishing‘ mehrfach multipliziert. Und schließlich sehen wir eine immer enger werdende Verschmelzung von traditioneller NLG-Technologie, Methoden aus Dialogsystemen (s. Unterkapitel 5.5), multilingualen Ansätzen (siehe Bateman et al. 2005) und multimodaler Präsentation (van der Sluis et al. 2007). Hier sind die gebündelten Chancen für innovative Produkte und nützliche Anwendungen einfach zu hoch, um noch lange unbedient zu bleiben.

5.6.5 Literaturhinweise

Für detailliertere Einführungen in NLG und ihre Methoden, Probleme und Architekturen sei auf Unterkapitel 3.8, McDonald (2000), Bateman und Zock (2003), und van der Linden (2000) verwiesen. Außerdem ist Reiter und Dale (2000) von besonderer Relevanz für angewandte Systeme.

5.7 Maschinelle und computergestützte Übersetzung

Susanne Jekat und Martin Volk

5.7.1 Einleitung

Die technische Entwicklung zieht eine große Menge von (meist) digitalisierten Texten nach sich (Technische Dokumentationen, Handbücher, Gebrauchsanweisungen, Werbematerial, Websites etc.). Gleichzeitig bedingt die fortschreitende Globalisierung, dass die genannten Texte möglichst zeitnah in verschiedenen Sprachen vorliegen müssen, um die dazugehörigen Produkte zu verkaufen. Daneben ergibt sich im Bereich Wissensmanagement und Recherche aufgrund der verbesserten Aktualität ein deutlicher Richtungswechsel weg vom Papiermedium hin zur digitalen Quelle. Für die beiden genannten und weitere Bereiche spielen die Übersetzung und – aufgrund der großen Textmengen und des Zeitdrucks insbesondere die maschinelle Übersetzung (MÜ) und die computergestützte Übersetzung – zunehmend wichtige Rollen. Im folgenden Abschnitt werden die Entwicklung und der aktuelle Stand der maschinellen und computergestützten Übersetzung dargestellt.

Unter einer Übersetzung wird üblicherweise die schriftliche Übertragung eines Textes, der in einer **Quellsprache** vorliegt, in einen entsprechenden Text in einer **Zielsprache** verstanden. Mit neuen MÜ-Systemen ist es aber auch möglich, einen gesprochenen Text zunächst zu verschriftlichen und dann zu übersetzen, eine Übersetzung akustisch auszugeben oder maschinell zu dolmetschen (vgl. z. B. das Projekt zum maschinellen Dolmetschen **Verbmobil**, beschrieben in Wahlster 2000). Neben der Bezeichnung MÜ ergibt sich für den gesamten Bereich auch die Bezeichnung **Machine Translation**, MT. Bei der Übersetzung in die Zielsprache soll möglichst die Bedeutung bzw. Funktion des Quelltextes erhalten bleiben, während besondere sprachliche, kulturelle oder andere, z. B. durch die geographische Situation einer Zielgruppe, bedingte Unterschiede im Text angepasst oder erklärt werden müssen. Dies erfordert ein hohes Maß an Wissen über inhaltliche und kulturelle Zusammenhänge in beiden Sprachräumen und – insbesondere bei literarischen Texten – eine individuelle kreative Leistung, um die Eigenschaften des Quelltextes zielsprachlich adäquat wiedergeben zu können.¹² Wissen, z. B. über technische und rechtliche Zusammenhänge und Kreativität bei deren Darstellung in der Zielsprache sind im Übersetzungsprozess aber auch bei der Anpassung von Technischen Dokumentationen oder Werbetexten notwendig. Seit der Erfindung von Computern wird versucht, diese für Übersetzungsaufgaben einzusetzen. So ist der interdisziplinäre Bereich MÜ entstanden, der Einflüsse von Linguistik, Mathematik, Informatik, Übersetzungswissenschaft

¹²Allerdings erscheint es nicht realistisch, maschinelle Übersetzungssysteme für die Übersetzung literarischer Texte einzusetzen.

und Elektrotechnik und damit zentrale Bereiche der Computerlinguistik in sich vereinigt.

Das ursprüngliche Ziel der MÜ war die vollautomatische, qualitativ hochwertige Übersetzung (= **fully accurate high quality translation (FAHQT)**), d.h. die MÜ sollte möglichst die Übersetzungsleistung eines menschlichen Übersetzers abbilden. Außerdem wurde lange Zeit davon ausgegangen, dass sich Texte 1:1 von einer Sprache in eine andere übertragen lassen und dass Sprache ein berechenbares Phänomen sei. Besonders deutlich wird dies im sogenannten Weaver-Memorandum (Weaver 1955), das historisch als Ausgangspunkt für die MÜ-Forschung und -Kommerzialisierung gesehen werden kann. In Bezug auf FAHQT wurde bereits 1960 von einigen Forschern erkannt, dass dieses Konzept vermutlich ein zu hochgestecktes Ziel für MÜ ist (vgl. Bar-Hillel 1960).

Die große Herausforderung der MÜ liegt darin, dass die Bedeutung und der Kontext eines Textes maschinell erfasst werden müssen, um ihn adäquat zu übersetzen. Hierzu bedarf es einer sehr guten computerlinguistischen Analyse des Ausgangstextes und einer entsprechend hochwertigen Generierung des zielsprachlichen Textes. Ferner wird im Allgemeinen die Modellierung von situativen Beschreibungen und – in Bezug auf die wachsende Menge von Fachtexten – vor allem von entsprechendem Fachwissen benötigt. Dieses **Wissen** (vgl. auch Unterkapitel 4.6) kann dann für Schlussfolgerungen verwendet werden, die über den Inhalt des eigentlichen Textes hinausgehen, z. B. zur Auflösung von lexikalischen und strukturellen Ambiguitäten (Mehrdeutigkeiten), Ellipsen und referenziellen Bezügen (siehe Abschnitt 3.7.2). Die Repräsentation und Verarbeitung nichtsprachlichen Wissens (siehe Unterkapitel 4.6) wirft aber verschiedene, bislang noch nicht gelöste Probleme auf. In Bezug auf die oben genannte FAHQT kann der menschliche Übersetzer in absehbarer Zeit nicht durch MÜ ersetzt werden. Allerdings erzielen aktuelle Systeme unter gewissen Beschränkungen bei der Eingabe (z. B. **Kontrollierte Sprache**) und auf eine fachlich eingegrenzte Domäne sowie bei der Anwendung statistischer Methoden (vgl. Abschnitt 5.7.4) bereits beachtliche Ergebnisse. Wenn der Anspruch auf höchste Übersetzungsqualität (**FAHQT**) aufgegeben wird, kann der Einsatz von MÜ auch gerade dann sinnvoll sein, wenn es um die Klassifizierung großer Mengen fremdsprachiger Texte, eine schnelle Einschätzung eines einzelnen fremdsprachigen Textes oder eine erste Anbahnung der Kommunikation mit anderssprachigen SprecherInnen geht.

Neben MÜ-Systemen, die als vollautomatische Systeme ganz oder fast ohne menschliche Intervention übersetzen sollten, gibt es verschiedene Werkzeuge zur Unterstützung des menschlichen Übersetzers, die in zusammenhängenden Systemen oder als einzelne Komponenten vorliegen. Man spricht dabei von **Computer-gestützter Übersetzung** (*computer-aided translation, CAT*), die in Abschnitt 5.7.6 behandelt wird). Die Grenzen zwischen manueller Übersetzung mit maschineller Unterstützung und maschineller Übersetzung mit menschlicher Unterstützung sind fließend. Unabhängig von der konkreten Realisierung eines MÜ-Systems ist es das gemeinsame Ziel, den Grad der Automatisierung des gesamten Übersetzungsprozesses zu steigern (siehe auch Abschnitt 5.7.7).

Übersetzungsschwierigkeiten können am besten anhand einiger Beispiele aufgezeigt werden. Interessant ist, dass sich selbst bei miteinander verwandten Spra-

chen, wie Deutsch und Englisch, beliebig viele Beispiele finden lassen, die – zumindest was die MÜ anbelangt – Probleme bereiten können.

Zu den bekanntesten Schwierigkeiten gehören hierbei die **lexikalischen Lücken** (*gaps*) oder **Nichtentsprechungen** (*mismatches*). Diese treten immer dann auf, wenn versprachlichte Konzepte nicht direkt von einer Sprache in eine andere transferiert werden können bzw. dort nicht in der gleichen Form existieren. Das englische Wort *wall* kann im Deutschen sowohl (*Innen-*)*Wand* als auch (*Außen-*)*Mauer* bedeuten. Im Falle einer Übersetzung von Englisch nach Deutsch muss daher das zur Unterscheidung notwendige Wissen aus dem Kontext erschlossen werden. Ein weiteres Beispiel ist das englische Wort *chair*, das u.a. mit *Stuhl* oder mit *Sessel* ins Deutsche übersetzt werden kann. Als letztes Beispiel sei *sich verwählen* genannt, das nur in Form der Umschreibung *to dial the wrong number* im Englischen ausdrückbar ist.

Eine andere Klasse von Übersetzungsproblemen bilden die sogenannten **Divergenzen**, die Unterschiede zwischen zwei Sprachen in der syntaktischen Struktur beschreiben. Um den Satz *Das Buch gefällt Eva* ins Englische zu übersetzen, muss eine Argumentvertauschung von Subjekt und Objekt vorgenommen werden: eine entsprechende Übersetzung lautet *Eva likes the book*. Zu den Divergenzen, die am häufigsten in der MÜ-Literatur diskutiert wurden, zählt das **Head Switching**, das eine Kombination von kategorialer und struktureller Divergenz beschreibt. Ein Beispiel hierfür ist die Übersetzung von *Eva geht gern* in *Eva enjoys to walk*. Das Adverb *gern* hat kein englisches Pendant (*likingly* ist ungrammatisch), fordert also die „Kopfvertauschung“ und damit die englische Realisierung mit Kontrollverb, das den ursprünglichen Satz einbettet.

5.7.2 MÜ-Ansätze

Dieser Abschnitt beschreibt die bekanntesten MÜ-Ansätze und stellt deren Vor- bzw. Nachteile heraus.

MÜ-Systeme können zunächst aufgrund der Sprachen und der Domänen (Wirtschaft, Biologie, Recht, usw.) unterschieden werden, die vom System behandelt werden. MÜ-Systeme unterteilen sich weiter in **unidirektionale** und **bidirektionale MÜ-Systeme**, wobei nur letztere eine Übersetzung in beide Richtungen vorsehen.

MÜ-Systeme lassen sich auch durch die Art der Benutzerinteraktion unterscheiden. So gibt es Systeme, bei denen keine Interaktion während der Übersetzung möglich ist (Batch-Systeme), und interaktive MÜ-Systeme. Erstere Systeme sehen Vor- und Nachbearbeitung vor, letztere erlauben beispielsweise die interaktive Auflösung von Ambiguitäten oder eine Auswahl von Übersetzungsalternativen. Ein interaktives System kann Vorschläge machen oder es stellt Fragen an die Benutzer und Benutzerinnen, um fehlendes Weltwissen zu kompensieren. Interaktive MÜ-Systeme stellen ein interessantes theoretisches Konzept dar, konnten sich aber bisher praktisch noch nicht durchsetzen.

Die einfachste Übersetzungsstrategie ist die **direkte Übersetzung**. Diese wurde von den meisten MÜ-Systemen der ersten Generation verfolgt. Ein Quelltext wird dabei nur auf oberflächennahen Beschreibungsebenen morphologisch

analysiert. Nachfolgend wird mit Hilfe eines bilingualen Wörterbuchs direkt in die Zielsprache übersetzt. Dort können sich einfache Operationen, wie die Anpassung der Wortreihenfolge, anschließen, die dann bereits den Zieltext ergeben.

Die Vorteile dieser Strategie liegen in der Geschwindigkeit und den niedrigen Kosten, die Qualität des Ausgabetextes ist jedoch meist ungenügend. Sind Quell- und Zielsprache lexikalisch und strukturell sehr ähnlich, so können manchmal brauchbare Ergebnisse mit diesem Ansatz erzielt werden.

Für bessere Ergebnisse sind aufwändigeren Übersetzungsstrategien nötig. Wir unterscheiden grundsätzlich zwischen **Regel-basierten** und **Daten-basierten** Strategien. Die Regel-basierten Verfahren beruhen auf manuell erstellten zweisprachigen Wörterbüchern sowie lexikalischen und grammatischen Entsprechungsregeln. Diese Verfahren werden weiter unterteilt in **Transfer-basierte** (siehe Abschnitt 5.7.3) und **Interlingua-basierte** Übersetzung (siehe Abschnitt 5.7.3).

Im Gegensatz zu den Regel-basierten Verfahren beruhen Daten-basierte Verfahren auf großen Mengen von Humanübersetzungen. Das heißt, das System erhält als Eingabe eine große Textmenge in der Ausgangssprache und die entsprechenden, manuell übersetzten Texte in der Zielsprache. Man spricht dann von parallelen Korpora oder Bitexten. Das System lernt aus den parallelen Korpora die Wortentsprechungen und Übersetzungsmuster selbstständig.

Daten-basierte Verfahren werden manchmal in Beispiel-basierte und statistische MÜ unterteilt. Bei den Beispiel-basierten Verfahren legt man Wert darauf, dass das System linguistisch plausible Einheiten lernt, während das System bei statistischen Verfahren beliebige statistisch „auffällige“ Wortsequenzen lernt. Obwohl die Beispiel-basierten Verfahren intuitiv plausibler sind, konnten sie sich bisher nicht gegen die statistischen Verfahren durchsetzen. Beispiel-basierte Verfahren sind rechenaufwändiger und ein Qualitätsvorsprung konnte nicht nachgewiesen werden. Deshalb beschränken wir uns in diesem Artikel auf die Darstellung der gegenwärtig dominierenden statistischen MÜ (siehe Abschnitt 5.7.4). Zur Vertiefung der Beispiel-basierten MÜ empfehlen wir Carl und Way (2003) und Way und Gough (2005).

5.7.3 Regel-basierte Systeme

Linguistische Defizite der einfachen direkten MÜ, wie das Fehlen einer syntaktischen und semantischen Analyse, führten zur Entwicklung des Transfer- und des Interlingua-Ansatzes, die in den folgenden beiden Abschnitten beschrieben werden.

Transfer

Der Transferansatz (siehe Abbildung 5.5) besteht prinzipiell aus 3 Verarbeitungsphasen. Zunächst wird die quellsprachliche Eingabe geparst und ansatzweise semantisch analysiert (siehe Unterkapitel 3.5 und 3.6). Hierfür werden häufig unifikationsbasierte Grammatikformalismen wie die **Head-Driven Phrase Structure Grammar**, **HPSG**, oder die **Lexical Functional Grammar**, **LFG** verwendet (vgl. Unterkapitel 2.3).

Anschließend wird die quellsprachliche Repräsentation durch die Anwendung von **Transferregeln** in eine abstrakte, zielsprachliche Repräsentation abgebildet. Eine hier vereinfacht dargestellte Regel wie

(5.2) E-N1 E-N2 wird transferiert zu F-N2 de F-N1

könnte dabei z. B. beschreiben, dass der englische Ausdruck *the installation configuration*, (N1, N2) ins Französische durch *La configuration d'installation*, (N2 de N1) übersetzt wird.

Transferregeln beschreiben Übersetzungsentsprechungen, die in ihrer Verwendung auf sinnvolle Kontexte eingeschränkt werden. Mit diesen Kontextbeschreibungen kann die Qualität der Übersetzungen beeinflusst werden. Schließlich wird in der Generierung aus der vom Transfer erzeugten zielsprachlichen Repräsentation wieder eine natürlichsprachliche Ausgabe erzeugt (siehe Unterkapitel 3.8).



Abbildung 5.5: Transfer-basierte Übersetzung

Ein Nachteil des Transfer-Ansatzes ist, dass bei einer Integration neuer Sprachen ins System oder einer Änderung der Übersetzungsrichtung mehrere neue Komponenten erstellt werden müssen.

Interlingua

Multilinguale Interlingua-Systeme (siehe Abbildung 5.6) verwenden eine sprachenunabhängige Zwischenrepräsentation, die durch eine Analyse der Eingabe erhalten wird und selbst als Eingabe für eine zielsprachliche Textgenerierung dient. Diese Repräsentation sollte neutral bezüglich aller im System verarbeiteten Sprachen sein und erlaubt daher keinen direkten Bezug mehr zwischen Quell- und Zielsprache, wie er noch bei der direkten Übersetzung und beim Transfer vorhanden war. Wiederum vereinfacht dargestellt, könnte der Satz *Guten Tag, Herr Prof. Klabunde* folgendermaßen repräsentiert werden: $\langle \text{Greet} \rangle; \langle \text{Title} \rangle; \langle \text{name:Klabunde} \rangle$.

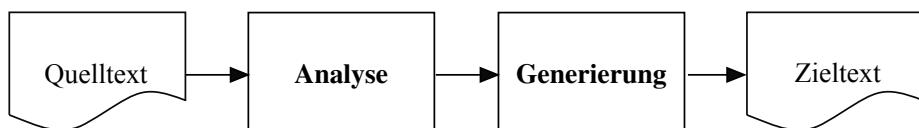


Abbildung 5.6: Interlingua-basierte Übersetzung

Der Vorteil des Transfer-Ansatzes gegenüber dem Interlingua-Ansatz liegt in der einfacheren Analyse- und Generierungsphase bei der Verarbeitung. Beim Transfer muss beispielsweise keine vollständige Bedeutungsanalyse gemacht werden und die zielsprachliche Wortwahl ist einfacher, weil direkt von der Ausgangsauf die Zielsprache zugegriffen wird. Das Hauptargument für einen Interlingua-Ansatz ist – wie oben bereits erwähnt – die geringe Anzahl der zu entwickelnden Systemkomponenten, insbesondere für multilinguale Systeme. Man benötigt nur eine Analyse- und Generierungskomponente für jede im System vorhandene Sprache, nicht jedoch bilinguale Module, wie zweisprachige Wörterbücher für alle Sprachpaare bzw. Übersetzungsrichtungen. Allerdings handelt es sich bei allen bisher entwickelten Systemen um Kompromisslösungen, was die Sprachneutraлизtät der Zwischenrepräsentation anbelangt. In Bezug auf die Integration neuer Sprachen oder Sprachrichtungen ist ein Interlingua-System weniger aufwändig als ein Transfer-System. Die Forderung nach einer umfassenden sprachunabhängigen Zwischenrepräsentation konnte jedoch bisher nicht eingelöst werden (vgl. auch Maegard 1999). Das liegt vielfach auch daran, dass die Bezeichnungen für Konzepte in den einzelnen Sprachen selten deckungsgleich sind, wie das o.a. Beispiel der Übersetzung von *wall* in *Mauer* oder *Wand* bereits verdeutlicht. Domänenabhängige Systeme können sich beispielsweise durch eine detaillierte Domänenmodellierung und eine spezifische Grammatik behelfen.

Es gibt eine Reihe von Regel-basierten MÜ-Systemen, die über die Jahre so optimiert wurden, dass sie sich in bestimmten Marktsegmenten etabliert haben. So stehen Regel-basierte MÜ-Systeme z. B. bei der EU und bei einigen Großbanken allen Mitarbeitern für schnelle Vorübersetzungen zur Verfügung. Bei der EU und bei großen Softwarehäusern werden solche Systeme auch zur Vorbereitung von Humanübersetzungen genutzt. Auch bei vielen Gratis-Übersetzungsdiensten im Internet handelt es sich gegenwärtig noch um Regel-basierte Systeme.

Sowohl Transfer als auch Interlingua sind durch einen hohen Grad an manueller Modellierung gekennzeichnet. Durch die steigende Verfügbarkeit von bi- und multilingualen Textkorpora (siehe Unterkapitel 4.1) ist in den letzten Jahren die **statistische maschinelle Übersetzung, SMÜ** sehr attraktiv geworden.

5.7.4 Statistische Maschinelle Übersetzung

Regel-basierte MÜ-Systeme arbeiten mit manuell erstellten zweisprachigen Wörterbüchern und großen Regelsammlungen. Die Entwicklung eines MÜ-Systems ist somit sehr aufwändig und erinnert in vielen Aspekten an ein komplexes Software-Entwicklungsprojekt. Man versucht, die Komplexität der Grammatikentwicklung durch konzeptuelle Modularisierung und geeignete Werkzeuge in den Griff zu bekommen. Die Abhängigkeiten der linguistischen Regeln und die umfassende Behandlung von sprachlichen Ausnahmen führen jedoch immer wieder zu Problemen bei der Robustheit und Wartbarkeit.

Statistische MÜ-Systeme heben sich insbesondere bezüglich des Entwicklungsaufwands und der Robustheit von Regel-basierten Systemen ab. Bei Vorliegen eines großen parallelen Korpus für das intendierte Sprachpaar kann ein SMÜ-System vollautomatisch erstellt werden. Außerdem sind SMÜ-Systeme robust

und liefern auch für ungrammatische oder unvollständige Eingabesätze Übersetzungsvorschläge.

Ausgangspunkt der SMÜ ist also ein großes Korpus humanübersetzter Texte, wenn möglich aus dem intendierten Anwendungsbereich. Wenn das SMÜ-System Dokumente einer Maschinenbau-Firma übersetzen soll, sollte auch das Eingabekorpus aus diesem Bereich stammen. Die übersetzten Sätze in diesem Eingabekorpus werden bei der SMÜ – bildlich gesprochen – in Wortfolgen zerschnitten und bei der Übersetzung neuer Sätze neu zusammengesetzt. Wie funktioniert dieses Verfahren im Detail?

Gegeben sei beispielsweise eine große Menge von Quartals- und Jahresberichten einer international operierenden Firma, übersetzt von Deutsch nach Englisch. Das SMÜ-System wird auf der Basis dieser Texte automatisch erstellt, indem Entsprechungen und ihre Wahrscheinlichkeiten abgeleitet werden. Diese Texte bezeichnet man deshalb als **Trainingskorpus**. Das Korpus wird in einem ersten Schritt auf Satzebene aligniert. So wird z. B. berechnet, dass der deutsche Satz in (5.3) dem englischen Satz in (5.4) entspricht.¹³

- (5.3) Die Zellstoff- und Papierindustrie sowie der Bausektor waren indessen weiterhin nachfrageschwach.
- (5.4) Demand continued to be weak in the pulp and paper and construction sectors.

Ein **Alignierungsprogramm**¹⁴ berechnet, welcher Satz der Ausgangssprache mit welchem Satz der Zielsprache übersetzt wurde. Zu diesem Zweck vergleicht das Programm die Satzlängen in Ausgangs- und Zielsprache (in Anzahl Zeichen) und nutzt übereinstimmende Wörter (z. B. Eigennamen) und Zahlen als Ankerpunkte. Auf diese Weise können Satzentsprechungen effizient und zuverlässig ermittelt werden. Dies umfasst auch die Möglichkeit, dass ein Satz der Ausgangssprache zwei oder drei Sätzen der Zielsprache entspricht (und umgekehrt). Satzalignierungssysteme sind nicht nur für die SMÜ sondern auch für alle anderen Arbeiten mit parallelen Texten und insbesondere auch für das Auffüllen von Translation Memories von zentraler Bedeutung (vgl. Abschnitt 5.7.6).

In einem zweiten Schritt wird die Alignierung verfeinert, indem das System berechnet, welche Wörter eines Satzes der Ausgangssprache mit welchen Wörtern des entsprechenden Satzes der Zielsprache übersetzt wurden. Die Basis für die Wortalignierung ist die Verteilung der Wörter im Korpus. So kann beispielsweise die Alignierung von *Zellstoff* und *pulp* darauf beruhen, dass *pulp* das einzige Wort ist, das in allen englischen Sätzen vorkommt, die mit deutschen Sätzen aligniert sind, die das Wort *Zellstoff* enthalten.

In unserem Beispieldatensatzpaar können die Entsprechungen *Zellstoff* – *pulp* und *Bausektor* – *construction sectors* recht einfach berechnet werden (wenn wir vom Numerus-Unterschied absehen). Jedoch ist es schwierig, eine Entsprechung für

¹³Das Beispieldatensatzpaar stammt aus einem Quartalsbericht des Maschinenbauers ABB und wird auch im Abschnitt 5.7.6 verwendet.

¹⁴Alignierung und der englische Begriff **Alignment** werden hier in Bezug auf denselben Sachverhalt verwendet.

das englische Wort *demand* zu finden, da die Übersetzung im deutschen Kompositum *nachfrageschwach* enthalten ist. Es ist also denkbar, dass sowohl *demand* als auch *to be weak* mit *nachfrageschwach* aligniert werden. Das deutsche Wort *indessen* hat gar keine Entsprechung im englischen Satz und wird deshalb mit einem leeren Wort aligniert.

Analog zur Satzalignierung muss man auch bei der Wortalignierung die Möglichkeit vorsehen, dass ein Wort in der Ausgangssprache mehreren Wörtern der Zielsprache entspricht (und umgekehrt). Da außerdem die Reihenfolge der Wörter in verschiedenen Sprachen variiert, muss man bei der Wortalignierung (anders als bei der Satzalignierung) überkreuzende Alignierungen zulassen. Dies macht die automatische Wortalignierung schwieriger als die Satzalignierung. Das Ergebnis der Wortalignierung ist also ein automatisch erzeugtes zweisprachiges Wörterbuch (bei dem man eine gewisse Rate an Fehlentsprechungen in Kauf nimmt). Typischerweise handelt es sich um ein Vollformenwörterbuch (d.h. um Entsprechungen von flektierten Wortformen und nicht um Grundformen). Das Alignierungssystem vermerkt zu jedem Wortpaar die Wahrscheinlichkeit der Entsprechung.

Aus den alignierten Wörtern wird in einem dritten Schritt ein aligniertes Wortfolgenlexikon erzeugt.¹⁵ Das System berechnet, welche Sequenzen von 2 bis 5 Wörtern der Ausgangssprache (sogenannten Wort-n-Grammen) welchen Wortsequenzen der Zielsprache entsprechen. In unserem Beispiel wird festgehalten, dass *weiterhin nachfrageschwach* eine Übersetzung von *demand continued to be weak* ist. Bei diesem Schritt werden auch Wahrscheinlichkeiten für die Wortfolgen-Entsprechungen berechnet. Dieses Wortfolgenlexikon mit den Entsprechungswahrscheinlichkeiten bildet das sogenannte **Übersetzungsmodell**. Es ist der Kern des SMÜ-Systems.

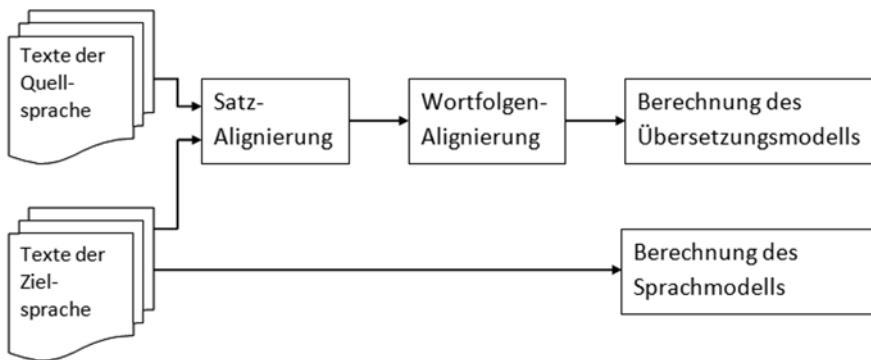


Abbildung 5.7: Training eines SMÜ-Systems

¹⁵Wir verwenden den Begriff „Wortfolgenlexikon“, der in der englischsprachigen Literatur als *phrase lexicon* bezeichnet wird. Dieser Begriff ist irreführend, da er suggeriert, dass im linguistischen Sinne Phrasen aligniert werden. Dies ist nicht der Fall. Es handelt sich um die Alignierung von irgendwelchen Wortfolgen.

Bei der Übersetzung wird ein Eingabesatz mit Hilfe des Übersetzungsmodells in mögliche Ausgabesätze, oder – genauer – in mögliche Wortfolgen der Zielsprache übersetzt. Dies leistet ein Modul, das **Decoder** genannt wird. Der Decoder probiert – vereinfacht dargestellt – alle möglichen Kombinationen von Wortsequenzen des Eingabesatzes und erzeugt mit Hilfe des parallelen Wortfolgenlexikons entsprechende Kombinationen von Wortsequenzen in der Zielsprache mit den zugehörigen Wahrscheinlichkeiten. Diese Wahrscheinlichkeiten erlauben das Aufstellen einer Rangfolge der erzeugten Zielsprache-Wortfolgen. Indem man die Wortfolge mit der höchsten Wahrscheinlichkeit liefert, wäre im Prinzip die maschinelle Übersetzung abgeschlossen. Es hat sich jedoch gezeigt, dass diese Wortfolge-Wahrscheinlichkeiten die Qualität der Ausgabe nicht zuverlässig vorhersagen, da sie die Übergänge zwischen den Wortfolgen in der Zielsprache nicht berücksichtigen.

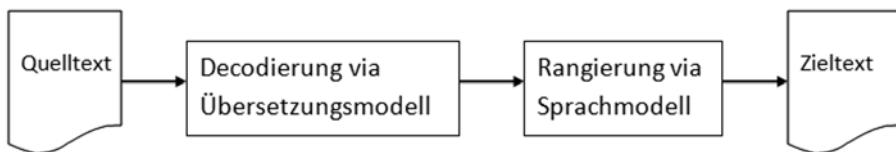


Abbildung 5.8: Übersetzen mit einem SMÜ-System

Deshalb erstellt man in der Trainingsphase neben dem Übersetzungsmodell auch ein sogenanntes **Sprachmodell** der Zielsprache. Dieses Sprachmodell beschreibt typische Wortfolgen innerhalb der Zielsprache. Es kann beispielsweise bestimmen, dass *demand continued to be weak* eine bessere (weil häufigere) Wortfolge eines englischen Satzes ist als *demand to be weak continued*. Das Sprachmodell wird auf einem großen einsprachigen Korpus der Zielsprache trainiert. Dies kann der zielsprachige Teil des Übersetzungskorpus sein, aber es kann auch ein anderes Korpus sein. Wichtig ist, dass dieses Korpus groß ist und möglichst der Textsorte und der Anwendungsdomäne entspricht. Technisch gesehen handelt es sich beim Sprachmodell um ein Hidden-Markov-Modell, das die Übergangswahrscheinlichkeiten von Wortfolgen aufgrund eines lokalen Kontextes lernt.

Die Idee des Sprachmodells hat die SMÜ übrigens bei der Automatischen Spracherkennung entliehen (siehe Unterkapitel 5.4). In beiden Fällen geht es darum, aus einer Liste von möglichen Wortfolgen eine grammatisch korrekte und natürlich klingende Abfolge auszuwählen. Bei der automatischen Spracherkennung werden diese möglichen Wortfolgen aufgrund der akustischen Analyse erzeugt, und bei der SMÜ aufgrund des Übersetzungsmodells.

Übersetzungsmodell und Sprachmodell wirken also entlang zweier Dimensionen der Übersetzung. Das Übersetzungsmodell sichert die Übersetzungstreue und das Sprachmodell die Flüssigkeit in der Zielsprache. Dies spiegelt sich auch in der mathematischen Grundformel wieder. Das Ziel der SMÜ ist es, eine Übersetzung \hat{T} zu finden, die die Wahrscheinlichkeit des zielsprachlichen Satzes T

unter Annahme des Eingabesatzes S optimiert.

$$\hat{T} = \operatorname{argmax}_T p(T|S)$$

Mit Hilfe des Satz von Bayes kann man diese Wahrscheinlichkeit umrechnen, so dass gilt:

$$\hat{T} = \operatorname{argmax}_T \frac{p(T)}{p(S)} * p(S|T) = \operatorname{argmax}_T p(T) * p(S|T)$$

Die Wahrscheinlichkeit $p(T)$ steht dann für die Informationen des Sprachmodells, und die bedingte Wahrscheinlichkeit $p(S|T)$ für die Informationen des Übersetzungsmodells. Das Produkt der beiden Wahrscheinlichkeiten liefert eine Kennzahl für die Bewertung des Ausgabesatzes.

Forschung und Werkzeuge

Die Forschungsaktivitäten zur SMÜ haben in den letzten Jahren einen enormen Aufschwung erfahren. Die großen Sprachtechnologie-Konferenzen (ACL, COLING) verzeichnen eine Vielzahl von Beiträgen zur SMÜ. Dieser Boom beruht vor allem auf der freien Verfügbarkeit leistungsfähiger SMÜ-Werkzeuge und großer paralleler Korpora. Diese ermöglichen allen Forschern die vollautomatische Erstellung eines SMÜ-Systems innerhalb weniger Tage. Gegenwärtig ist der SMÜ-Werkzeugkasten Moses¹⁶ das Maß aller Dinge. Moses nutzt das Alignierungssystem GIZA, dessen Ausgabe zu einem Wortfolgen-basierten Übersetzungsmodell umgerechnet wird. Es umfasst weiterhin einen Beam-Search Decoder, und es erlaubt gar die Integration linguistischer Zusatzinformation (z. B. Wortarten oder Kasusinformation).¹⁷

Ein großer Teil der Forschung beruht auf frei-verfügbaren Korpora der Europäischen Union. Besonders beliebt sind das Europarl-Korpus mit den Redebbeiträgen des Europäischen Parlaments (rund 40 Millionen Wörter in 11 EU-Sprachen) sowie das Acquis Communautaire, die Sammlung der europäischen Gesetzestexte und Vereinbarungen (10–30 Millionen Wörter in 22 EU-Sprachen). Für andere Textsorten sind parallele Korpora in ähnlicher Größenordnung schwierig zu finden. Kleinere Sammlungen von parallelen Korpora für Software-Handbücher und Film-Untertitel befinden sich auf der OPUS-Webseite.¹⁸

Der größte Vorteil von SMÜ-Systemen ist, dass sie schnell und vollautomatisch erstellt werden können, wenn parallele Korpora in ausreichender Größe und Qualität vorliegen. Die Übersetzungsqualität von SMÜ-Systemen im Vergleich zu Regel-basierten Systemen wird unterschiedlich bewertet. Auch die Fehlertypen sind unterschiedlich. Man kann wohl allgemein sagen, dass gute SMÜ-Systeme sich der Qualität von guten Regel-basierten MÜ-Systemen annähern, aber bisher nur in Ausnahmefällen das Niveau der besten und langjährig erprobten Regel-basierten Systeme erreicht haben.

¹⁶ Unter <http://www.statmt.org/moses/> kann Moses herunter geladen werden.

¹⁷ Der Nutzen der linguistischen Information zur Verbesserung der Übersetzungsqualität in SMÜ-Systemen ist jedoch umstritten.

¹⁸ OPUS steht für Open Source Corpus: <http://urd.let.rug.nl/tiedeman/OPUS/>.

Als wichtige Probleme bei der SMÜ gelten:

1. Es ist schwierig, Übersetzungsfehler zu lokalisieren und zu beheben. Das Zusammenspiel der verschiedenen statistischen Werte in der Alignierung sowie im Übersetzungs- und Sprachmodell führen zu einem komplex verzahnten System, bei dem eine eindeutige Fehlerursache häufig nicht ausgemacht werden kann.
2. Auch bei großen Trainingskorpora bleibt ein daraus abgeleitetes Vollformenlexikon unvollständig. Insbesondere Sprachen mit reicher Flexionsmorphologie (wie z. B. Finnisch oder Türkisch) oder dynamischer Kompositatbildung (wie z. B. Deutsch oder Schwedisch) führen zu immer neuen Wortformen, die dem SMÜ-System unbekannt sind. Das Hinzufügen von entsprechenden Morphologie-Modulen in der Vorverarbeitung kann dieses Problem entschärfen.

Die kurzen Entwicklungszeiten von SMÜ-Systemen machen diese Systeme attraktiv für Organisationen mit einem großen Übersetzungsvolumen. Die Europäische Union unterhält weltweit den größten Übersetzungsdienst und ist deshalb seit Jahrzehnten einer der größten Nutzer von MÜ-Systemen. Um die Palette der verfügbaren Sprachpaare zu erweitern, fördert die EU SMÜ-Projekte, u.a. das große Euromatrix-Projekt.¹⁹ Im Rahmen dieses Projekts werden SMÜ-Systeme zwischen allen Sprachpaaren der EU erstellt. Diese Arbeiten haben u.a. interessante Ergebnisse über die unterschiedlichen Schwierigkeiten bei der automatischen Übersetzung zwischen den europäischen Sprachen zu Tage gebracht. So liegen die Übersetzungsergebnisse zwischen den romanischen Sprachen deutlich an der Spitze, während automatische Übersetzungen von oder nach Deutsch und Finnisch am schlechtesten abschneiden (siehe Koehn 2005). Im Euromatrix-Projekt wurden aber auch SMÜ-Methoden und -Werkzeuge weiterentwickelt, so dass jetzt verbesserte Verfahren zur Schätzung der Wahrscheinlichkeiten sowie hybride Architekturen zur Kombination von statistischen Verfahren mit linguistischen Informationen bereitstehen.

Statistische MÜ im praktischen Einsatz

Bisher gibt es erst wenige Beispiele für den praktischen Einsatz von SMÜ-Systemen. Eine erfolgreiche Anwendung ist die Übersetzung von Fernsehuntertiteln in Skandinavien. Die Computerlinguistik-Gruppe der Universität Stockholm hat in Kooperation mit einer Untertitel-Firma SMÜ-Systeme erstellt, die Film- und Fernsehuntertitel von Schwedisch nach Dänisch und Norwegisch übersetzen (vgl. Volk 2008). In Skandinavien werden fremdsprachige Fernsehsendungen nicht synchronisiert, sondern im Originalton mit Untertiteln in der jeweiligen Landessprache ausgestrahlt.

Bei der Untertitel-Firma werden die schwedischen Untertitel auf der Basis des (meist englisch-sprachigen) Filmskripts und Videos von speziell geschulten Übersetzern erstellt. Dieser Schritt umfasst die Übersetzung, die Zeitkodierung (mit

¹⁹Weitere Informationen zu Euromatrix findet man unter <http://www.euromatrix.net/>.

Ein- und Ausblendezeit) und meist eine Kürzung der Originaläußerungen. In einem zweiten Schritt wurden bisher die schwedischen Untertitel manuell übersetzt. Dieser Schritt wird nun durch maschinelle Übersetzung unterstützt. Das SMÜ-System übersetzt die schwedischen Untertitel nach Dänisch (bzw. Norwegisch), der Übersetzer kontrolliert die vorgeschlagenen Untertitel und korrigiert sie gegebenenfalls. Die beiden SMÜ-Systeme sind seit Anfang 2008 im praktischen Einsatz und haben zu einer deutlichen Zeitsparnis bei der Übersetzung geführt.

Die Erstellung dieser SMÜ-Systeme profitierte von drei positiven Aspekten. Erstens sind Untertitel kurze sprachliche Einheiten mit geringer sprachlicher Komplexität. Aufgrund der Gegebenheiten des Mediums ist ein Untertitel beschränkt auf maximal zwei Zeilen mit insgesamt 74 Zeichen (einschließlich Leerzeichen). Zweitens sind Schwedisch, Dänisch und Norwegisch eng verwandte Sprachen, die sich bezüglich Wortbildung gleichen (alle drei sind Komposita-Sprachen) und die sich bezüglich Wortreihenfolge wenig unterscheiden. Drittens konnte die Untertitel-Firma eine große Anzahl von humanübersetzten Untertiteln in den gewünschten Sprachpaaren als Trainingsmaterial zur Verfügung stellen: mehr als 50 Millionen Wörter pro Sprache. Durch die Zeitkodierung der Untertitel war zudem eine natürliche Basis für die Alignierung gegeben.

Weitere Beispiele für den Praxiseinsatz von SMÜ-Systemen liefert die Firma Language Weaver, die inzwischen SMÜ-Systeme für mehr als 20 Sprachpaare anbietet und zum Marktführer avanciert ist. Die Firma zielt auf Übersetzungs-services für große Behörden und Organisationen. Ihre Produkte wurden aber bereits auch von MÜ-Herstellern integriert, die bisher auf Regel-basierte Systeme gesetzt hatten.

5.7.5 Evaluation von MÜ-Systemen

Lange Zeit wurde die Qualität von MÜ-Systemen manuell ermittelt. Dabei wird einem Übersetzer der Ausgangstext und die Ausgabe des MÜ-Systems vorgelegt. Der Übersetzer hat die Aufgabe, jeden Ausgabesatz auf einer Skala zu bewerten (z. B. unverständlich, akzeptabel, perfekt). Um ein verlässliches Gesamтурteil zu erreichen, müssen mehrere Übersetzer dieselben Sätze bewerten. Oft werden auch verschiedene Dimensionen der Übersetzungsqualität getrennt betrachtet (z. B. grammatische Korrektheit, Flüssigkeit, Übersetzungstreue). Dies macht die manuelle Bewertung von Übersetzungen zu einem aufwändigen Prozess.

Seit einigen Jahren arbeiten die MÜ-Forscher deshalb mit automatischer Bewertung. Dabei wird die Ausgabe des MÜ-Systems mit einer oder mehreren von Humanübersetzern erstellten Referenzübersetzungen verglichen. Für den Vergleich gibt es unterschiedliche Methoden. Man kann z. B. die Anzahl der grundlegenden Wortoperationen (Ersetzungen, Einfügungen, Weglassungen) zählen, um von der Systemausgabe zur Referenzübersetzung zu kommen. Dies entspricht der **Levenshtein-Distanz** (vgl. Kapitel 6), die auch zum Vergleich von Buchstabenketten angewendet wird. Dieses Vergleichsmaß ist jedoch sehr grob.

Papineni et al. (2001) haben deshalb ein Verfahren vorgeschlagen (genannt BLEU), bei dem nicht einzelne Wörter verglichen werden sondern Folgen von

zwei, drei oder vier Wörtern (vgl. Kap. 6). Die Entwickler behaupten, dass die von BLEU gelieferten Werte eine hohe Korrelation mit dem menschlichen Qualitätsurteil aufweisen. Die Korrelation steigt, wenn mehrere Referenzübersetzungen vorliegen (was jedoch in der Praxis selten ist).

Dieses automatische Evaluationsverfahren erfreut sich seither großer Beliebtheit sowohl in seiner ursprünglichen als auch in modifizierter Fassung. So beruht beispielsweise der NIST-Score auf dem BLEU-Score mit dem Unterschied, dass Wörter mit geringer Häufigkeit ein größeres Gewicht bekommen.

NIST²⁰, das amerikanische *National Institute of Standards and Technology*, hat einen großen Einfluss auf die Forschungen in SMÜ, weil dieses Institut seit 2001 alljährlich einen Wettbewerb ausrichtet, um SMÜ-Systeme für eine Reihe von Sprachpaaren zu vergleichen. Zuletzt war auffällig, dass Teilnehmer, die Zugang zu großen parallelen Korpora haben (wie z. B. Suchmaschinenhersteller) besonders gut abgeschnitten haben.

Inwieweit die Werte von BLEU und NIST tatsächlich menschlichen Qualitätsurteilen entsprechen, ist in den vergangenen Jahren heftig diskutiert worden. Einige Untersuchungen haben deutliche Zweifel an dieser These angebracht (z. B. Callison-Burch et al. 2006). Insbesondere wurde darauf hingewiesen, dass die automatischen Evaluationsverfahren SMÜ-Systeme gegenüber Regel-basierten Systemen bevorzugen, da sie auf Statistiken beruhen, die Ähnlichkeiten mit den in der SMÜ benutzten Wortfolgen haben. Für den Vergleich zwischen SMÜ-Systemen aber auch von unterschiedlichen Varianten eines SMÜ-Systems werden die automatischen Evaluationsverfahren jedoch weiterhin gern genutzt.

5.7.6 Computergestützte Übersetzung – CAT

Für die Arbeit professioneller Übersetzer stehen seit mehr als 10 Jahren Softwarewerkzeuge zur Verfügung, mit denen mehrsprachige Lexikon-, Terminologie- und Textdatenbanken (Alignments, vgl. 5.7.4) genutzt oder vom Übersetzer selbst aufgebaut und gepflegt werden können (**Computer-Aided Translation, CAT**). Allerdings waren die Softwarewerkzeuge zur computergestützten Übersetzung bisher kaum Gegenstand computerlinguistischer Forschung und werden fast ausschließlich kommerziell entwickelt (vgl. Reinke 2004, S. 4).

Dieser Umstand mag teilweise dadurch bedingt sein, dass es sich um spezielle Anwendungen für Experten handelt, die eng an deren Bedürfnisse und Arbeitsweise angepasst werden. Hieraus ergibt sich aber auch, dass die CAT-Werkzeuge oft nicht oder nur annähernd nach konsistenten Konzepten oder internationalen Standards, z. B. dem Anspruch an Änderbarkeit und Übertragbarkeit aus der ISO (2004), entwickelt werden und somit häufig sowohl den in Kapitel 6 diskutierten Qualitätsmerkmalen als auch translationswissenschaftlichen Erkenntnissen nur bedingt entsprechen. In einigen Fällen wird sogar der Übersetzungsprozess selbst durch die CAT-Software umstrukturiert bzw. determiniert. Auch das Wiederauffinden ähnlicher aber nicht identischer Übersetzungseinheiten als Referenzmaterial scheint aufgrund der geringen Forschungsaktivität im Bereich

²⁰Die MÜ-Evaluationen von NIST sind beschrieben unter <http://www.nist.gov/speech/tests/mt/>.

CAT nicht zufriedenstellend gelöst zu sein (vgl. Reinke 2004, S. 385). Zur Verbesserung bestehender oder Entwicklung neuer CAT-Systeme und übersetzungsrelevanter texttechnologischer Anwendungen ist daher die Expertise professioneller Übersetzer relevant. Gleichzeitig kann aufgrund der wachsenden Nachfrage nach Übersetzungen der Bereich CAT als relevantes Berufsfeld für Computerlinguisten angesehen werden. So listet Hutchins (2008) weltweit mehr als 190 Firmen auf, die sich mit MÜ und CAT beschäftigen.

Die zentralen Komponenten einer CAT-Umgebung sind die Arbeitsumgebungen für das Übersetzen, das Alignment und die Erstellung von Terminologieeinträgen.

Wenn der Übersetzer mit einem CAT-System übersetzt, speichert das System übersetzte Satzpaare aus Ausgangssprache und Zielsprache in einer Datenbank. Es speichert also den englischen Satz zusammen mit dem deutschen Satz, wie in dem bereits in Abschnitt 5.7.4 angeführten Beispiel (5.5) deutlich wird.

- (5.5) Die Zellstoff- und Papierindustrie sowie der Bausektor waren indessen weiterhin nachfrageschwach.
Demand continued to be weak in the pulp and paper and construction sectors.

Wenn derselbe englische Satz später erneut übersetzt werden soll, bietet das System die gespeicherte deutsche Übersetzung automatisch an. Der Übersetzer kann die gespeicherte Übersetzung direkt übernehmen oder entsprechend anpassen.

Dass ein identischer Satz wieder übersetzt werden soll, kann innerhalb eines Textes vorkommen (z. B. Warnungshinweise in Bedienungsanleitungen), in unterschiedlichen Versionen desselben Textes (z. B. in leicht geänderten Hilfetexten zu verschiedenen Software-Versionen) oder auch in neuen Texten (z. B. wichtige Dialogelemente wie *Wann sehen wir uns wieder?* wiederholen sich in Untertiteln zu verschiedenen Fernsehserien). Dennoch stellt die Übersetzung des genau gleichen Satzes einen Idealfall dar. Wenn für einen zu übersetzenenden Satz keine exakte Entsprechung gespeichert ist, ist es jedoch auch nützlich, einen **ähnlichen** Satz im Übersetzungsspeicher zu finden.

Deshalb bieten die CAT-Systeme auch eine Ähnlichkeitssuche an. Die Ähnlichkeit von Sätzen wird dabei über die Anzahl identischer bzw. abweichender Wörter berechnet. Wenn also zu einem Eingabesatz von 9 Wörtern (wie in Beispiel 5.6) ein Satz in der Datenbank gefunden wird, der in allen Wörtern übereinstimmt, aber 2 Wörter länger ist (wie in Beispiel 5.5), so ergibt das eine Ähnlichkeit von 9/11 oder 82%.

- (5.6) Die Papierindustrie sowie der Bausektor waren indessen weiterhin nachfrageschwach.

Dabei wird üblicherweise nicht unterschieden, um welche Wortklassen es sich bei den Differenzen handelt. Ein fehlendes Adverb wird genauso gewichtet wie ein fehlendes Substantiv.

Es ist klar, dass Übersetzungsvorschläge aus der Datenbank nur bis zu einem gewissen Schwellwert nützlich sind. Es ist kaum sinnvoll, dass für einen Eingabesatz mit 20 Wörtern ein Übersetzungsvorschlag gemacht wird, wenn nur eine Ähnlichkeit von 50% vorliegt. Deshalb ist üblicherweise ein Schwellwert von 80% oder 85% vordefiniert, den der Benutzer jedoch nach seinen aktuellen Bedürfnissen ändern kann.

Die Übersetzungsspeicher-Funktionalität hat für professionelle Übersetzer mehrere Vorteile:

1. Bereits geleistete Arbeit kann leicht wieder verwendet werden.
2. Übersetzungsvorschläge des Systems sind zuverlässig, da nur Humanübersetzungen angeboten werden.
3. Das System unterstützt die konsistente Übersetzung.

Übersetzungsspeicher haben jedoch auch Nachteile:

1. Eine gespeicherte Übersetzung von schlechter Qualität findet bei Bedienungsfehlern ihren Weg in weitere Dokumente.
2. Eine gespeicherte Übersetzung passt manchmal nicht in den aktuellen Kontext. Eine Übernahme kann die Textkohärenz verletzen.

Das **Alignierungsprogramm** berechnet, welcher Satz der Ausgangssprache mit welchem Satz der Zielsprache übersetzt wurde, vgl. auch Abschnitt 5.7.4. Diese Verbindungen können anschließend von dem menschlichen Übersetzer nachbearbeitet werden.

Außerdem können Alignments auch sozusagen nebenbei erstellt werden, wenn der Übersetzer seine Übersetzung mit einem CAT-System anfertigt.

Terminologieinträge enthalten neben den Termini selbst zusätzliche Informationen, z. B. eine linguistische Analyse und eine Definition des Begriffs, **Konkordanzen** und zielsprachliche Entsprechungen. Für die Erstellung von Terminologieinträgen kann auf weitere computerlinguistische Anwendungen wie Korrekturprogramme, (vgl. Unterkapitel 5.1), automatische Termextraktion (vgl. Unterkapitel 5.2) oder die Texttechnologie zurückgegriffen werden.

Eine in das CAT-System integrierte MÜ-Komponente kann auf der Basis der Alignments für ähnliche Texte eine Vorübersetzung erstellen, die zur Berechnung des Zeitaufwandes und der Kosten für die vollständige Übersetzung verwendet wird. Häufig schicken die Auftraggeber von Übersetzungen bereits vorhandene Datenbankeinträge zusammen mit dem neu zu übersetzenden Text an den Übersetzer. Allerdings kann man in Bezug auf die MÜ-Komponenten in CAT-Systemen unseres Erachtens nicht von MÜ sprechen, da die Alignments von Menschen erstellt bzw. nachbearbeitet wurden.

5.7.7 Aktueller Stand und Perspektiven

Der Übersetzungsmarkt ist ein riesiges Geschäft, das mit starkem Zeitdruck und mit einer wachsenden Informationsflut konfrontiert ist. So sind im indexierbaren WWW nach Gulli und Signorini (2005) 11.5 Milliarden Seiten vorhanden.

Sicherlich muss nicht alles übersetzt werden, aber die meisten Produkte werden gegenwärtig mit Beschreibungen, technischen Dokumentationen und Gebrauchsanweisungen in der Zielsprache des Käuferlandes versehen.

Bei kürzer werdenden Produktzyklen und einer steigenden Anzahl von Produkten auf dem Markt wächst der Übersetzungsbedarf sehr schnell an.

Professionelle Übersetzer setzen heute vor allem Technologien aus dem Bereich CAT ein, mit deren Hilfe verschiedene Typen mehrsprachiger Korpora (Terminologeeinträge, alignierte Texte) erstellt und für weitere Übersetzungen genutzt werden können. Auf der Basis dieser Korpora ist in einigen Fällen auch ein effektiver Einsatz von Systemen zur maschinellen Übersetzung möglich (vgl. 5.7.6).

Seit mehr als 55 Jahren wird an MÜ-Systemen geforscht und entwickelt. Die Attraktivität aus Forschungssicht besteht unter anderem darin, dass verschiedenste Entwicklungsergebnisse aus dem Bereich der Computerlinguistik in einem System kombiniert werden können. Einige kommerzielle MÜ-Systeme werden schon jahrelang für den Großrechnerbetrieb angeboten und sind mittlerweile für den Endverbraucher als PC-Software erhältlich bzw. stehen im Internet frei zur Verfügung. Die großen qualitativen Durchbrüche sind bisher allerdings nicht eingetreten: Für qualitativ hochwertige Übersetzungen – sowohl in literarischen und technischen als auch in entscheidungskritischen Bereichen – werden weiterhin professionelle Übersetzer eingesetzt. Allerdings sind die Übersetzer außerhalb von literarischen Übersetzungen zwingend auf die Verwendung von CAT-Systemen und texttechnologischen Werkzeugen angewiesen, um den wachsenden Anforderungen begegnen zu können. Wie erwähnt, sind die vorhandenen – meist kommerziellen – Systeme allerdings selten für den Übersetzungsprozess optimiert. Damit steht dem Forschungsdesiderat *Kombination computerlinguistischer Entwicklungen in einem MÜ-System* ein echter Entwicklungsbedarf im Bereich CAT gegenüber.

In Autorensystemen, wie sie beispielsweise zur Erstellung von Handbüchern für technische Geräte verwendet werden, wird teilweise **kontrollierte Sprache** (*controlled language*) eingesetzt, um die anschließende (automatische) Weiterverarbeitung – also auch die MÜ – zu vereinfachen. Diese Systeme schränken sowohl den Wortschatz als auch die grammatischen Konstruktionen ein, die vom Autor verwendet werden können. Man erhofft sich davon, auch unabhängig von einer Übersetzung, verständlichere und konsistenter Texte. Auf Bedarf muss der Autor auch Mehrdeutigkeiten in der Quellsprache auflösen, was aus Sicht der MÜ einer Vorbearbeitung entspricht.

Insgesamt zeichnet sich ein wachsender Bedarf an MÜ-Anwendungen ab, der aus Forschungssicht nicht allein kommerziellen Entwicklern überlassen werden sollte.

5.7.8 Literaturhinweise

Eine gute Übersicht zur Maschinellen Übersetzung (mit einer kurzen und nicht mehr ganz aktuellen Einführung in SMÜ) findet sich in Somers (2003).

Vertiefende Darstellungen bezüglich der verschiedenen Aspekte der SMÜ finden sich z. B. in Bezug auf die verschiedenen Ansätze zur automatischen Satzalignie-

rung in Kapitel 13 von Manning und Schütze (2003). Och und Ney (2003) bieten den fundierten Hintergrund und eine Anzahl praktischer Experimente dazu.

Die SMÜ wurde ab Anfang der 90er Jahre bekannt. Brown et al. (1990) sowie Brown et al. (1993) beschreiben die Grundideen der SMÜ und die bahnbrechenden Experimente bei IBM. Diese Arbeiten wurden später erweitert zur Wortfolgen-basierten SMÜ und beschrieben in Koehn et al. (2003) und Och und Ney (2004). Ein umfassendes Buch zur SMÜ ist für den Sommer 2009 angekündigt (Koehn 2009). Die gängigen SMÜ-Werkzeuge GIZA++ und Moses sind beschrieben in Och und Ney (2000) und Koehn et al. (2007).

Reinke (2004) gibt einen umfassenden Überblick über den Stand der Forschung im Bereich CAT, Massion (2005) vergleicht CAT-Systeme.

6 Evaluation von sprachverstehenden und -generierenden Systemen

Kapitelherausgeber: Hans-Peter Hutter und Susanne Jekat

In diesem Kapitel wird aufgezeigt, wie die Evaluation sprachverarbeitender Systeme gestaltet werden kann. Dazu werden zunächst grundlegende Begriffe eingeführt (Unterkapitel 6.1), die verschiedenen Gründe für eine Evaluation dargelegt (Abschnitt 6.1.1) und daraus das weitere Vorgehen abgeleitet. Anschließend werden die Evaluationsmethoden (Abschnitt 6.1.2) und Qualitätsmerkmale (Abschnitt 6.1.3) vorgestellt. Die eingeführten Begriffe und Konzepte zur Evaluation werden am Beispiel der Evaluation von Systemen mit unterschiedlichen zentralen Funktionen (Spracherkennung, Dialogsysteme, Sprachsynthese und Maschinelle Übersetzung, vgl. Unterkapitel 6.2) aufgezeigt.

6.1 Einführung

Nach Sonntag (1999, S. 7) ist Evaluation

die *Bestimmung eines Wertes*. Das heißt, einem Gegenstand wird ein bestimmter Wert zugeordnet. Diese Zuordnung muss sinnvoll sein, das heißt, es muss ein bestimmter Maßstab bestehen, nach dem diese Zuordnung erfolgt. Die Qualität einer Messmethode basiert auf ihrer Validität und Reliabilität, das heißt, sie muss genau das messen, was sie zu messen vorgibt, und die Messergebnisse müssen reproduzierbar sein.

6.1.1 Warum wird evaluiert?

Die zwei wesentlichen Gründe für die Evaluation von Softwaresystemen sind die

- **Systemverbesserung** und der
- **Systemvergleich**.

Die Systemverbesserung kann entweder auf der Basis der Evaluation durchgeführt oder – als Ergebnis – durch die Evaluation dokumentiert werden. Wichtig

für den Entwurf und die Bewertung von evaluierenden Maßnahmen für sprachverstehende und sprachgenerierende Systeme ist, dass die Evaluation nicht nur einzelne Aspekte des jeweiligen Systems beurteilt, sondern auch die zentrale Funktion des Systems mit den Eigenschaften ihrer einzelnen Komponenten und ihrer Zusammenarbeit untereinander in Verbindung setzen muss.

Eine 80 % Wörterkennungsrate mag beispielsweise ein gutes Ergebnis für einen Spracherkenner sein. Wenn diese hohe Erkennungsrate darauf basiert, dass häufig vorkommende und dadurch leicht trainierbare Wörter wie *ich* oder *nicht* immer erkannt werden, viele andere aber nicht, kann trotz der hohen Wörterkennungsrate u. U. eine sehr niedrige Satzerkennungsrate resultieren, die die Weiterverarbeitung erheblich erschwert.

Die zentrale Funktion und die Eigenschaften eines Systems haben starken Einfluss auf die Konzeption der Evaluation (vgl. auch King et al. 2003). Dabei kann sich eine Evaluation eher an den Bedürfnissen der potenziellen Benutzer einer Software orientieren. Diese sogenannte **benutzerorientierte Evaluation** wird auch als Adäquatheitsevaluation (engl. *adequacy evaluation*, King et al. 1996) bezeichnet. Das Ziel dabei ist festzustellen, wie gut sich ein bestimmtes System für den vorgesehenen Einsatz mit den vorgesehenen Benutzern eignet, evtl. auch im Vergleich zu anderen Systemen. Die **entwicklerorientierte Evaluation**, die auch als Fortschrittsevaluation (engl. *progress evaluation*, King et al. 1996) bezeichnet wird, richtet sich dagegen nach den Bedürfnissen der Systementwickler. Ziel dabei ist es, den Entwicklungsfortschritt zu messen bzw. zu dokumentieren und verschiedene Systeme zu vergleichen. Benutzerorientierte und entwicklerorientierte Evaluation schließen einander nicht aus, sie können kombiniert werden, um aus einer Evaluation Ergebnisse für beide Interessengruppen bereitzustellen.

6.1.2 Wann und wie wird evaluiert?

Die Evaluation eines Systems beginnt mit dem Festlegen der Anforderungen an das System zu Beginn eines Forschungs- oder Entwicklungsprojekts. Nicht nur im Forschungsbereich Sprachverarbeitung werden immer häufiger sogenannte Evaluationskampagnen für die verschiedenen Systeme organisiert, sondern inzwischen auch in weiteren Bereichen der Softwareentwicklung. Bei diesen treten die Systeme der beteiligten Forschungslabors oder -konsortien in einem Wettbewerb gegeneinander an. Dabei wird zu Beginn festgelegt, nach welchen Kriterien die Systeme getestet werden und welche Daten für das Training und die Optimierung der Systeme verwendet werden dürfen. Die eigentlichen Testdaten für die Evaluation erhalten die Teilnehmer jedoch nicht. Organisationen, die regelmäßig solche Evaluationskampagnen durchführen sind in den USA vor allem DARPA (Defense Advanced Research Agency¹) und NIST (National Institute of Standards and Technology², vgl. auch Abschnitt 6.2.5), in Europa sind es beispielsweise die ELRA/ELDA (European Language Ressource Agency³) oder CLEF/TrebleCLEF (Cross Language Evaluation Forum⁴).

¹<http://www.darpa.mil>

²<http://www.nist.gov>

³<http://www.elra.info>

⁴<http://www.trebleclef.eu>

Für die Entwicklung von kommerziellen sprachverarbeitenden Systemen schlagen Cohen et al. (2004, S. 38) verschiedene Phasen vor, ähnlich wie bei der Entwicklung von Softwaresystemen:

1. Anforderungsdefinition
2. High-Level Design
3. Detail-Design
4. Implementation
5. Test
6. Tuning.

Die Anforderungen an das zu entwickelnde System werden in der ersten Phase, der Anforderungsdefinition im Detail zusammen mit dem Kunden festgelegt. In dieser Phase geht es nicht nur darum, die gewünschte Funktionalität, sondern vor allem auch das Zielpublikum und den Geschäftskontext, in dem das System zum Einsatz kommt, zu verstehen. Daraus werden zusammen mit dem Kunden die Erfolgsfaktoren, anhand derer das System am Ende des Projekts evaluiert wird, abgeleitet.

Sowohl in Forschungs- wie auch in Entwicklungsprojekten werden die Systeme in den meisten Fällen mit Hilfe eines sogenannten **Blackbox-Tests** evaluiert. Bei diesem **dynamischen Testverfahren** wird das System nur von außen bewertet, indem über eine vordefinierte Schnittstelle Testdaten an das System angelegt werden und die vom System daraus generierten Resultate mit den erwarteten Resultaten (Referenzwerte) verglichen werden. Dieses Testverfahren hat den Vorteil, dass verschiedene Systeme gleich welcher Herkunft mit demselben Testverfahren getestet werden können, solange diese die geforderten Schnittstellen enthalten. Für kommerzielle Systeme wird diese Art von Test auch bei der Abnahme des Systems durch den Kunden sehr häufig angewendet.

Im Gegensatz dazu wird beim sogenannten **Glass- bzw. White-Box-Test** das Innenleben des zu testenden Systems ebenfalls berücksichtigt bzw. evaluiert. Diese Art von Tests werden vor allem während der Entwicklung des Systems zur Fehleranalyse oder Systemdiagnose verwendet. Auch dieser Test gehört zu den dynamischen Testverfahren, bei denen das System mit konkreten Testdaten ausgeführt wird.

Daneben existieren verschiedene **statische Testverfahren** wie *Walkthrough*, *Inspektion*, *Review* oder verifizierende bzw. analysierende Verfahren. Bei diesen Testverfahren wird das zu testende System nicht ausgeführt sondern anhand der Dokumentation evaluiert (vgl. z. B. Balzert 1998, S. 397, Jones und Galliers 1996).

Bei der Entwicklung von kommerziellen Sprachverarbeitungssystemen, insbesondere beim Design von Voice User Interfaces (VUI), wird das zu entwickelnde System von der ersten Entwicklungsphase an regelmäßig von Benutzern aus dem Zielpublikum evaluieren Cohen et al. 2004:

- Da in den frühen Phasen der Systementwicklung meistens noch kein lauffähiges System verfügbar ist, werden die ersten Tests als sogenannte **Wizard-of-Oz Tests** durchgeführt, um einerseits Designfragen im Highlevel-Design zu klären und andererseits realistische Sprachdaten für das Training, die Parameter-Optimierung oder spätere Systemtests zu sammeln. Dabei interagieren die Benutzer mit einem scheinbar fertigen System, das sich gemäß dem geplanten Dialogkonzept verhält. In Wirklichkeit sitzt hinter dem User Interface eine Person, die die Spracheingaben anhört und die gemäß dem Dialogkonzept vorgesehene Antwort generiert. Dabei geht die Person nach einem detailliert vorgegebenen Drehbuch vor. Dieses Verfahren lässt in einer sehr frühen Entwicklungsphase bereits Usability-Evaluationen zu, mit denen wichtige Designentscheidungen abgestützt werden können. Das Verhalten der Benutzer bei Fehlern des Sprachverarbeitungssystems lässt sich damit allerdings nur soweit prüfen, wie diese Fehler von den Testern antizipiert werden können.
- Bei **Labortests** wird das System im Labor mit aufgenommenen Sprachkorpora getestet, um Entwicklungsfortschritte zu belegen. Solche Korpora werden von verschiedenen Organisationen wie beispielsweise der ELRA⁵ oder dem LDC⁶ gesammelt und verbreitet.
- **Regressionstest:** Vor allem bei sprachverarbeitenden Systemen ist es sehr wichtig, die Labortests so zu organisieren, dass sie jederzeit möglichst automatisiert wiederholt werden können. Damit kann nach einer Änderung am System einerseits der Entwicklungsfortschritt und andererseits die weiterhin korrekte Funktionalität des Systems belegt werden.
- Beim **Feldtest** bzw. **Pilottest** wird das mehr oder weniger fertige System im realen Einsatz mit realen Benutzern getestet. Die gesamte sprachliche oder allenfalls multimodale Interaktion der Benutzer mit dem System wird dabei aufgenommen und die Systemaktionen werden protokolliert. Diese Daten werden einerseits für Parameteroptimierungen (Lexikon, Grammatik, Schwellwerte, Time-Outs) verwendet. Andererseits können damit Problemstellen im Dialog ausfindig gemacht werden. Schließlich sind die aufgenommenen Daten auch für spätere Regressionstests gut zu gebrauchen.

6.1.3 Was wird evaluiert?

Da sprachverarbeitende Systeme heute immer Softwaresysteme sind, bilden die Standards für die Evaluation von Softwaresystemen die Grundlage für die Evaluation dieser Systeme. Bei der Evaluation von Softwaresystemen geht es grundsätzlich um die **Qualität**. Diese wird in der ISO/IEC-Norm 9126 ISO/IEC (2001, S. 20) definiert als:

⁵<http://www.elra.info>

⁶<http://www.ldc.upenn.edu>

Die Gesamtheit der Eigenschaften eines Software-Produkts, die sich auf dessen Eignung beziehen, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen.

Die Version 2001 dieser Norm führt neben der **internen** und **externen** Qualität eines Systems neu die **Benutzungsqualität (quality in use)** ein, die weiter unter näher erläutert wird. Die interne Qualität beschreibt die Eigenschaften des Systems aus system-interner Sicht und ist vor allem bei der Entwicklung des Systems wichtig. Die externe Qualität beschreibt die Eigenschaften des Systems von außen betrachtet. Diese Sicht ist bei der Abnahme eines Systems durch den Auftraggeber oder bei Vergleichen verschiedener Systeme wichtig. Für die interne und externe Qualität definiert die ISO/IEC 9126 Norm folgende sechs Hauptmerkmale für die Softwarequalität, die in weitere Merkmale unterteilt sind:

- **Funktionalität**

Sind die notwendigen Funktionen mit den nötigen Eigenschaften vorhanden, um festgelegte oder implizierte Anforderungen zu erfüllen, wenn das System unter vereinbarten Bedingungen benutzt wird?

- **Zuverlässigkeit**

Wie gut kann das System sein Leistungsniveau unter festgelegten Bedingungen über einen festgelegten Zeitraum bewahren?

- **Usability**

Wie aufwändig und attraktiv ist die Benutzung und die Erlernbarkeit des Systems aus der Sicht einer festgelegten oder vorausgesetzten Benutzergruppe?

- **Effizienz**

Wie verhält sich das Leistungsniveau zum Umfang der eingesetzten Betriebsmittel unter festgelegten Bedingungen?

- **Änderbarkeit**

Wieviel Aufwand ist notwendig, um eine vorgegebene Änderung durchzuführen (auf Grund von Korrekturen, Verbesserungen, neuen Umgebungen, geänderten Anforderungen)?

- **Übertragbarkeit**

Wie gut lässt sich die Software von einer Umgebung in eine andere übertragen (Hardware-, Software- oder organisatorische Umgebungen)?

Die **Benutzungsqualität** beschreibt die Fähigkeit von Mitgliedern einer vorgegebenen Benutzergruppe, mit dem System vorgegebene Aufgaben effizient, effektiv, sicher und zu ihrer Zufriedenheit ausführen zu können. Die Definition der Benutzungsqualität entspricht genau dem Qualitätsbegriff **Usability**, wie er in der Norm ISO/IEC 9241-11:1998 (vgl. ISO/IEC 1998) bereits definiert wurde. Die spätere ISO/IEC-Norm 9126 (ISO/IEC 2001) erweitert die drei Hauptmerkmale der Usability um das Merkmal Sicherheit (Safety):

- **Effektivität** (Effectiveness)

Wie vollständig und genau kann ein Benutzer ein vorgegebenes Ziel erreichen?

- **Produktivität** (Productivity; in ISO 9241 mit Effizienz bezeichnet)

Wieviel Aufwand braucht ein Benutzer, um ein vorgegebenes Ziel zu erreichen, im Verhältnis zur Genauigkeit und Vollständigkeit, mit der das Ziel erreicht wird?

- **Zufriedenheit** (Satisfaction)

Wie angenehm ist die Benutzung des Produkts bzw. wie wenig Unannehmlichkeiten sind damit verbunden?

- **Sicherheit** (Safety)

Wieviele Risiken (für den Benutzer, das System, das Geschäft, die Öffentlichkeit, die Umwelt) sind mit der Benutzung des Systems verbunden?

Eine hohe Benutzungsqualität ist letztendlich das Ziel jedes kommerziellen Systems. Sie kann aber erst evaluiert werden, wenn das System in der realen Umgebung in Betrieb ist. Die Evaluation der internen Qualität und externen Qualität kann dazu verwendet werden, Prognosen für die spätere Benutzungsqualität abzuleiten. Popescu-Belis (2008) beschreibt, wie die drei Qualitätsbegriffe interne, externe und Benutzungsqualität auf entsprechende HLT (Human-Language-Technology)-Systeme angewendet werden können, um die Benutzungsqualität bereits in Evaluations-Kampagnen zu berücksichtigen. Popescu-Belis et al. (2006) schlagen dazu das **Generic Contextual Quality Model (GC-QM)** vor, innerhalb dessen Experten für jeden Benutzungskontext eines HLT-Systems die wichtigen externen Qualitätsfaktoren bestimmen und gewichten. Bei der Evaluation eines HLT-Systems bezüglich eines Benutzungskontexts werden sodann die externen Qualitätsmerkmale des Systems mit den durch den Benutzungskontext vorgegebenen Gewichtungen verrechnet, um die Eignung des HLT-Systems für den entsprechenden Benutzungskontext zu bestimmen.

6.2 Evaluationskriterien für sprachverarbeitende Systeme

Wie bereits in den Abschnitten 6.1.1 und 6.1.3 erwähnt, muss die jeweilige Evaluation an die Eigenschaften des zu evaluierenden Systems angepasst werden. Im Folgenden werden typische Evaluationskriterien für verschiedene sprachverarbeitende Systeme kurz erläutert.

6.2.1 Spracherkennungssysteme

Übersicht

Für die Spracherkennung werden heute vorwiegend statistische Verfahren basierend auf sogenannten Hidden-Markov-Modellen (HMM) eingesetzt. Mit die-

sen wird ein MAP(Maximum-A-Posteriori)-Erkenner implementiert, der die wahrscheinlichste Wortfolge W^* anhand der a-posteriori-Wahrscheinlichkeiten $P(W_i|O)$ ermittelt. Es lässt sich leicht beweisen, dass ein MAP-Erkenner theoretisch die kleinstmögliche Fehlerrate besitzt. Die MAP-Wahrscheinlichkeiten $P(W_i|O)$ werden dabei nicht direkt bestimmt, sondern gemäß Formel (6.1) (auch als **Bayes-Formel** bekannt) aus drei anderen Wahrscheinlichkeiten berechnet, die einfacher geschätzt werden können.

$$W^* = \operatorname{argmax}_i P(W_i|O) = \operatorname{argmax}_i \frac{P(O|W_i)P(W_i)}{P(O)} \quad (6.1)$$

Unter der Voraussetzung, dass alle Wortfolgehypothesen W_i basierend auf derselben Beobachtungssequenz O verglichen werden, kann $P(O)$ weggelassen werden, da diese Wahrscheinlichkeit für alle W_i gleich (und > 0) ist. Die Wahrscheinlichkeiten $P(O|W_i)$ in der Formel (6.1) stellen das sogenannte **akustische Modell** dar. Diese Wahrscheinlichkeiten werden mit Hilfe der Hidden-Markov-Modelle für alle Wortfolgehypothesen W_i geschätzt.

Der zweite ebensowichtige Teil eines HMM-Erkenners wie das akustische Modell ist das **Sprachmodell** $P(W_i)$. Mit letzterem werden für alle möglichen Wortfolgen W_i (auch Hypothesen genannt) die a-priori Wahrscheinlichkeit berechnet. Das Sprachmodell kann als Grammatik (Regelwerk) oder als statistisches Sprachmodell vorliegen. Es enthält Wissen über die Syntax, aber auch die Semantik und Pragmatik der Erkennungsaufgabe. Es muss genauso sorgfältig trainiert werden wie das akustische Modell.

Der bei HMM-Erkennern normalerweise verwendete Viterbi-Dekodierungsalgorithmus berechnet nur die beste Wortfolgehypothese W^* . Häufig ist man aber auch an der zweit- oder drittbesten Satzhypothese interessiert. Eine Erweiterung des Viterbi-Algorithmus, der sogenannte N-best Viterbi-Algorithmus, ermöglicht die Berechnung der N besten Satzhypothesen. Diese werden in einer N-Besten Liste oder kompakter in einem **Worthypothesengraphen** (im Folgenden WHG) wie in Abbildung 6.1 dargestellt. Da die Berechnung der N-Besten Liste wesentlich aufwändiger ist, als der einfache Viterbi-Algorithmus, gibt es Näherungsverfahren für den exakten N-Best Viterbi-Algorithmus.

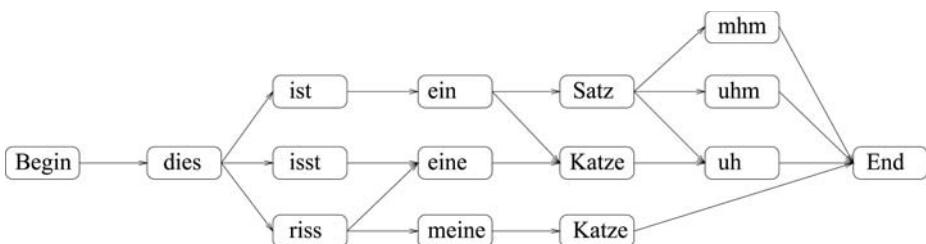


Abbildung 6.1: Worthypothesengraph

Für das Training und die Evaluation von Spracherkennungssystemen werden normalerweise drei verschiedene Sprachdatenkorpora benötigt: Das **Trainingsset**

wird verwendet, um die statistischen Modelle, z. B. die HMM der Grundelemente, zu trainieren. Das sogenannte **Entwicklungsset** wird sodann verwendet, um verschiedene Parameter des Erkennungssystems zu optimieren. Für die eigentliche Evaluation braucht man sodann ein eigenes von den anderen zwei Datensets möglichst unabhängiges **Testset** (gleiche Domäne aber andere Sprachaufnahmen von anderen Sprechern).

Evaluation der Spracherkennung

Das am häufigsten verwendete Evaluationskriterium für Spracherkenner ist die **Wortfehlerrate (WER)** oder **Wortakkurtheit (WA)**, wobei die Beziehung gilt: $WER = 100\% - WA$. Für die Evaluation wird ein Testset verwendet, indem für jede Testäußerung eine Referenztranskription vorliegen muss, mit der die erkannte Wortfolge verglichen wird. Für die Berechnung der Wortfehlerrate wird die **minimale Editierdistanz**, auch **Levenshtein-Editierdistanz** genannt, zwischen der vom Erkenner hypothetisierten Wortfolge und der Referenz berechnet. Diese Art der Evaluation wird dementsprechend **referenz-basierte Evaluation** genannt. Dabei wird die erkannte Wortfolge (Hypothese) so auf die Referenzwortfolge abgebildet, dass die Summe der notwendigen Korrekturen minimal wird (Okuda, Tanaka und Kasai 1976). Jede notwendige Korrektur wird sodann als Fehler gezählt. Dabei unterscheidet man zwischen Ersetzungen (substitutions), Auslassungen (deletions) und Einfügungen (insertions). Die Wortfehlerrate errechnet sich sodann aus der Anzahl Ersetzungen N_{sub} , Auslassungen N_{del} und Einfügungen N_{ins} gemäß der Formel:

$$WER = 100\% \cdot \frac{N_{sub} + N_{ins} + N_{del}}{N} \quad (6.2)$$

wobei N die Anzahl Wörter im Referenzsatz darstellt. Die optimale Abbildung der Hypothese auf die Referenz kann mit der Dynamischen Programmierung, insbesondere mit dem sogenannten **Dynamic Time Warping (DTW)**-Algorithmus berechnet werden Huang, Acero und Hon (2001). Abbildung 6.2 zeigt ein Beispiel für die Berechnung der Wortfehlerrate.

Referenz :	***	Elvis	sagt	wir	fahren	nach	Memphis
Hypothese:	ja	Elvis	sagt	vier	fahren	***	Memphis
Fehlerklasse:	INS	-	-	SUB	-	DEL	-
Wortfehlerrate:	$WER = 100\% \cdot \frac{1+1+1}{6} = 50\%$						

Abbildung 6.2: Berechnung der Wortfehlerrate

Bei der Evaluation von N-Best-Spracherkennern ist zu berücksichtigen, dass die Wortfehlerrate sinkt, je größer die Anzahl N der ausgegebenen Hypothesen ist. Liefert der N-Best-Erkenner einen WHG, so wird die Wortfehlerrate normalerweise bestimmt, indem die Wortakkurtheit zwischen der Referenz und der zur Referenz ähnlichsten aller im WHG vorhandenen Hypothesen berechnet wird.

Die Wortfehlerrate hängt somit von der Anzahl der Pfade im generierten WHG ab. Sie wird deshalb in Funktion der mittleren Dichte des WHG angegeben bzw. aufgetragen (siehe z. B. Malenke et al. 2000). Die mittlere *WHG-Dichte* aller im Test generierten Worthypothesengraphen wird berechnet aus der Anzahl Kanten (N_{Kanten}) aller im Test generierten WHG und der gesamten Anzahl im Testset gesprochenen Wörter in den zugehörigen Äußerungen ($N_{Wörter}$):

$$Dichte = \frac{N_{Kanten}}{N_{Wörter}}$$

Die Wortfehlerrate oder Wortakkuratheit ist ein geeignetes Maß, um den technischen Fortschritt eines Systems zu belegen, solange die Trainings- und vor allem die Testbedingungen gleich gehalten werden. So werden bei Evaluationskampagnen die Korpora für das Training, die Parameteroptimierung und den eigentlichen Test genau vorgegeben.

Wie man aus Formel (6.1) ersieht, hat neben den akustischen Modellen auch die Qualität des Sprachmodells eines Spracherkennungssystems einen entscheidenden Einfluss auf die Erkennungsrate. Die Qualität eines Sprachmodells aber auch die Komplexität der Erkennungsaufgabe kann mit Hilfe der **Perplexität** beschrieben werden, mit

$$PP(W) = 2^{H(W)} \quad (6.3)$$

$$H(W) = -\frac{1}{N_W} \log_2 P(W) \quad (6.4)$$

W ist dabei eine genügend lange Wortfolge aus dem Testset und N_W die Anzahl Wörter in der Wortfolge. Wird $P(W)$ mit Hilfe eines allgemeinen Sprachmodells, z. B. einem Bigram-Sprachmodell, berechnet, so gibt die Testset-Perplexität einen Hinweis auf die Komplexität der Erkennungsaufgabe. Die Perplexität PP kann nämlich interpretiert werden als der mittlere Verzweigungsfaktor am Ende jedes Wortes im Erkennungsnetzwerk. So hat z. B. ein Ziffernfolgenkenner mit 10 gleich wahrscheinlichen Ziffern eine Perplexität von 10. Im Vergleich dazu hatte der Wall-Street-Journal-Task, der von DARPA für die Evaluation von kontinuierlichen Spracherkennungssystemen verwendet wurde, bei Vokabularen von 5'000–20'000 Wörtern bloß Perplexitäten zwischen 80 und 240 (Paul und Baker 1992).

Wird für die Berechnung der Perplexität das Sprachmodell des zu evaluierenden Erkenners verwendet, so erhält man eine Schätzung für die Güte des Sprachmodells des Erkenners. Dieses ist umso besser, je näher dessen Perplexität an der Perplexität der Anwendungssprache an sich ist, die aus der Testset-Perplexität geschätzt werden kann (Huang et al. 2001, S. 561).

Obwohl die Perplexität einen Hinweis auf die Komplexität der Erkennungsaufgabe gibt, hängt die Schwierigkeit der Erkennungsaufgabe von weiteren Faktoren ab, vor allem von der Ähnlichkeit der Vokabularwörter. Ein Vergleich der Wortfehlerraten verschiedener Spracherkennungssysteme für unterschiedliche Aufgaben oder verschiedene Sprachen ist deshalb auch bei ähnlicher Perplexität problematisch.

Bei kommerziellen Systemen ist es sehr wichtig, dass die verwendeten Sprachdaten für das Trainingsset, Entwicklungsset und Testset möglichst genau mit dem Einsatzgebiet übereinstimmen. Dies betrifft u.a. die Aufnahmequalität (Telefonsprache, Mobilfunk- vs. Festnetz, Breitbandsprache), die Sprechumgebung (Büro, Auto, überall), die Sprechart (meistens Spontansprache), das Anwendungsgebiet (Sprachmodell) und vor allem die Benutzergruppe (Alter, Geschlecht, Sprachregion, Jargon). Aus diesem Grund werden bei kommerziellen Systemen die verwendeten Sprachdaten in möglichst realitätsnaher Umgebung mit realen Benutzern aus der Zielpopulation gesammelt. Zu Beginn der Entwicklung werden oft Wizard-of-Oz Tests dafür eingesetzt, später werden mit dem Pilotensystem Sprachdaten für die Parameteroptimierung und die Abnahmetests gesammelt.

Evaluation kommerzieller Systeme

Bei kommerziellen Spracherkennungssystemen spielen neben der Wortfehlerrate noch weitere Kriterien für die empfundene Qualität des Systems eine wichtige Rolle. Dazu gehört einerseits die **Vokubularabdeckung** bzw. die **Out-Of-Vocabulary-Rate (OOV-Rate)**. Die OOV-Rate besagt, wieviel Prozent der vom Benutzer verwendeten Wörter dem System unbekannt sind. Die OOV-Rate sinkt mit wachsender Vokabulargröße. Sie liegt beispielsweise für eine allgemeine Diktieraufgabe auf Englisch bei ca. 20% für eine Vokabulargröße von 20'000 Wörtern und sinkt bis auf 5% bei einer Vokabulargröße von 60'000 Wörtern. Für eine OOV-Rate von 0.5% benötigt man über 200'000 Wörter im Vokabular. Für stark flektierte Sprachen wie Deutsch ist die OOV-Rate ca. doppelt so hoch wie für Englisch bei gleicher Vokabulargröße (Huang et al. 2001, S. 579).

Eine Vergrößerung des Vokabulars führt aber automatisch zu einer Erhöhung der Perplexität, was sich wiederum negativ auf die Wortakkuratheit auswirkt. So steigt die Perplexität im oben erwähnten Beispiel von ca. 180 bei einer Vokabulargröße von 20'000 Wörtern auf über 400 bei einer Vokabulargröße von 60'000 Wörtern. Bei kommerziellen Systemen geht es also darum, eine möglichst gute Vokabularabdeckung mit möglichst geringer Perplexität zu erreichen. Dazu muss das Sprachmodell genau auf das spätere Einsatzgebiet abgestimmt werden.

Da es in praktisch eingesetzten Spracherkennungssystemen immer Wörter gibt, die das System nicht kennt, ist die Strategie der Zurückweisung (**Rejection**) ein wichtiges Thema. Dazu berechnet der Spracherkenner für jede erkannte Hypothese eine sogenannte **Konfidenz**, die aussagt, wie sicher er sich ist, dass die erkannte Hypothese die richtige ist. Die Konfidenz kann berechnet werden, indem man beispielsweise die berechnete Wahrscheinlichkeit für die erkannte Hypothese vergleicht mit der Wahrscheinlichkeit, dass dieselbe Beobachtungssequenz O durch irgendeine andere Hypothese erzeugt worden ist. Dazu schätzen viele Erkenner die Wahrscheinlichkeit $P(O)$ in Formel (6.1) mit Hilfe von $P(O) = \sum_{W_i} P(W_i)P(X|W_i)$. Diese Schätzung kann mit einem speziellen HMM für jede anliegende Beobachtungssequenz O berechnet werden. Details dazu sind beispielsweise in Huang, Acero und Hon (2001, S. 435) zu finden.

Mit Hilfe der errechneten Konfidenz und den dafür festgelegten Schwellwerten kann der Spracherkenner entscheiden, wann er eine Äußerung akzeptiert oder zurückweist (oder allenfalls eine Bestätigung verlangt). Da die Festlegung dieses Schwellwerts sowohl einen Einfluss auf die Anzahl Zurückweisungen als auch auf die eigentliche Wortfehlerrate hat, sollten bei einem kommerziellen System immer beide, die Wortfehlerrate und die Zurückweisungsrate, angegeben werden.

Schließlich spielt bei kommerziellen Systemen natürlich auch das **Echtzeitverhalten** eine wesentliche Rolle, da die Benutzer nur Systeme akzeptieren, die ohne merkliche Verzögerung reagieren. Bei Spracherkennungssystemen bedeutet Echtzeit, dass der Spracherkenner die Eingabe so schnell verarbeiten kann, wie die Benutzer sprechen. Damit steht das Erkennungsresultat bereit, sobald der Benutzer seine Eingabe abgeschlossen hat.

6.2.2 Evaluation von Dialogsystemen

Die meisten heutigen Sprachdialogsysteme sind aufgabenorientiert, wie z. B. Auskunftssysteme oder Buchungssysteme (Peckham 1993, Walker et al. 2000), in jüngerer Zeit werden aber auch nicht aufgabenorientierte Dialogsysteme entwickelt, z. B. Info- oder Edutainment-Systeme (vgl. Bernsen und Dybkjaer 2004; Moeller et al. 2004). Verschiedene nationale und internationale Projekte befassten und befassen sich mit der Evaluation von Sprachdialogsystemen, wie z. B. EAGLES (King et al. 1996) und das Nachfolgeprojekt ISLE⁷, ATIS (Pallett et al. 1994), DISC (Dybkjaer et al. 1998), SQALE (Young et al. 1997), ELSE⁸, Evalda⁹, COMMUNICATOR (Walker et al. 2000; Walker et al. 2001) oder NICE (Bernsen und Dybkjaer 2004; Dybkjaer et al. 2004).

Technische Evaluation

Die Evaluation von Dialogsystemen umfasst die Usability- sowie die technische Evaluation. Letzterer werden die Evaluation der einzelnen Komponenten zugeordnet. Für die Spracherkennung (vgl. 6.2.1) sind dies die Wortfehlerrate, Satzfehlerrate, Vokabularabdeckung, Perplexität, Echtzeitverhalten. Für die Sprachausgabe sind es Verständlichkeit, Natürlichkeit, Wohlklang, für die Sprachverständenskomponente die lexikalische Abdeckung, Grammatikabdeckung und das Echtzeitverhalten. Da Spontansprache eigenen grammatischen Gesetzen folgt, ist die Grammatikabdeckung aber ein kontrovers diskutiertes Thema. Neue Metriken wie z. B. die **Konzeptakkuratheit** (Boros et al. 1996) gewinnen deshalb mehr und mehr an Bedeutung. Die Konzeptakkuratheit misst, inwieweit die Sprachverständenskomponente die wichtigsten Konzepte aus der Spracheingabe erfasst hat. Diese basiert wie die Wortakkuratheit auf dem Zählen von Einfügungen, Auslassungen und Ersetzungen. Für eine umfassendere Beurteilung der Verständenskomponente über mehrere Äußerungen hinaus schlagen Glass et al.

⁷http://www.ilc.cnr.it/EAGLES96/isle/ISLE_Home_Page.htm

⁸<http://www.limsi.fr/TLP/ELSE>

⁹<http://www.elda.fr/rubrique25.html>

(1996) die **Query-Dichte** und **Konzept-Effizienz** als Maße vor. Die Query-Dichte misst die mittlere Anzahl neuer Konzepte pro User-Anfrage, während die Konzept-Effizienz die mittlere Anzahl Dialogwechsel angibt, die nötig ist, um dem System ein neues Konzept beizubringen.

Auf Gesamtsystemebene wurden verschiedenste quantitative Qualitätsmaße für die Evaluation von Dialogsystemen vorgeschlagen (Larsen 2003; Dybkjaer et al. 2004), wie z. B.:

- Prozentsatz der korrekten Systemantworten
- Prozentsatz der erfolgreichen Transaktionen
- Prozentsatz der Korrekturdialogäußerungen
- Prozentsatz der vom Benutzer initiierten Dialogakte
- Anzahl der Hilfe-Anforderungen
- Anzahl der Barge-Ins¹⁰ des Benutzers
- Anzahl erledigter Aufgaben und Teilaufgaben
- Dauer des Gesamtdialogs oder Zeitspanne für die Erledigung einer Aufgabe oder Teilaufgabe
- Mittlere Antwortzeiten der Benutzer und des Systems
- Prozentsatz der Äußerungen mit mehr als einem Wort
- Mittlere Länge der Äußerungen

Larsen (2003) setzt diese externen Qualitätsmaße in Beziehung zu verschiedenen Usability- und anderen Qualitäts-Kriterien für Sprachdialogsysteme, wie sie in Möller (2002) beschrieben sind.

Usability-Evaluation

Im Gegensatz zur technischen Evaluation von Sprachdialogsystemen (SDS), ist die Usability-Evaluation dieser Systeme noch wenig etabliert. Dybkjaer und Bernsen (2000) schlagen dreizehn Usability-Kriterien für task-orientierte Sprachdialogsysteme vor, die jede Person ohne vorheriges Training verwenden können soll. Dabei haben das System und der Benutzer ein gemeinsames Ziel, das möglichst schnell und effizient erreicht werden soll, beispielsweise das Buchen eines Fluges oder die Durchführung einer Bank-Transaktion. Die Usability-Kriterien für solche Systeme werden im Folgenden kurz erläutert:

¹⁰Barge-In bedeutet, dass der Benutzer die Systemausgabe mit einer Spracheingabe unterbricht, sofern das System dies erlaubt.

- **Erkennungsrate**

Die Erkennungsrate der Spracherkennung spielt eine zentrale Rolle auch bezüglich der Usability, da sie sich direkt auf die Effektivität, Produktivität und Benutzerzufriedenheit auswirkt. Neben der eigentlichen Erkennungsrate spielt dabei auch eine wichtige Rolle, wie das System bei auftretenden Fehlern reagiert.

- **Natürlichkeit der Spracheingaben**

Der Benutzer sollte einerseits so natürlich mit dem SDS sprechen können wie mit einer Person in der entsprechenden Anwendungssituation. Andererseits möchte man das Erkenntniskatalog und die Erkennungsgrammatik so klein wie möglich halten, um die Erkennungsrate zu optimieren. Diese beiden gegensätzlichen Forderungen können nur erfüllt werden, wenn die natürlichsprachlichen Dialoge zwischen Menschen in der entsprechenden Anwendungssituation genau studiert und das Vokabular und die Grammatikabdeckung darauf abgestimmt werden. Entscheidenden Einfluss auf die verwendeten Sprachmuster des Benutzers beim Dialog mit einem SDS haben die Systemprompts. Diese müssen deshalb ebenfalls sehr sorgfältig entworfen werden.

- **Qualität der Sprachausgabe**

Von guten SDS erwarten die Benutzer, dass die Sprachausgabe klar verständlich, mit natürlicher Prosodie und mit einer angenehmen Stimme und in angemessenem Tempo erfolgt.

- **Ausgabe adäquater Information**

Die vom SDS ausgegebenen Informationen sollten nicht nur korrekt und relevant sein, sondern auch richtig in der Menge (weder zu knapp noch zu viel). Es ist auch zu beachten, dass die Stimme der Sprachausgabe die vom Benutzer wahrgenommene **Persona** des gesamten Systems bestimmt und deshalb bewusst anhand der Anwendungsdomäne und des Zielpublikums ausgewählt werden sollte (vgl. Cohen et al. 2004).

- **Angemessenes Feedback**

Das SDS sollte dem Benutzer explizit oder implizit bestätigen, dass es die eingegebenen Informationen richtig verstanden hat. Zudem sollte es den Benutzer informieren, was es gerade macht bzw. welche Aktionen es ausgeführt hat. Bei heiklen Aktionen ist allenfalls eine vorgängige Information mit Abbruchmöglichkeit vorzusehen.

- **Angemessenheit der Dialoginitiative**

Wie gut ist die Dialoginitiative (System- oder Benutzer-initiiert) und deren Wechsel an den Aufgaben- und Benutzerkontext angepasst?

- **Natürlichkeit des Sprachdialogs**

Ist der Dialog aus Sicht des Benutzers logisch, aufgabengerecht und erwartungsgemäß strukturiert?

- **Abdeckungsgrad der Aufgabenstellung bzw. der Anwendungsdomäne**

Erfüllt das System die Erwartungen des Benutzers bezüglich Serviceumfang und Servicequalität für die gegebene Aufgabenstellung und Anwendungsdomäne?

- **Fähigkeit des Systems zur Argumentation**

Besitzt das System genügend Wissen über den Kontext, die Anwendung und die Welt, um als hilfsreicher Gesprächspartner des Benutzers in der gegebenen Dialogsituation agieren zu können?

- **Hilfestellung bei der Interaktion**

Bekommt der Benutzer während dem Dialog genügend und die richtige Hilfestellung (unaufgefordert, auf Wunsch oder bei Problemen), damit er das Gefühl bekommt und behält, die volle Kontrolle über den Dialog zu haben?

- **Adäquate Fehlerbehandlung**

Das System muss sowohl Fehler des Systems (Erkennungsfehler, Falscherkennungen, Zurückweisungen) als auch des Benutzers (Falscheingaben, Missverständnisse) unterscheiden und situationsgerecht behandeln können.

- **Anpassungsfähigkeit an unterschiedliche Benutzergruppe**

Kann sich das System an verschiedene Benutzergruppen (System-/Anwendungsexperten, System-/Anwendungsunerfahrene) angemessen anpassen?

- **Angemessenes Angebot an Modalitäten**

Ist Sprachein- bzw. -ausgabe, ev. in Kombination mit anderen Modalitäten, die richtige Modalität für den Benutzer, um die anstehende Aufgabe optimal zu erledigen?

Usability und Benutzerzufriedenheit

Usability und Benutzerzufriedenheit (user satisfaction) sind nicht dasselbe, obwohl letzteres direkt von ersterem abhängt. Während Usability vor allem beschreibt, wie gut das System benutzbar ist, hängt die Benutzerzufriedenheit auch von vielen externen Faktoren ab wie Verfügbarkeit des Systems, Kosten, konkurrierende Technologien (Dybkaer et al. 2004, S. 47). Für kommerzielle Systeme ist letztendlich die Benutzerzufriedenheit für den Erfolg entscheidend. Möller (2002) setzt ausgehend von den Quality-of-Service-Anforderungen für telefonbasierte Sprachdialogsysteme verschiedenste Usability- und andere Qualitätskriterien in Bezug zur Benutzerzufriedenheit.

Die Benutzerzufriedenheit wird häufig mit Hilfe von Fragebogen oder Interviews zu ermitteln versucht. Dabei werden den Benutzern, meist per Fragebogen, Aussagen zum System präsentiert, zu denen sie den Grad ihrer Zustimmung auf einer sogenannten **Likert-Skala** angeben müssen (Oppenheim 1966). Diese bietet typischerweise 5, 7 oder 9 Stufen zur Auswahl, die von „überhaupt nicht einverstanden“ bis „vollkommen einverstanden“ reichen.

Die Entwicklung von Fragebogen für Benutzerbefragungen ist ein eigenes Forschungsgebiet mit langer Tradition. Bevor ein Fragebogen für eine bestimmte Benutzerbefragung verwendet wird, sollte wie bei den anderen Qualitätsmaßen die **Validität** und **Reliabilität** des Fragebogens abgeklärt werden. Die Validität geht der Frage nach, ob die gestellten Fragen im Fragebogen bezüglich der gegebenen Problemstellung, hier der Evaluierung der Benutzerzufriedenheit, überhaupt relevant sind. Die Reliabilität sagt aus, wie reproduzierbar die Testresultate mit dem gegebenen Fragebogen sind.

Für die Benutzerzufriedenheitsbefragungen im Zusammenhang mit Benutzerschnittstellen wurden verschiedene Fragebogen entwickelt, z. B. das Software-Usability-Measurements-Inventory¹¹ oder der QUIS-Fragebogen von der University of Maryland (Chin et al. 1988). Eine detaillierte Diskussion dieser Fragebogen im Zusammenhang mit Sprachdialogsystemen ist in Larsen (2003) zu finden.

Das **PARADISE**-Framework (Walker et al. 2000) versucht, aus quantitativen Qualitäts-Messungen Aussagen über die zukünftige Benutzerzufriedenheit abzuleiten. Dabei wird postuliert, dass die Benutzerzufriedenheit von den zwei Hauptfaktoren Task-Erfolgsrate und Dialog-Kosten beeinflusst wird. Die Task-Erfolgsrate wird mit Hilfe der Kappa-Statistik ausgedrückt, wobei κ berechnet wird aus:

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)} \quad (6.5)$$

$P(A)$ ist dabei der Prozentsatz der vom System erfolgreich erkannten Ziele oder Subziele, während $P(E)$ den Prozentsatz der (Sub-)Ziele darstellt, der durch Zufall richtig erkannt worden wäre. $\kappa = 0$ heißt also, dass das System nicht mehr (Sub-)Ziele des Benutzers erkannt hat, als man durch Zufall erraten hätte. Ein κ von 1 erreicht ein System, wenn es alle eingegebenen (Sub-)Ziele richtig erkannt hat.

Die Dialogkostenfaktoren werden aufgeteilt in Effizienz- (z. B. Anzahl Äußerungen oder Dialogdauer) und Qualitätsfaktoren (z. B. Erkennungsrate, Anzahl Timeouts, Zurückweisungen, Korrekturdialoge).

Im PARADISE-Framework wird sodann die Benutzerzufriedenheit als Performanzfunktion aufgefasst und in Beziehung zu den gemessenen Usability-Kriterien gesetzt. Die aktuelle Benutzerzufriedenheit in Bezug auf das System wird zuerst mit Hilfe von Fragebogen gemessen. Sodann wird die Benutzerzufriedenheit als lineare Funktion von gewichteten Faktoren, nämlich den gemessenen Usability-Kriterien, aufgefasst. Die Gewichtung der einzelnen Faktoren wird mit Hilfe der Multivariaten Linearen Regression (MLR) bestimmt. Die errechnete Gewichtung jedes Faktors ist ein Maß für den Einfluss, den das entsprechende Usability-Kriterium auf die Benutzerzufriedenheit hat. Mit der so errechneten Performanzfunktion für die Benutzerzufriedenheit kann aus der Messung der einzelnen Usability-Kriterien für ein konkretes Sprachdialogsystem die zu erwartende Benutzerzufriedenheit im Voraus abgeschätzt werden.

¹¹<http://www.sumi.ie>

Das PARADISE-Framework wurde bereits für die Evaluation verschiedenster Sprachdialogsysteme (Walker et al. 1997; Walker et al. 2000; Larsen 2003) eingesetzt und konnte bisher ca. 40% der Varianz in der Performanzfunktion erklären. Nicht unerwartet war dabei die Task-Erfolgsrate bzw. κ der wichtigste Faktor für die Benutzerzufriedenheit.

Im Zusammenhang mit dem Projekt SmartKom wurde das PARADISE-Framework erweitert, um auch multimodale task-orientierte Sprachdialogsysteme modellieren zu können (Beringer et al. 2002).

6.2.3 Informationssuchsysteme

Die Evaluation von Informationssuchsystemen (auch **Information-Retrieval-Systeme** oder IR-Systeme genannt) hat bereits eine lange Tradition. Die für IR-Systeme verwendeten Evaluationsverfahren wurden bereits im obigen Abschnitt 6.1.2 beschrieben. Um verschiedene Ansätze und Systeme zu vergleichen, aber auch um die Kosten zu teilen, werden heute Evaluationen in größer angelegten Kampagnen durchgeführt, wie z. B. TREC¹² (USA), CLEF¹³ (Europa), NTCIR¹⁴ (Japan und östl. Asien) oder FIRE¹⁵ (Indien).

6.2.4 Sprachsynthesesysteme

Für die Evaluation von Sprachsynthesesystemen sind vor allem drei, relativ schwierig zu testende Aspekte von Bedeutung (vgl. Campbell 2008):

- Verständlichkeit
- Natürlichkeit
- Attraktivität (Likeability)

In Bezug auf die Verständlichkeit bei der Sprachsynthese weist Campbell (2008, S. 40) darauf hin, dass diese vor allem aufgabenorientiert, z. B. in Umgebungen getestet werden sollte, in denen Hintergrundgeräusche die Verständlichkeit beeinträchtigen können. Einen Hinweis auf die Verständlichkeit der Sprachsynthese kann in aufgabenorientierten Evaluationen außerdem das Messen von Zeitverzögerungen in den Antworten der Benutzer geben. Allerdings sind die hier genannten Methoden in Bezug auf die Sprachsynthese bisher eher die Ausnahme, in den meisten Fällen wird im Labor unter idealen Bedingungen evaluiert.

Auch die Natürlichkeit der Sprachsynthese ist ein Konzept, das ein Hörer zwar fast immer intuitiv erfassen kann, das aber schwer quantifizierbar ist. Daher gibt es in diesem Bereich noch zahlreiche offene Forschungsfragen.

Unter **Likeability** wird die Angemessenheit der Stimme in Bezug auf die Funktion des Systems und in Bezug auf die Vorstellung, die der Benutzer (vgl. Campbell 2008, S. 41) von der dazugehörigen Firma oder dem Produkt hat.

¹²<http://trec.nist.gov>

¹³<http://www.trebleclef.eu>

¹⁴<http://research.nii.ac.jp/ntcir>

¹⁵<http://www.isical.ac.in/~clia>

Wie bei der Natürlichkeit der Sprachsynthese kann Likeability mit quantitativen Methoden nicht vollständig erfasst werden.

Das ECESS-Konsortium (**ECESS – European Center of Excellence for Speech Synthesis**) hat sich zum Ziel gesetzt, die Entwicklung in der Textverarbeitung, Prosodie-Verarbeitung und der Sprachsynthese zu unterstützen und verwendet dafür als ein wichtiges Element die Evaluation. Hierzu wurde eine experimentelle Plattform eingerichtet, mit deren Hilfe web-basierte Evaluationen ortsunabhängig durchgeführt werden können (**RES – Remote Evaluation System**, vgl. Höge et al. 2008). In einer der ersten mit Hilfe von RES vorgenommenen Evaluationen wurden Module zur Textverarbeitung u.a. im Hinblick auf die folgenden Aspekte getestet:

- Normalisierung von Wörtern, die nicht dem Sprachstandard entsprechen, z. B. Abkürzungen wie Mr. für Mister
- Auffinden von Satzgrenzen

Manche Bereiche der akustischen Sprachverarbeitung werden derzeit noch traditionell evaluiert (vgl. auch Natürlichkeit und Attraktivität), d. h. Trainings- und Testdaten sowie ein Evaluationsskript werden an die evaluierenden Forschenden geschickt, damit sie die Evaluation selbst durchführen und über die erzielten Ergebnisse berichten können, aber es ist geplant, die *Remote Evaluation* auch für diese Bereiche zu entwickeln.

6.2.5 Maschinelle Übersetzung

Bei der Evaluation von Systemen zur Maschinellen Übersetzung (auch **MÜ** oder **MT** genannt, vgl. 5.7), ergibt sich neben der oben ausgeführten Evaluation sprachverarbeitender Systeme das zusätzliche Problem der Evaluation der eigentlichen Übersetzung. Wie im Unterkapitel 5.7 ausgeführt, ist eine 1:1 Übersetzung nur sehr selten möglich, bei stark unterschiedlichen Sprachen oder kulturellen und geographischen Gegebenheiten muss in der Übersetzung sogar gegenüber dem Ausgangstext verändert oder ergänzt werden (z. B. wenn die Beschreibung der Inbetriebnahme eines Kühlschrances, der in einer Region mit normalen Temperaturen hergestellt wird, für Benutzer übersetzt wird, die in einer Region mit sehr niedrigen Temperaturen leben). Für den Prozess der Übersetzungsevaluation ergeben sich daher zusätzliche Forschungsfragen (vgl. z. B. Williams 2004). Grob umschrieben ist die Evaluation der Qualität einer Übersetzung von zahlreichen Faktoren abhängig (Zielgruppe, Qualität des Ausgangstextes, betroffenes Sprachpaar, zentrale Funktion des Ausgangstextes, erwünschte Funktion des Zieltextes etc.) und damit eigentlich grundsätzlich abhängig vom Kontext, in dem die Übersetzung erstellt wird. Gleichzeitig kann die Übersetzungsqualität auch abhängig von dem Zweck, zu dem eine Übersetzung erstellt wird, unterschiedlich bewertet werden. Wenn dem Benutzer eine Sprache völlig unbekannt ist, kann evtl. eine maschinell erzeugte Rohübersetzung, die syntaktisch und morphologisch nicht zielsprachlich adäquat ist, hilfreich sein, nicht aber als Ersatz für eine hochwertige Humanübersetzung verwendet werden. Verschiedene Initiativen widmen sich dem Problem der Evaluation von MT:

In der FEMTI-Initiative (King et al. 2003) wird Expertise im Bereich MT-Evaluation zusammengestellt und ausgewertet. Dabei wird versucht, die Qualitätsmerkmale von MT-Systemen einerseits und deren Benutzungskontexte andererseits in einem ersten Schritt zu klassieren und in einem zweiten Schritt zueinander in Beziehung zu setzen.

Die bereits in Abschnitt 6.1.2 erwähnte NIST-Initiative (**NIST – National Institute of Standards and Technology**) ist eine Initiative der amerikanischen Regierung mit dem erklärten Ziel,

to support machine translation (MT) research and help advance the state-of-the-art in machine translation technology (NIST 2008, S. 1)

Es wird hervorgehoben, dass es im Moment keine einzelne Messmethode gibt, die alle Aspekte eines MT-Systems erfassen kann. Aus diesem Grund sind Messmethoden ebenfalls Thema der Forschung im Bereich MT. Im Rahmen der NIST Initiative werden regelmäßig verschiedene Reihen von Evaluationen durchgeführt, deren Ergebnisse für weitere Forschung im Bereich verwendet werden. In NIST (2008) werden z. B. maschinell erzeugte Übersetzungen mit denjenigen verglichen, die von menschlichen Übersetzern angefertigt werden und als Messmethode vor allem verschiedene Varianten von BLEU eingesetzt. BLEU (vgl. Papineni et al. 2001) ist eine Messmethode, bei der die Übereinstimmung von Textsequenzen einer maschinell erzeugten Übersetzung mit einer entsprechenden hochqualitativen Humanübersetzung verglichen wird. Dabei wird die Qualität der maschinell erzeugten Übersetzung umso höher bewertet, je mehr mit der Humanübersetzung übereinstimmende Sequenzen in ihr enthalten sind. Es vererben sich aber somit auch die oben beschriebenen Probleme der Übersetzungsevaluation auf diese Messmethode, denn es ist – wie erwähnt – nicht immer eindeutig festzulegen, welches die adäquate Referenzübersetzung ist oder ob eine völlig von der Humanübersetzung abweichende Formulierung nicht ebenfalls eine gute Übersetzung repräsentiert. Allerdings erscheint es bei der Bearbeitung von Massendaten derzeit nicht anders möglich, als mit Referenztexten zu arbeiten.

Maschinelles Dolmetschen

Noch komplexer wird die Evaluation von MT, wenn es um Dolmetschsysteme geht. Wie Jekat und Tessiore 2000 ausführen, ist eine Wort-für-Wort Übertragung keine adäquate Strategie für maschinelles (und Human-) Dolmetschen, bei dem im Gegensatz zur schriftlichen Übersetzung gesprochene Sprache in gesprochene Sprache übertragen wird. Wichtig beim maschinellen und menschlichen Dolmetschen ist der Erhalt der Funktion einer quellsprachlichen Eingabe durch die Ausgabe, wobei der Fortgang des Gesprächs nicht durch allzu lange Wartezeiten behindert werden sollte. Diese Wartezeiten werden z. B. durch gestörten Input (z. B. unvollständige Sätze, Abbrüche etc., die in gesprochener Sprache häufig vorkommen) und/oder die Suche nach semantisch und syntaktisch perfekter Ausgabe ausgelöst, aber auch durch die Länge der Eingabe. Im Folgenden wird die Evaluation des maschinellen Dolmetschsystems VERBMOBIL (Wahlster

2000) vorgestellt, weil VERBMOBIL eines der wichtigsten deutschen Dolmetsch-Projekte der letzten Jahre repräsentiert und u.a. aufsetzend auf dieser Forschung weitere Dolmetschsysteme entwickelt werden, die vorwiegend mit statistischen Methoden arbeiten (Waibel und Fügen 2008, vgl. auch Unterkapitel 5.7). In der Evaluation von VERBMOBIL werden die semantischen und syntaktischen Aspekte der Ein- und Ausgabe getrennt von der Translationsqualität bewertet. Eine Dolmetschleistung wird auch dann als einigermaßen erfolgreich (*intermediate*) bewertet, wenn sie zwar Fehler enthält, die Fortführung des Gesprächs aber gesichert ist. So wird die Übersetzungsleistung abhängig vom Dialogfortgang bewertet und zu folgenden anderen Größen in Beziehung gesetzt:

- Konfiguration des Systems,
- Leistung der Spracherkennung (s. dort),
- Linguistische Merkmale der Eingabe,
- Linguistische Merkmale der Ausgabe.

Ein weiteres Maß für die Bewertung der Translationsleistung ist die **Dialogerfolgsrate**: Jeder Dialog enthält bestimmte Themen, die von den Teilnehmenden angesprochen werden. In der Evaluation wird festgehalten, ob ein Thema erfolgreich behandelt worden ist oder nicht. Die hier angewendete Wertemenge der Translations- oder Dolmetschleistung ist { *gut (good)*, *mittel (intermediate)*, *schlecht (bad)* } (vgl. auch Jekat et al. 1999). Die Beziehungen zwischen der Translationsleistung und den übrigen genannten Faktoren werden in Prozentzahlen ausgedrückt.

Die Evaluation von VERBMOBIL erfolgt in drei Phasen: **Benutzerphase**, **Vorbereitungsphase** und **Validierungsphase**. In der Benutzerphase werden Tests durchgeführt, deren Ergebnisse in der Vorbereitungsphase so aufbereitet werden, dass in der Validierungsphase eine Messung der Eigenschaften stattfinden kann. In der **Validierungsphase** wird zur Auswertung der Dialoge das Evaluationswerkzeug GET (Graphical Evaluation Tool, Jekat und Tessiore 2000) verwendet. Mit GET kann eine Bewertung der Translationsleistung vorgenommen werden. Außerdem bietet GET die Möglichkeit, Einzelaspekte der Translationsleistungen statistisch zueinander in Beziehung zu setzen (etwa die Gewichtung des Faktors „Übersetzungsqualität“ im Verhältnis zum Faktor „Fortsetzung der Kommunikation“).

An der Darstellung wird bereits deutlich, dass diese Form der Evaluation sehr aufwändig und teuer ist und daher weniger für Massendaten geeignet ist oder für kommerzielle Systeme Anwendung finden wird. Gleichzeitig demonstriert aber die hier vorgestellte Evaluation die Komplexität der MT und ihrer Bewertung und zeigt ungelöste Probleme im computerlinguistischen Forschungsbereich Evaluation von MT auf.

6.2.6 Fazit

Evaluationen spielen bei der Entwicklung und dem Vergleich von Sprachverarbeitungssystemen eine immer wichtigere Rolle. Das Konzept der Evaluation

muss dabei an das zu evaluierende System und die Zielsetzung der Evaluation sorgfältig angepasst werden. Im Forschungsbereich stellen die regelmäßig organisierten Evaluationskampagnen eine wesentliche Triebfeder für den Fortschritt in der Sprachtechnologie dar. Für die Beurteilung der Benutzungsqualität von kommerziellen Systemen für einen konkreten Anwendungskontext ist es aber nach wie vor unabdingbar, die Systeme im ihrem realen Anwendungskontext, d. h. mit realen Benutzern, die ihre realen Aufgaben mit Hilfe der Systeme erledigen, zu evaluieren.

6.2.7 Literaturhinweise

Eine gute Übersicht über die Evaluation von verschiedenen sprachverarbeitenden Systemen bietet Dybkjaer et al. (2008).

Literaturverzeichnis

- Abeillé, A. (Hrsg.) (2003). *Treebanks: Building and Using Parsed Corpora*. Kluwer Academic Publishers.
- Abney, S. (1991). Parsing by chunks. In R. C. Berwick, S. P. Abney und C. Tenney (Hrsg.), *Principle-Based Parsing: Computation and Psycholinguistics*, S. 257–278. Dordrecht: Kluwer Academic Publishers.
- Abney, S. (1996). Part-of-speech tagging and partial parsing. In K. Church, S. Young und G. Bloothooft (Hrsg.), *Corpus-Based Methods in Language and Speech*. Dordrecht: Kluwer Academic Publishers.
- Adjukiewicz, K. (1935). Die syntaktische Konnektivität. *Studia Philosophica* 1, 1–27.
- Adler, S., A. Berglund, J. Caruso, S. Deach, T. Graham, P. Grossi, E. Gutentag, A. Milowski, S. Parnell, J. Richman und S. Zilles (2001). Extensible Stylesheet Language (XSL) 1.0. Technische Spezifikation, W3C.
- Agichtein, E. und L. Gravano (2000). *Snowball*: extracting relations from large plain-text collections. In *ACM DL*, S. 85–94.
- Ahn, D., V. Jijkoun, G. Mishne, K. Müller, M. de Rijke und S. Schlobach (2004). Using Wikipedia at the TREC QA Track. In *Proceedings of TREC 2004*.
- Aho, A. V. und M. J. Corasick (1975). Fast pattern matching: An aid to bibliographic search. *Communications of the ACM* 18(6), 333–340.
- Aho, A. V. und J. D. Ullman (1972). *The Theory of Parsing, Translation, and Compiling. Volume I: Parsing*. Series in Automatic Computation.
- Alexandersson, J., T. Becker und N. Pfleger (2006). *SmartKom - Foundations of Multimodal Dialogue Systems*, Chapter Overlay, S. 255–268. Heidelberg: Springer.
- Alexandersson, J., B. Buschbeck-Wolf, T. Fujinami, S. Koch, E. Maier, E. Maier, N. Reithinger, B. Schmitz und B. Schmitz (1998). Dialogue Acts in VERBMOBIL-2. verbmobil-report 226. Technischer Bericht, DFKI Saarbrücken, Universität Stuttgart, Technische Universität Berlin, Universität des Saarlandes. Second Edition.
- Allen, J. (1995). *Natural Language Understanding*. The Benjamin/Cummings Series in Computer Science. Menlo Park, CA: Benjamin Cummings.
- Alshawi, H. (Hrsg.) (1992). *The Core Language System*. Cambridge/Mass.: MIT Press.
- Alshawi, H. und R. Crouch (1992). Monotonic semantic interpretation. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics (ACL'92)*, Newark, Delaware, S. 32–39.
- Anderson, A., M. Bader, E. Bard, E. Boyle, G. Doherty, S. Garrod, S. Isard, J. Kowtko, J. McAllister, J. Miller, C. Sotillo, H. Thompson und R. Weinert (1991). The HCRC MapTask Corpus. *Language and Speech* 34(4), 351–366.
- Androultsopoulos, I., J. Oberlander und V. Karkaletsis (2007). Source authoring for multilingual generation of personalised object descriptions. *Natural Language Engineering* 13(3), 191–233.
- Angell, R. C., G. E. Freund und P. Willett (1983). Automatic spelling correction using a trigram similarity measure. *Information Processing and Management* 19(4), 255–261.
- Ankolekar, A., P. Buitelaar, P. Cimiano, P. Hitzler, M. Kiesel, M. Krötzsch, H. Lewen, G. Neumann, M. Sintek, T. Tserendorj und R. Studer (2006). Smartweb: Mobile access to the semantic web. In *Proc. of the Demo Session at the International Semantic Web Conference*. Athens GA, USA, Nov.
- Appelt, D. und D. Israel (1997). *Building information extraction systems*. Washington: Tutorial during the 5th ANLP. <http://www.ai.sri.com/~appelt/ie-tutorial/>.
- Apresjan, J. (1973). Regular polysemy. *Linguistics* 142, 5–32.

- Artiles, J., S. Sekine und J. Gonzalo (2007). The semeval-2007 weps evaluation: Establishing a benchmark for the web people search task. In *the Workshop on Semantic Evaluation at ACL-2007, Prague, Czech.*
- Artstein, R. und M. Poesio (2008). Inter-coder agreement for computational linguistics. *Computational Linguistics 34*(4), 555–596.
- Asahara, M. und Y. Matsumoto (2003). Japanese named entity extraction with redundant morphological analysis. In *HLT-NAACL*.
- Ashby, M. und J. Maidment (2005). *Introducing Phonetic Science*. Cambridge: Cambridge University Press.
- Azari, D., E. Horvitz, S. Dumais und E. Brill (2003). A decision making perspective on web question answering. In *UAI*.
- Baader, F., D. Calvanese, D. L. McGuinness, D. Nardi und P. F. Patel-Schneider (2003). *The Description Logic Handbook: Theory, Implementation, Applications*. Cambridge University Press, Cambridge, UK.
- Baayen, H. (2001). *Word Frequency Distributions*. Dordrecht: Kluwer.
- Baayen, H. (2008). *Analyzing Linguistic Data: A Practical Introduction to Statistics*. Cambridge: Cambridge University Press.
- Backofen, R., T. Becker, J. Calder, J. Capstick, L. Dini, J. Dörre, G. Erbach, D. Estival, S. Manandhar, A.-M. Mineur, G. van Noord, S. Oepen und H. Uszkoreit (1996). The eagles formalisms working group final report. Technischer Bericht, DFKI, Saarbrücken.
- Baeza-Yates, R. und B. Ribeiro-Neto (1999). *Modern Information Retrieval*. Wokingham, UK: Addison Wesley.
- Bagga, A. und B. Baldwin (1998). Algorithms for scoring coreference chains. In *Proceedings of the 1st International Conference on Language Resources and Evaluation*, Granada, Spain, 28–30 May 1998, S. 563–566.
- Baker, C. F., C. J. Fillmore und B. Cronin (2003). The Structure of the FrameNet Data Base. *International Journal of Lexicography 16*(3), 281–296.
- Baker, J. (1975). The dragon system - an overview. *IEEE Transactions on Acoustics, Speech, Signal Processing 23*(1), 24–29.
- Baldwin, B. (1997). CogNIAC: High precision coreference with limited knowledge and linguistic resources. In *Proceedings of the ACL Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Text*, Madrid, Spain, July 1997, S. 38–45.
- Balzert, H. (1998). *Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Heidelberg: Spektrum.
- Banko, M., M. J. Cafarella, S. Soderland, M. Broadhead und O. Etzioni (2007). Open information extraction from the web. In *IJCAI*, S. 2670–2676.
- Bar-Hillel, Y. (1953). A quasi-arithmetical notation for syntactic description. *Language 29*, 47–58.
- Bar-Hillel, Y. (1954). Logical syntax and semantics. *Language 30*, 230–237.
- Bar-Hillel, Y. (1960). The present status of automatic translation of languages. In *Advances in Computers*, S. 91–163.
- Bar-Hillel, Y. und E. Shamir (1960). Finite-state-languages: Formal representations and adequacy problems. *Bulletin of the Research Council of Israel 8*, 155–166.
- Barendregt, H. (1992). Lambda Calculi with Types. In S. Abramsky, D. Gabbay und T. Maibaum (Hrsg.), *Handbook of Logic in Computer Science*, Volume II. Oxford: Oxford University Press.
- Baroni, M. und S. Bernardini (2004). BootCaT: Bootstrapping corpora and terms from the web. In *Proceedings of Fourth International Conference on Language Resources and Evaluation*, Lisbon, Portugal, S. 1313–1316.
- Baroni, M., S. Bernardini, A. Ferraresi und E. Zanchetta. (2009). The WaCky Wide Web: A collection of very large linguistically processed Web-crawled corpora. *Journal of Language Resources and Evaluation*. Online-First: 10.2.2009, <http://www.springerlink.com>.
- Baroni, M. und A. Kilgarriff (2006). Large linguistically-processed web corpora for multiple languages. In *Proceedings of EACL-2006*, Trento, Italy.

- Bateman, J. A. (1997). Enabling Technology for Multilingual Natural Language Generation: The KPML Development Environment. *Natural Language Engineering* 3, 15–55.
- Bateman, J. A., I. Kruijff-Korbayová und G.-J. Kruijff (2005). Multilingual Resource Sharing Across Both Related and Unrelated Languages: An Implemented, Open-Source Framework for Practical Natural Language Generation. *Research on Language and Computation* 3(2), 191–219.
- Bateman, J. A., B. Magnini und F. Rinaldi (1994). The generalized Italian, German, English upper model. In *Proceedings of the ECAI94 Workshop: Comparison of Implemented Ontologies*, Amsterdam.
- Bateman, J. A. und C. L. Paris (1989). Phrasing a Text in Terms the User Can Understand. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, Michigan, S. 1511–1517. IJCAI'89.
- Bateman, J. A. und C. L. Paris (1991). Constraining the development of lexicogrammatical resources during text generation: towards a computational instantiation of register theory. In E. Ventola (Hrsg.), *Recent Systemic and Other Views on Language*, S. 81–106. Amsterdam: Mouton.
- Bateman, J. A. und M. Zock (2003). Natural language generation. In R. Mitkov (Hrsg.), *Oxford Handbook of Computational Linguistics*, Chapter 15, S. 284–304. Oxford: Oxford University Press.
- Batori, I. und W. Lenders (Hrsg.) (1989). *Computerlinguistik / Computational Linguistics. Ein internationales Handbuch*. Berlin: de Gruyter.
- Baum, L. und J. Eagon (1967). An inequality with applications to statistical estimation for functions of markov processes and to a model for ecology. *Bulletin American Mathematic Society* 73, 360–363.
- Baum, L. und T. Petrie (1966). Statistical inference for probabilistic functions of finite state markov chains. *Annals of Mathematical Statistics* 37, 1554–1563.
- Baum, L., T. Petrie, G. Soules und N. Weiss (1972). A maximization technique occurring in the statistical analysis of probabilistic functions of markov processes. *Inequalities* 3, 1–8.
- Baum, L. und G. Sell (1968). Growth transformation for functions on manifolds. *Pacific Journal of Mathematics* 27, 211–227.
- Beaver, D. I. (1997). Presupposition. In J. van Benthem und A. ter Meulen (Hrsg.), *Handbook of Logic and Language*, S. 939–1009. Elsevier, MIT Press.
- Beesley, K. (1998). Arabic morphology using only finite-state operations. In *Proceedings of the Workshop on Computational Approaches to semitic Languages*, S. 50–57. Montréal.
- Belew, R. K. (2000). *Finding out about: a cognitive perspective on search engine technology and the WWW*. New York, NY, USA: Cambridge University Press.
- Belz, A. (2007). Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering* 14(4), 431–455.
- Bengtsson, E. und D. Roth (2008). Understanding the value of features for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, Waikiki, Honolulu, Hawaii, 25–27 October 2008, S. 294–303.
- Berger, A. L., S. A. Della Pietra und V. J. Della Pietra (1996). A maximum entropy approach to natural language processing. *Computational Linguistics* 22(1), 39–71.
- Beringer, N., U. Kartal, K. Louka, F. Schiel und U. Trk (2002). A procedure for multimodal interactive system evaluation. In *Proceedings of the LREC Workshop on Multimodal Resources and Multimodal Systems Evaluation*, Las Palmas, S. 77–80.
- Berkhin, P. (2006). A survey of clustering data mining techniques. In A. Kogan, C. Nicholas und M. Teboulle (Hrsg.), *Grouping Multidimensional Data: Recent Advances in Clustering*, S. 25–71.
- Berners-Lee, T., J. Hendler und O. Lassila (2001). The semantic web. *Scientific American* 284(5), 24–30.
- Bernsen, N. O. und L. Dybkjaer (2004). Evaluation of spoken multimodal conversation. In *ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces*, New York, NY, USA, S. 38–45. ACM.

- Biber, D. (1988). *Variation Across Speech and Writing*. Cambridge University Press.
- Bick, E. (2005). Grammar for fun: IT-based grammar learning with VISL. In P. J. Henriksen (Hrsg.), *CALL for the Nordic Languages*. Samfunds litteratur.
- Bierwisch, M. (1983). Semantische und konzeptuelle repräsentationen lexikalischer einheiten. In R. Ruzicka und W. Motsch (Hrsg.), *Untersuchungen zur Semantik*, S. 61–99. Berlin: Akademie Verlag.
- Bies, A., M. Ferguson, K. Katz und R. MacIntyre (1995). *Bracketing Guidelines for Treebank II Style Penn Treebank Project*. University of Pennsylvania.
- Bies, A. und M. Maamouri (2003). Penn Arabic Treebank guidelines. Technischer Bericht, LDC, University of Pennsylvania.
- Bikel, D. M., R. M. Schwartz und R. M. Weischedel (1999). An algorithm that learns what's in a name. *Machine Learning* 34(1-3), 211–231.
- Bird, S. (1992). Finite-state phonology in HPSG. In *Proceedings of COLING 92*, Nantes, France, S. 74–80. COLING 92.
- Bird, S. und E. Klein (1993). Enriching HPSG phonology. Research Paper EUCCS/RP-56, Centre for Cognitive Science, University of Edinburgh.
- Bird, S., E. Klein und E. Loper (2009). *Natural Language Processing with Python*. O'Reilly.
- Bird, S. und M. Liberman (1999). Annotation graphs as a framework for multidimensional linguistic data analysis. Technischer Bericht, Linguistic Data Consortium, University of Pennsylvania, Philadelphia.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- Black, E., S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini und T. Strzalkowski (1991). A procedure for quantitatively comparing the syntactic coverage of English grammars. *Speech Communication* 33(1,2), 306–311.
- Black, E., F. Jelinek, J. Lafferty, D. M. Magerman, R. Mercer und S. Roukos (1993). Towards history-based grammars: using richer models for probabilistic parsing. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, Morristown, NJ, USA, S. 31–37. Association for Computational Linguistics.
- Blackburn, P. und J. Bos (1999). Working with Discourse Representation Theory. Lecture Notes, Online verfügbar: <http://www.comsem.org>.
- Blackburn, P. und J. Bos (2005). *Representation and Inference for Natural Language*. CSLI Publications.
- Blackburn, P., J. Bos und K. Striegnitz (2006). *Learn Prolog Now!*, Volume 7 of *Texts in Computing*. College Publications.
- Blair, C. R. (1960). A program for correcting spelling errors. *Information and Control* 3, 60–67.
- BLF. URL: <https://ilt.kuleuven.be/blf/>. Stand: 02.07.2009.
- Block, H. U. (1995). Stochastisches Parsing mit Kopfgrammatiken. *Prozedurale Anforderungen an die maschinelle Sprachverarbeitung. VerbMobil Report Nr. 60*, 18–21.
- Boas, H. C. (2005). From Theory to Practice: Frame Semantics and the Design of FrameNet. In S. Langer und D. Schnorbusch (Hrsg.), *Semantik im Lexikon*, S. 129–160. Tübingen: Narr.
- Bod, R. (1995). *Enriching Linguistics with Statistics: Performance Models of Natural Language*. Ph. D. thesis, Department of Computational Linguistics, Universiteit van Amsterdam.
- Boersma, P. (1999). Praat! Technischer Bericht, Institute of Phonetic Sciences, University of Amsterdam, <http://www.praat.org>.
- Boguraev, B. und T. Briscoe (Hrsg.) (1989). *Computational Lexicography for Natural Language Processing*. London/New York: Longman.
- Boitet, C., G. Pierre und M. Quézel-Ambrunaz (1978). Manipulation d'arborescences et parallélisme: le système robra. In *Proc. 7th International Conference on Computational Linguistics, Coling-1978*.

- Boros, M., W. Eckert, F. Gallwitz, G. Hanrieder und H. Niemann (1996). Towards understanding spontaneous speech: Word accuracy vs. concept accuracy. In *Proceedings of the Fourth International Conference on Spoken Language Processing , ICSLP 96*, S. 1009–1012.
- Borthwick, A., J. Sterling, E. Agichtein und R. Grishman (1998). Nyu: Description of the mene named entity system as used in muc-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*.
- Bos, J. (1995). Predicate logic unplugged. In *Proceedings of the 10th Amsterdam Colloquium*, Amsterdam, Holland, S. 133–142. ILLC/Department of Philosophy, University of Amsterdam.
- Bos, J. (2002). *Underspecification and Resolution in Discourse Semantics*. Ph. D. thesis, Universität des Saarlandes.
- Böttcher, M. (1996). *Unifikation disjunktiver Attributterme*. Sankt Augustin: GMD Infix.
- Bouchard, A., P. Liang, T. Griffiths und D. Klein (2007). A probabilistic approach to diachronic phonology. In *Proceedings of Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning*, Prague, Czech Republic, S. 887–896.
- Bouillon, P. und F. Busa (Hrsg.) (2001). *The Language of Word Meaning*. Cambridge: Cambridge University Press.
- Bouma, G., E. König und H. Uszkoreit (1988). A flexible graph-unification formalism and its application to natural-language processing. *IBM Journal of Research and Development* 32(2).
- Bouma, G. und G. van Noord (1993). Head-driven parsing for lexicalist grammars: Experimental results. In *EACL*, S. 71–78.
- Boyd, A., M. Dickinson und D. Meurers (2008). On detecting errors in dependency treebanks. *Research on Language and Computation* 6(2), 113–137.
- Brachman, R. und J. Schmolze (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science* 9, 171 – 216.
- Brachman, R. J. (1979). On the epistemological status of semantic networks. In N. Findler (Hrsg.), *Associative Networks*, S. 3–50. New York: Academic Press.
- Brachman, R. J. und H. J. Levesque (1985). *Readings in Knowledge Representation*. Los Altos, CA: Morgan Kaufmann.
- Brachman, R. J. und H. J. Levesque (2004). *Knowledge Representation and Reasoning*. Morgan Kaufmann.
- Brachman, R. J., D. L. McGuiness, P. F. Patel-Schneider und L. A. Resnick (1990). Living with CLASSIC: when and how to use a KL-ONE-like language. In J. Sowa (Hrsg.), *Principles of semantic networks*. San Mateo, US: Morgan Kaufmann.
- Brants, S., S. Dipper, S. Hansen, W. Lezius und G. Smith (2002). The TIGER treebank. In *Proceedings of the First Workshop on Treebanks and Linguistic Theories (TLT 2002)*, Sozopol, Bulgarien, S. 24–41.
- Brants, T. (2000a). Inter-annotater agreement for a German newspaper corpus. In *Proceedings of the Second International Conference on Language Resources and Engineering (LREC)*, Volume 3, S. 1435–1439. <http://www.coli.uni-sb.de/~thorsten/publications/Brants-LREC00.ps.gz>.
- Brants, T. (2000b). Tnt - a statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference, (ANLP-2000)*, Seattle, WA.
- Brants, T. und A. Franz (2006). Web 1T 5-gram Version 1. Linguistic Data Consortium, Philadelphia.
- Brants, T. und O. Plaehn (2000). Interactive corpus annotation. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC)*, Athen, Griechenland, S. 453–459.
- Bray, T., J. Paoli, C. M. Sperberg-McQueen und E. Maler (2000). Extensible Markup Language (XML) 1.0 (Second Edition). Technische Spezifikation, W3C.

- Brennan, S. E., M. W. Friedman und C. J. Pollard (1987). A centering approach to pro-nouns. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, Stanford, Cal., 6–9 July 1987, S. 155–162.
- Bresnan, J. (1982). *The Mental Representation of Grammatical Relations*. Cambridge, MA: MIT Press.
- Bresnan, J., A. Cueni, T. Nikitina und R. Baayen (2007). Predicting the Dative Alternation. In G. Bouma, I. Kraemer und J. Zwarts (Hrsg.), *Cognitive Foundations of Interpretation*, S. 69–94. Royal Netherlands Academy of Arts and Sciences.
- Brew, C. und M. Moens (2000). Data-intensive linguistics. url = <http://citeseervx.ist.psu.edu/viewdoc/download?doi=10.1.1.15.7759&rep=re&p1&type=pdf>.
- Brickley, D. und R. Guha (2003). RDF Vocabulary Description Language 1.0: RDF Schema. Technische Spezifikation (Working Draft), W3C.
- Bridle, J. (1990). Alpha-nets: A recurrent 'neural' network architecture with a hidden markov model interpretation. *Speech Communication* 9, 83–92.
- Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics* 21(4), 543–565.
- Brill, E. und R. C. Moore (2000). An improved model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting of the ACL*, Hong Kong, S. 286–293.
- Brin, S. (1998). Extracting patterns and relations from the world wide web. In *WebDB*, S. 172–183.
- Briscoe, T. und N. Waegner (1992). Robust stochastic parsing using the inside-outside algorithm. In *AAAI Workshop on probabilistically-based NLP Techniques*.
- Broeder, D., H. Brugman, A. Russel und P. Wittenburg (2000). Browsable corpus: accessing linguistic resources the easy way. *LREC 2000 Workshop*.
- Bronstein, I. und K. Semendjajew (1987). *Taschenbuch der Mathematik*. Thun: Harri Deutsch.
- Brookshear, J. (1989). *Formal Languages, Automata, and Complexity*. Redwood City: The Benjamin/Cummings Publishing Company.
- Brown, P., J. Cocke, S. D. Pietra, V. D. Pietra, F. Jelinek, J. Lafferty, R. Mercer und P. Roossin (1990). A statistical approach to machine translation. *Computational Linguistics* 16(2), 79–85.
- Brown, P., S. D. Pietra, V. D. Pietra und R. Mercer (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics* 19(2), 263–311.
- Brusilovsky, P. (2003). Adaptive navigation support in educational hypermedia: the role of student knowledge level the case for meta-adaption. *British Journal of Educational Technology* 34(4), 487–497.
- Bucher, W. und H. Maurer (1984). *Theoretische Grundlagen der Programmiersprachen*. Mannheim: Bibliographisches Institut.
- Buitelaar, P. (1998). *CORELEX: Systematic Polysemy and Underspecification*. Ph. D. thesis, Brandeis University, Waltham, Massachusetts.
- Bunescu, R. C. (2007). *Learning from Information Extraction: From Named Entity Recognition and Disambiguation To Relation Extraction*. Ph. D. thesis, University of Texas at Austin.
- Bunescu, R. C., E. Gabrilovich und R. Mihalcea (2008). AAAI 2008 Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy. <http://lit.csci.unt.edu/~wikiai08/>.
- Bunescu, R. C., R. Ge, R. J. Kate, E. M. Marcotte, R. J. Mooney, A. K. Ramani und Y. W. Wong (2005). Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine* 33(2), 139–155.
- Bunescu, R. C. und R. J. Mooney (2005). Subsequence kernels for relation extraction. In *NIPS*.
- Bunescu, R. C. und M. Pasca (2006). Using encyclopedic knowledge for named entity disambiguation. In *EACL*.

- Bunt, H., M. Kipp, M. Maybury und W. Wahlster (2005). *Multimodal Intelligent Information Presentation*, Volume 27 of *Text, Speech and Language Technology*, Chapter Fusion and Coordination for Multimodal Interactive Information Presentation, S. 325–340. Kluwer Academic.
- Burchardt, A., K. Erk, A. Frank, A. Kowalski, S. Pado und M. Pinkal (2006). The SALSA corpus: a German corpus resource for lexical semantics. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, Genoa, Italy, S. 969–974.
- Burger, S., K. Weilhammer, F. Schiel und H. Tillmann (2000). Verbmobil Data Collection and Annotations. In W. Wahlster (Hrsg.), *Verbmobil: Foundations of Speech-to-Speech Translation*, S. 539–551. Berlin: Springer.
- Burnett, D. C., M. R. Walker und A. Hunt (2004, September). Speech synthesis markup language (SSML) version 1.0. <http://www.w3.org/TR/speech-synthesis/>.
- Busemann, S. (2000). Generierung natürlichsprachlicher Texte. In G. Goerz, C.-R. Rollinger und J. Schneeberger (Hrsg.), *Einführung in die Künstliche Intelligenz*. München und Wien: Oldenbourg Verlag.
- Busemann, S. (2005). Ten Years After: An Update on TG/2 (and Friends). In G. Wilcock, K. Jokinen, C. Mellish und E. Reiter (Hrsg.), *Proceedings of the 10th European Workshop on Natural Language Generation*, Aberdeen, S. 32–39.
- Bußmann, H. (1990). *Lexikon der Sprachwissenschaft*. Stuttgart: Alfred Kröner Verlag.
- Butt, M., T. Holloway King, M. Niño und F. Segond (2000). *A Grammar Writer's Cookbook*. Stanford: CSLI Publications.
- Byron, D. K. (2002). Resolving pronominal reference to abstract entities. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Penn., 7–12 July 2002, S. 80–87.
- Cabré Castellví, M. T., R. Estopà Bagot und J. Vivaldi Palatresi (2001). Automatic term detection: A review of current systems. In D. Bourigault, C. Jacquemin und M.-C. L'Homme (Hrsg.), *Recent Advances in Computational Terminology*, S. 53–88. Amsterdam/Philadelphia: John Benjamins.
- Cafarella, M. J., D. Downey, S. Soderland und O. Etzioni (2005). Knowitnow: Fast, scalable information extraction from the web. In *HLT/EMNLP*.
- Cahill, L. (1998). Lexicalization in Applied NLG Systems. Rags deliverable, University of Brighton.
- Calhoun, S., M. Nissim, M. Steedman und J. Brenier (2005). A framework for annotating information structure in discourse. In A. Meyers (Hrsg.), *Proceedings of the ACL '05 Workshop on Frontiers in Corpus Annotation II: Pie in the Sky*, Ann Arbor, Michigan.
- Califf, M. und R. Mooney (1999). Relational learning of pattern-match rules for information extraction. In *Proceedings of the AAAI-99*, S. 328–334.
- Callison-Burch, C., M. Osborne und P. Koehn (2006). Re-evaluating the role of BLEU in machine translation research. In *Proceedings of the Thirteenth Conference of the European Chapter of the Association for Computational Linguistics*, Trento.
- Callmeier, U. (2001). Efficient parsing with large-scale unification grammars. Diplomarbeit, Universität des Saarlandes, Informatik, Saarbrücken.
- Campbell, N. (2008). Chapter 2: Evaluation of speech synthesis. from reading machines to talking machines. In L. Dybkjaer, H. Hemsen und W. Minker (Hrsg.), *Evaluation of Text and Speech Systems*, S. 29–64. Springer:UK.
- Carberry, S. und J. Clarke (1997). Generating clinical exercises of varying difficulty. In A. Jameson, C. Paris und C. Tasso (Hrsg.), *Proceedings of the Sixth International Conference on User Modeling (UM97)*, Berlin, S. 273–275. Springer Verlag.
- Cardie, C. und K. Wagstaff (1999). Noun phrase coreference as clustering. In *Proceedings of the 1999 SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, College Park, Md., 21–22 June 1999, S. 82–89.
- Carenini, G. und J. Moore (2006). Generating and evaluating evaluative arguments. *Artificial Intelligence Journal* 170(11), 925–952.

- Carl, M. und A. Way (Hrsg.) (2003). *Recent Advances in Example-Based Machine Translation*, Volume 21 of *Text, Speech and Language Technology*. Springer.
- Carletta, J. (1996). Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics* 22(2), 249–254.
- Carletta, J., S. Evert, U. Heid, J. Kilgour, J. Robertson und H. Voormann (2003). The NITE XML toolkit: flexible annotation for multi-modal language data. *Behavior Research Methods, Instruments and Computers* 3(35), 353–363.
- Carletta, J., A. Isard, S. Isard, J. C. Kowtko, G. Doherty-Sneddo und A. H. Anderson (1997). The reliability of a dialogue structure coding scheme. *Computational Linguistics* 23, 13–31.
- Carpenter, B. (1992). *The Logic of Typed Feature Structures With Applications to Unification Grammars, Logic Programming and Constraint Resolution*. Cambridge University Press.
- Carpenter, B. und G. Penn (1994). The attribute logic engine user's guide version 2.0.1. Technical report, Carnegie Mellon University.
- Carroll, J., A. Copestake, D. Flickinger und V. Poznanski (1999). An efficient generator for (semi-)lexicalist grammars. In *Proc. of the 7th European Workshop on Natural Language Generation (ENLGW-99)*, S. 86–95.
- Carson-Berndsen, J. (1998). *Time Map Phonology: Finite State Models and Event Logics in Speech Recognition*. Dordrecht: Kluwer Academic Publishers.
- Carstensen, K.-U. (1991). Aspekte der Generierung von Wegbeschreibungen. Technischer Bericht 190, IBM Deutschland GmbH, Projekt LILOG.
- Carstensen, K.-U. (2009a). From realist to cognitivist ontologies. In S. Clematide, M. Klenner und M. Volk (Hrsg.), *Searching Answers – A Festschrift in Honour of Michael Hess*.
- Carstensen, K.-U. (2009b). *Sprachtechnologie – Ein Überblick*. <http://www.cl.uzh.ch/CL/carstens/Materialien/Sprachtechnologie.pdf>: Web publication.
- Carston, R. und S. Uchida (1998). *Relevance Theory: Applications and Implications*. Amsterdam: John Benjamins.
- Carter, D. (1997). The TreeBanker: a tool for supervised training of parsed corpora. In *Proceedings of the ACL ENVGRAM Workshop*, Madrid, Spanien.
- Cassidy, S. und J. Harrington (1996). Emu: an enhanced speech data management system. In *Proc. of SST'96*, Adelaide.
- Cavnar, W. und J. Trenkle (1994). N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, US, S. 161–175.
- Chakrabarti, S. (2002). *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan-Kauffman.
- Charniak, E., K. Knight und K. Yamada (2003). Syntax-based language models for statistical machine translation. In *Proceedings MT Summit IX*.
- Chen, B. und P. Fung (2004). Biframenet: Bilingual frame semantics resource construction by cross-lingual induction. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*.
- Chen, Q., M. Zhou und M. Li (2007). Improving query spelling correction using web search results. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL 2007)*, Prag, Tschechische Republik, S. 181–189.
- Chesley, P., B. Vincent, L. Xu und R. Srihari (2006). Using Verbs and Adjectives to Automatically Classify Blog Sentiment. Technical Report SS-06-03, AAAI Spring Symposium.
- Chin, D. (1989). KNOME: Modeling what the user knows in UC. In A. Kobsa und W. Wahlster (Hrsg.), *User Models in Dialog Systems*, S. 74–107. Berlin Heidelberg New York Tokyo: Springer Verlag, Symbolic Computation Series.
- Chin, J. P., V. A. Diehl und K. L. Norman (1988). Development of an instrument measuring user satisfaction of the human-computer interface. In *Proceedings of SIGCHI '88, ACM*, S. 213–218.

- Chomsky, N. (1957). *Syntactic Structures*. The Hague: Mouton.
- Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control* 2(2), 137–167.
- Chomsky, N. (1962). A Transformational Approach to Syntax. In Hill (Hrsg.), *Proceedings of the Third Texas Conference on Problems of Linguistic Analysis in English on May 9-12, 1958*, Texas, S. 124–158. (Reprinted in *Structure of Language*, edited by Fodor and Katz. New York: Prentice-Hall, 1964; reprinted as "Une Conception Transformationnelle de la Syntaxe." *Language* 4 (December 4, 1966): 39–80; Reprinted in *Classics in Linguistics*, edited by Hayden, Alworth and Tate, 337–71. New York: Philosophical Library, 1967).
- Chomsky, N. (1965). *Aspects of the Theory of Syntax*. Cambridge, MA: MIT Press.
- Chomsky, N. (1981). *Lectures on Government and Binding*. Dordrecht: Foris.
- Church, K. W. (1988). A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, S. 136–143.
- Church, K. W. und W. A. Gale (1991). Probability scoring for spelling correction. *Statistics and Computing* 1, 93–103.
- Cimiano, P. (2006). *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer Verlag.
- Clark, J. (1999). XSL Transformations (Version 1.0). Technische Spezifikation, W3C.
- Clark, J. und S. DeRose (1999). XML Path Language (XPath) – Version 1.0. Technische Spezifikation, W3C.
- Clarke, C. L. A., G. V. Cormack, T. R. Lynam, C. M. Li und G. L. McLearn (2001). Web reinforced question answering (multitest experiments for trec 2001). In *TREC*.
- Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20, 37–46.
- Cohen, M. H., J. P. Giangola und J. Balogh (2004). *Voice user interface design*. Boston: Addison Wesley.
- Cole, R., J. Mariani, H. Uszkoreit, G. Varile, A. Zaenen, V. Zue und A. Zampolli (Hrsg.) (1997). *Survey of the State of The Art in Human Language Technology*. Cambridge: Cambridge University Press.
- Coleman, J. S. (2005). *Introducing Speech and Language Processing*. Cambridge: Cambridge University Press.
- Colineau, N. und C. L. Paris (2007). Tailoring and the Efficiency of Information Seeking. In *Proceedings of the 11th International Conference on User Modeling (UM 2007)*, Corfu, Greece, S. 440–444.
- Collins, M. und Y. Singer (1999). Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.
- Collins, M. J. (1999). *Head-driven statistical models for natural language parsing*. Ph. D. thesis, University of Pennsylvania.
- Colmerauer, A. (1970). Les systèmes-q, un formalisme pur analyser et synthétiser des phrases sur ordinateur. Technischer Bericht, TAUM, Université Montréal.
- Conway, D. (1999). *Object oriented Perl*. Greenwich, CT: Manning Publ.
- Copestake, A. (2002). *Implementing Typed Feature Structure Grammars*. Number 110 in CSLI Lecture Notes. Stanford: Center for the Study of Language and Information.
- Copestake, A., D. Flickinger, C. Pollard und I. A. Sag (2005). Minimal Recursion Semantics – An Introduction. *Research on Language and Computation* 3, 281–332.
- Cristianini, N. und J. Shawe-Taylor (2004). *Kernel Methods for Pattern Classification*. Cambridge University Press, Cambridge, MA.
- Cruse, D. (1986). *Lexical Semantics*. Cambridge: Cambridge University Press.
- Cucerzan, S. (2007). Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of EMNLP-CoNLL 2007*, S. 708–716.

- Cucerzan, S. und E. Brill (2004). Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, Barcelona, Spanien, S. 293–300.
- Culotta, A. und J. S. Sorensen (2004). Dependency tree kernels for relation extraction. In *ACL*, S. 423–429.
- Cutler, A. und D. R. Ladd (1983). *Prosody: Models and Measurements*. Heidelberg: Springer-Verlag.
- Cziferszky, A. und S. Winter (2002). Automatisches Generieren von Wanderrouten. In J. Strobl, T. Blaschke und G. Griesebner (Hrsg.), *Angewandte Geographische Informationsverarbeitung XIV*, S. 77–86. Heidelberg: Wichmann.
- Dale, R., S. Geldof und J.-P. Prost (2005). Using natural language generation in automatic route description. *Journal of Research and Practice in Information Technology* 37(1), 89–105.
- Dale, R., H. Moisl und H. Somers (Hrsg.) (2000). *Handbook of Natural Language Processing*. Marcel Dekker Inc.
- Dale, R. und E. Reiter (1995). Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science* 19, 233–263.
- Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM* 7(3), 171–176.
- De Smedt, K., H. Horacek und M. Zock (1995). Some problems with current architectures in natural language generation. In G. Adorni und M. Zock (Hrsg.), *Trends in Natural Language Generation - An Artificial Intelligence Perspective*. Springer-Verlag.
- DeGroot, M. H. und M. J. Schervish (2002). *Probability and Statistics* (3rd Ausg.). Boston: Addison Wesley.
- Dempster, A. P., N. M. Laird und D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B* 39(1), 1–38.
- DeRose, S. J. (1988). Grammatical category disambiguation by statistical optimization. *Computational Linguistics* 14, 31–39.
- Derwojedowa, M., M. Piasecki und S. Szpakowicz (2007). Polish wordnet on a shoestring. In G. Rehm, A. Witt und L. Lemnitzer (Hrsg.), *Datenstrukturen für linguistische Ressourcen und ihre Anwendungen. Proc. der GLDV Frühjahrstagung*, Tübingen, S. 169–178. Narr.
- Dickinson, M. und D. Meurers (2005a). Detecting annotation errors in spoken language corpora. In *Proceedings of the Special Session on Treebanks for Spoken Language and Discourse at the 15th Nordic Conference of Computational Linguistics (NODALIDA-05)*, Joensuu, Finnland.
- Dickinson, M. und D. Meurers (2005b). Detecting errors in discontinuous structural annotation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, Ann Arbor, MI.
- Dietterich, T. G., R. H. Lathrop und T. Lozano-Pérez (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artif. Intell.* 89(1-2), 31–71.
- Dittrich, O. (1902). Die sprachwissenschaftliche Definition der Begriffe "Satz" und "Syntax". *Philosophische Studien* 19, 93–127.
- Dominus, M. J. (2005). *Higher-order Perl: a guide to program transformation*. Amsterdam: Elsevier.
- Dörre, J. und M. Dorna (1993). CUF - a formalism for linguistic knowledge representation. In J. Dörre (Hrsg.), *Computational Aspects of Constraint-based Linguistic Description*. Esprit Basic Research Project.
- Downey, D., S. Schoenmackers und O. Etzioni (2007). Sparse information extraction: Unsupervised language models to the rescue. In *ACL*.
- Dowty, D., R. Wall und S. Peters (1981). *Introduction to Montague Semantics*. Dordrecht: Reidel.

- Drach, E. (1937). *Grundgedanken der Deutschen Satzlehre*. Frankfurt/M.: Diesterweg.
- Draxler, C. (1999). WWWSigTranscribe – a Java Extension of the WWWTranscribe Tool-box. In *MATISSE Workshop*, London.
- Draxler, C. (2008). *Korpusbasierte Sprachverarbeitung - eine Einführung*. narr Studienbücher. Tübingen: Gunter Narr Verlag.
- Draxler, C. und K. Jänsch (2004). Speechrecorder – a universal platform independent multi-channel audiorecording software. In *Proc. of LREC*, Lisbon, S. 559–562.
- Duden (2004). *Die deutsche Rechtschreibung* (23rd Ausg.). Bibliographisches Institut & F. A. Brockhaus AG.
- Duden (2006). *Deutsches Universalwörterbuch* (6th Ausg.). Bibliographisches Institut & F. A. Brockhaus AG.
- Dumais, S. T., M. Banko, E. Brill, J. J. Lin und A. Y. Ng (2002). Web question answering: is more always better?. In *SIGIR*, S. 291–298.
- Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics* 19, 61–74.
- Dutoit, T. (1997). *An Introduction to Text-to-Speech Synthesis*. Dordrecht: Kluwer.
- Dybkaer, L., N. Bernsen, R. Carlson, L. Chase, N. Dahlbäck, K. Failenschmid, U. Heid, P. Heisterkamp, A. Jönsson, H. Kamp, I. Karlsson, J. Kuppevelt, L. Lamel, P. Paroubek und D. Williams (1998). The DISC approach to spoken language systems development and evaluation. In *Proceedings of the 1st International Conference on Language Resources and Evaluation, LREC 1998*, Granada, Spain, S. 185–189.
- Dybkaer, L. und N. O. Bernsen (2000). Usability issues in spoken dialogue systems. *Nat. Lang. Eng.* 6(3-4), 243–271.
- Dybkaer, L., N. O. Bernsen und W. Minker (2004). Evaluation and usability of multimodal spoken language dialogue systems. *Speech Communication* 43(1-2), 33–54.
- Dybkaer, L., H. Hemsen und W. Minker (2008). *Evaluation of text and speech systems*. UK: Springer.
- Earley, J. (1970). An efficient context-free parsing algorithm. *Communications of the ACM* 13(2), 94–102.
- Ebbertz, M. (2002). Web Languages. URL <http://www.netz-tipp.de/sprachen.html>.
- Ebbinghaus, H.-D., J. Flum und W. Thomas (1992). *Einführung in die mathematische Logik*. Mannheim: BI Wissenschaftsverlag.
- Echihabi, A. und D. Marcu (2003). A noisy-channel approach to question answering. In *ACL*.
- Eckert, M. und M. Strube (2000). Dialogue acts, synchronising units and anaphora resolution. *Journal of Semantics* 17(1), 51–89.
- Eichler, K., H. Hemsen, M. Löckelt, G. Neumann und N. Reithinger (2008). Interactive dynamic information extraction. In *KI*, S. 54–61.
- ELDIT. URL: <http://dev.eurac.edu:8081/MakeEldit1/Eldit.html>. Stand: 02.07.2009.
- Elhadad, M. und J. Robin (1999). Surge: A comprehensive plug-in syntactic realization component for text generation. *Computational Linguistics* 4.
- Endres-Niggemeyer, B. (2004). Automatisches Textzusammenfassen. In H. Lobin und L. Lemmitzer (Hrsg.), *Texttechnologie*, S. 410–432. Tübingen: Stauffenburg.
- Engelberg, S. und L. Lemmitzer (2008). *Lexikographie und Wörterbuchbenutzung* (3. Ausg.). Tübingen: Stauffenburg.
- Erk, K. und L. Priese (2002). *Theoretische Informatik: eine umfassende Einführung*. Berlin/New York: Springer.
- Ernesti, J. und P. Kaiser (2008). *Python: das umfassende Handbuch*. Bonn: Galileo Press.
- Esling, J. (1990). Computer Coding of the IPA: Supplementary Report. *Journal of the International Phonetic Association* 20(1).
- Etzioni, O. (2008). Machine reading at web scale. In *WSDM*.
- Etzioni, O., M. J. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld und A. Yates (2005). Unsupervised named-entity extraction from the web: An experimental study. *Artif. Intell.* 165(1), 91–134.

- Evans, R. und G. Gazdar (1989). Inference in DATR. In *Proc. 4th Conference of the European Chapter of the ACL, EACL-1989*, Kopenhagen, S. 66–71.
- Evans, R. und G. Gazdar (1996). DATR: A language for lexical knowledge representation. *Computational Linguistics* 22, 167–216.
- Evans, R., P. Piwek, L. Cahill und N. Tipper (2006). Natural language processing in CLIME, a multilingual legal advisory system. *Natural Language Engineering* 14(1), 101–132.
- Evermann, J. (2005). Towards a cognitive foundation for knowledge representation. *Information Systems Journal* 15(2), 147–178.
- Evert, S., U. Heid, B. Säuberlich, E. Debus-Gregor und W. Scholze-Stubenrecht (2004). Supporting corpus-based dictionary updating. In *Proceedings of the 11th EURALEX International Congress*, Lorient, France, S. 255–264.
- Fawcett, R. und G. Tucker (1990). Demonstration of genesys: A very large semantically based systemic functional grammar. In *Proc. of COLING-90*, S. 47–49.
- Feldt, S., K. Fagja und C. Pause (2008). *Voice Business. Ratgeber für den Einsatz von Sprachlösungen in Unternehmen*. Hannover: telepublic Verlag.
- Fellbaum, C. (Hrsg.) (1998). *WordNet: An Electronic Lexical Database*. Cambridge, Mass.: MIT Press.
- Ferraresi, A., E. Zanchetta, M. Baroni und S. Bernardini (2008). Introducing and evaluating ukWaC, a very large web-derived corpus of English. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) – Can we beat Google?*, Marrakech, Morocco.
- Figueredo, A. und G. Neumann (2008). Finding distinct answers in web snippets. In *WEBIST* (2), S. 26–33.
- Figueredo, A., G. Neumann und J. Atkinson (2008). Searching for definitions answers on the web using surface patterns. *IEEE Computer*.
- Fillmore, C. J. (1968). The Case for Case. In E. Bach und R. T. Harms (Hrsg.), *Universals in Linguistic Theory*, S. 1–88. New York: Holt, Rinehart & Winston.
- Fillmore, C. J. und S. Atkins (1992). The Semantics of RISK and its Neighbors. In A. Lehrer und E. F. Kittay (Hrsg.), *Frames, Fields and Contrasts: New Essays in Semantic and Lexical Organization*, S. 75–102. Hillsdale: Erlbaum.
- Fillmore, C. J., C. F. Baker und H. Sato (2002). Seeing Arguments through Transparent Structures. In *Proc. LREC 2002*, Gran Canaria, S. 787–791.
- Firth, J. R. (1968). A synopsis of Linguistic Theory. In *Selected Papers of J.R. Firth, 1952–1959*, S. 168–205. London: F.R. Palmer.
- Fischer, K. und J. A. Bateman (2006). Keeping the initiative: an empirically-motivated approach to predicting user-initiated dialogue contributions in HCI. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, S. 185–192. Association for Computational Linguistics.
- Fliedner, G. (2001). Überprüfung und Korrektur von Nominalkongruenz im Deutschen. Diplomarbeit, Universität des Saarlandes, Computerlinguistik, Saarbrücken. <http://www.coli.uni-saarland.de/~fliedner/publications/Fliedner0%1\Ueberpruefung.pdf>.
- Floridi, L. (2005). Is information meaningful data. *Philosophy and Phenomenological Research* 70, 351–370.
- Foata, D. und A. Fuchs (1999). *Wahrscheinlichkeitsrechnung*. Basel; Boston; Berlin: Birkhäuser.
- Fong, S. (1991). *Computational Properties of Principle-Based Grammatical Theories*. Ph. D. thesis, MIT.
- Foord, M. J. und C. Muirhead (2009). *IronPython in Action*. Manning Publ.
- Forney, G. (1973). The viterbi algorithm. *Proceedings IEEE* 61(3), 268–277.
- Foster, M. E. (2007). Associating facial displays with syntactic constituents for generation. In *Proceedings of the Linguistic Annotation Workshop*, Prague, Czech Republic, S. 25–32. Association for Computational Linguistics.
- Foth, K. (2006). Eine umfassende Constraint-Dependenz-Grammatik des Deutschen. Technischer Bericht, Universität Hamburg, Hamburg.

- Foth, K. A. und W. Menzel (2006). Hybrid parsing: using probabilistic models as predictors for a symbolic parser. In *Proc. 21st Int. Conference on Computational Linguistics and 44th Annual Meeting of the ACL, Coling-ACL-2006*, S. 321–328.
- Fourcin, A., G. Harland, W. Barry und V. Hazan (Hrsg.) (1989). *Speech Input and Output Assessment*. Chichester: Ellis Horwood.
- Francis, W. und H. Kučera (1979). Brown Corpus Manual – Manual of information to accompany A Standard Corpus of Present-Day Edited American English, for use with Digital Computers. revised edition, Brown University, <http://khnt.hit.uib.no/icame/manuals/brown>.
- Francopoulo, G., N. Bel, M. George, N. Calzolari, M. Monachini, M. Pet und C. Soria (2009). Multilingual resources for NLP in the lexical markup framework (LMF). *Language Resources and Evaluation* 43(1), 57–70. Multilingual Language Resources and Interoperability.
- Freitag, D. (1998). Information extraction from html: Application of a general learning approach. In *Proceedings of the 15th AAAI*.
- Friedl, J. E. F. und A. Oram (2006). *Mastering regular expressions* (3. ed Ausg.). Sebastopol, Ca.: O'Reilly.
- Gabrilovich, E. und S. Markovitch (2006). Overcoming the Brittleness Bottleneck using Wikipedia: Enhancing Text Categorization with Encyclopedic Knowledge. In *AAAI*, Boston, MA, S. 1301–1306.
- Galanis, D. und I. Androutsopoulos (2007). Generating Multilingual Descriptions from Linguistically Annotated OWL Ontologies: the NaturalOWL System. In *Proceedings of the 11th European Workshop on Natural Language Generation (ENLG 2007)*, Schloss Dagstuhl, Germany, S. 143–146.
- Gamut, L. (1987). *Logic, Language, and Meaning*. Chicago: University of Chicago Press.
- Gangemi, A., N. Guarino, C. Masolo und A. Oltramari (2003). Sweetening WordNet with DOLCE. *AI Magazine* 24(3), 13–24.
- Garofolo, J. und J. Fiscus (1996). NIST SPHERE - SPeech HEader REsources. Technischer Bericht, National Institute for Standards and Technology.
- Gazdar, G. (1979). *Pragmatics*. New York: Academic Press.
- Gazdar, G., E. Klein, G. Pullum und I. Sag (1985). *Generalized Phrase Structure Grammar*. Oxford: Basil Blackwell.
- Gazdar, G. und C. Mellish (1989a). *Natural Language Processing in Lisp : an Introduction to Computational Linguistics*. Wokingham, England; Reading, Mass.: Addison-Wesley.
- Gazdar, G. und C. Mellish (1989b). *Natural Language Processing in Prolog : an Introduction to Computational Linguistics*. Wokingham, England Reading, Mass.: Addison-Wesley.
- Gazdar, G. und G. Pullum (1982). Generalized phrase structure grammar: A theoretical synopsis. Technischer Bericht, Bloomington, Indiana.
- Ge, N., J. Hale und E. Charniak (1998). A statistical approach to anaphora resolution. In *Proceedings of the Sixth Workshop on Very Large Corpora*, Montréal, Canada, S. 161–170.
- Germesin, S., T. Becker und P. Poller (2008, September). Domain-specific classification methods for disfluency detection. In *Interspeech'08*.
- Geumann, A., D. Oppermann und F. Schaeffler (1997). The Conventions for Phonetic Transcription and Segmentation of German used for the Munich Verbmobil Corpus. Technischer Bericht 129, Institut für Phonetik und Sprachliche Kommunikation, Universität München.
- Geurts, B. (1999). *Presuppositions and Pronouns*. London: Elsevier.
- Gibbon, D., I. Mertins und R. Moore (2000). *Handbook of Multimodal and Spoken Dialogue Systems: Resources, Terminology and Product Evaluation*. New York: Kluwer Academic Publishers.
- Gibbon, D., R. Moore und R. Winski (1997). *Handbook of Standards and Resources for Spoken Language Systems*. Berlin: Mouton de Gruyter.

- Gibbon, D. und H. Richter (1984). *Intonation, Accent and Rhythm: Studies in Discourse Phonology*. Berlin: Mouton de Gruyter.
- Gibbon, D. und G. Strokin (1998). Zdatr 2.0 manual. Technischer Bericht, Universität Bielefeld, <http://coral.lili.uni-bielefeld.de/DATR/>.
- Glass, J., J. Polifroni, S. Seneff und V. Zue (1996). Data collection and performance evaluation of spoken dialogue systems: The MIT experience. In *In Proceedings of the International Conference on Spoken Language Processing (ICSLP 2000)*, Volume 4, Beijing, S. 1–4.
- Goldberg, E., N. Driedger und R. R. Kittredge (1994). Using natural language processing to produce weather forecasts. *IEEE Expert* 9, 45–53.
- Golding, A. R. und Y. Schabes (1996). Combining trigram-based and feature-based methods for context-sensitive spelling correction. In *Proceedings of the 34th Annual Meeting of the ACL*, Santa Cruz, CA, S. 71–78.
- Gonzalo, J., F. Verdejo und I. Chugur (1999). Using EuroWordNet in a concept-based approach to cross-language text retrieval. *Applied Artificial Intelligence* 13(7), 647–678.
- Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika* 40(3/4), 237–264.
- Görz, G., C.-R. Rollinger und J. Schneeberger (Hrsg.) (2003). *Handbuch der Künstlichen Intelligenz*. München: Oldenbourg.
- Götz, T., W. Meurers und D. Gerdemann (1997). The ConTroll manual. Technical report, Seminar für Sprachwissenschaft, Universität Tübingen.
- Götze, M., T. Weskott, C. Endriss, I. Fiedler, S. Hinterwimmer, S. Petrova, A. Schwarz, S. Skopeteas und R. Stoel (2007). Information Structure. In S. Dipper, M. Götze und S. Skopeteas (Hrsg.), *Information Structure in Cross-Linguistic Corpora*, Number 07 in Interdisciplinary Studies on Information Structure (ISIS), S. 147–187.
- Granger, S. (2002). A Bird's-eye view of learner corpus research. In S. P.-T. Sylviane Granger, Joseph Hung (Hrsg.), *Computer Learner Corpora, Second Language Acquisition and Foreign Language Teaching*, S. 3–33. Amsterdam / Philadelphia: John Benjamins.
- Grefenstette, G. und P. Tapanainen (1994). What is a word, what is a sentence? problems of tokenization. In *Proceedings of the 3rd Conference on Computational Lexicography and Text Research*, Budapest, S. 78–87.
- Grewendorf, G., F. Hamm und W. Sternefeld (1987). *Sprachliches Wissen*. Frankfurt/M.: Suhrkamp.
- Grice, H. (1957). Meaning. *Philosophical Review* 66, 377–88.
- Grice, H. P. (1975). Logic and conversation. *Syntax and Semantics* 3, 41–85.
- Gries, S. (2008). *Statistik für Sprachwissenschaftler*. Number 13 in Studienbücher zur Linguistik. Göttingen: Vandenhoeck & Ruprecht.
- Gries, S. (2009). *Quantitative corpus linguistics with R: a practical introduction*. London, New York: Routledge, Taylor & Francis Group.
- Grishman, R. und B. Sundheim (1996). Message Understanding Conference – 6: A Brief History. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING '96)*, Kopenhagen, Denmark, Europe.
- Grosz, B. J., A. K. Joshi und S. Weinstein (1995). Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics* 21(2), 203–225.
- Gruber, T. (1995). Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies* 43, 907 – 928.
- Gruber, T. (2008). Collective knowledge systems: Where the Social Web meets the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web* 6(1), 4–13.
- Guarino, N. (1995). Formal ontology, conceptual analysis and knowledge representation. *International Journal of Human and Computer Studies* 43(5-6), 625–640.
- Gulli, A. und A. Signorini (2005). The indexable web is more than 11.5 billion pages. <http://www.cs.uiowa.edu/~asignori/web-size/>: University of Iowa.

- Haghghi, A. und D. Klein (2007). Unsupervised coreference resolution in a nonparametric Bayesian model. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, Prague, Czech Republic, 23–30 June 2007, S. 848–855.
- Hajičová, E., Z. Kirschner und P. Sgall (1999). A Manual for Analytic Layer Annotation of the Prague Dependency Treebank (English translation). Technischer Bericht, ÚFAL MFF UK, Prague, Czech Republic.
- Hajič, J., A. Böhmová, E. Hajičová und B. Vidová-Hladká (2000). The Prague Dependency Treebank: A three-level annotation scenario. In A. Abeillé (Hrsg.), *Treebanks: Building and Using Parsed Corpora*. Kluwer Academic Publishers.
- Hakkani-Tur, D., H. Ji und R. Grishman (2005). Information extraction to improve cross-lingual document retrieval. In *Proc. RANLP workshop on Multi-source, Multilingual Information Extraction and Summarization*.
- Hall, T. (2000). *Phonologie: Eine Einführung*. Berlin: Walter de Gruyter.
- Halliday, M. (1994). *Introduction to Functional Grammar*. London: Edward Arnold.
- Hammond, M. (2002). *Programming for linguists : JavaTM technology for language researchers*. Oxford: Blackwell.
- Hammond, M. (2003). *Programming for linguists : Perl for language researchers*. Malden, MA: Blackwell.
- Hamp, B. und H. Feldweg (1997). GermaNet - a Lexical-Semantic Net for German. In P. Vossen, N. Calzolari, G. Adriaens, A. Sanfilippo und Y. Wilks (Hrsg.), *Proceedings of the ACL/EACL-97 workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, S. 9–15. Madrid.
- Handke, J. (1995). *The Structure of the Lexicon*. Berlin: Mouton de Gruyter.
- Harabagiu, S. M., R. C. Bunescu und S. J. Maiorano (2001). Text and knowledge mining for coreference resolution. In *Proceedings of the 2nd Conference of the North American Chapter of the Association for Computational Linguistics*, Pittsburgh, Penn., 2–7 June 2001, S. 55–62.
- Harnad, S. (1990). The symbol grounding problem. *Physica D* 42, 335 – 346.
- Harrington, J. (2009). *Acoustic Phonetics*. The Handbook of Phonetic Sciences. Blackwell.
- Hausmann, F. (1985). Lexikographie. In C. Schwarze und D. Wunderlich (Hrsg.), *Handbuch der Lexikologie*, S. 367–411. Königstein: Athenäum.
- Hausser, R. (2000). *Grundlagen der Computerlinguistik. Mensch–Maschine Kommunikation in natürlicher Sprache*. Berlin: Springer.
- Hausser, R. (2001). *Foundations of Computational Linguistics. Human-Computer Communication in Natural Language. 2nd Ed.* Berlin/New York: Springer.
- Heim, I. (1990). On the projection problem for presuppositions. In S. Davis (Hrsg.), *Pragmatics*, S. 397–405. Oxford: Oxford University Press.
- Helbig, H. (2001). *Die semantische Struktur natürlicher Sprache*. Springer-Verlag.
- Helbig, H. und C. Gnörlich (2002). Multilayered Extended Semantic Networks as a Language for Meaning Representations in NLP Systems. In *Proceedings of 6th Annual Meeting of the Gesellschaft für Semantik: Sinn und Bedeutung VI*, S. 99–113. Osnabrück.
- Henschel, R. und J. Bateman (1997). Application-driven automatic subgrammar extraction. In *Proceedings of ACL/EACL97 Workshop: “ENVGRAM: Computational Environments for grammar development and linguistic engineering*. Association for Computational Linguistics.
- Heyer, G., U. Quasthoff und T. Wittig (2006). *Text Mining: Wissensrohstoff Text*. Herdecker/Bochum: W3L-Verlag.
- Hindle, D. und M. Rooth (1993). Structural ambiguity and lexical relations. *Computational Linguistics* 19, 103–120.
- Hinrichs, E. W., J. Bartels, Y. Kawata, V. Kordoni und H. Telljohann (2000). The Tübingen Treebanks for Spoken German, English, and Japanese. In W. Wahlster (Hrsg.), *Verbmobilität: Foundations of Speech-to-Speech Translation*. Berlin: Springer.
- Hitzler, P., M. Krötzsch, S. Rudolph und Y. Sure (2007). *Semantic Web: Grundlagen*. Springer.

- Hitzler, P. und K.-U. Kühnberger (2009). The importance of being neural-symbolic – a wilde position. In B. Goertzel, P. Hitzler und M. Hutter (Hrsg.), *Artificial General Intelligence*, Second Conference on Artificial General Intelligence, AGI, Arlington, Virginia, USA.
- Hladka, B. und O. Kucera (2008). An annotated corpus outside its original context: A corpus-based exercise book. In *Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications (ACL)*, Columbus, OH, S. 36–43.
- Hobbs, J. R. (1978). Resolving pronominal references. *Lingua* 44, 311–338.
- Höge, H., Z. Kacic, B. Kotnik, M. Rojc, N. Moreau und H.-U. Hain (2008). Evaluation of modules and tools for speech synthesis – The ECESS framework. In *Proceedings of the 6th International Conference on Language Resources and Evaluation, LREC 2008*, S. o.S. Marrakech, Morocco: LREC.
- Höhle, T. (1986). Der Begriff "Mittelfeld", Anmerkungen über die Theorie der topologischen Felder. In *Akten des Siebten Internationalen Germanistenkongresses 1985*, Göttingen, S. 329–340.
- Hopcroft, J. und J. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation*. Reading, Mass: Addison-Wesley.
- Hopcroft, J. und J. Ullman (1994). *Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie*. Bonn: Addison-Wesley. Deutsche Übersetzung von Hopcroft und Ullman 1979.
- Horacek, H. (1996). On expressing metonymic relations in multiple languages. *Machine Translation* 11, 109–158.
- Horn, L. (1968). *On the semantic properties of logical operators in English*. Ph. D. thesis, UCLA.
- Hors, A. L., P. L. Hégaret, L. Wood, G. Nicol, J. Robie, M. Champion und S. Byrne (2000). Document Object Model (DOM) Level 2 Core Specification. Technische Spezifikation, W3C.
- Hovy, E. (1993). Automated discourse generation using discourse structure relations. *Artificial Intelligence* 63, 341–385.
- Huang, X., A. Acero und H.-W. Hon (2001). *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Englewood Cliffs, NJ: Prentice-Hall.
- Huffman, S. (1996). Learning information extraction patterns from examples. In Wermter, Riloff und Scheler (Hrsg.), *Connectionist, Statistical, and Symbol Approaches to Learning for Natural Language Processing*, Volume 1040 of *LNAI*, Berlin, Springer.
- Hunt, A. und S. McGlashan (2003). Speech recognition grammar specification version 1.0.
- Hunt, A. J. und A. W. Black (1996). Unit selection in a concatenative speech synthesis system using a large speech database. In *Proceedings of the IEEE International Conference on Acoustics and Speech Signal Processing (München, Germany)*, Volume 1, S. 373–376.
- Hutchins, J. (2008). *Compendium of Translation Software. directory of commercial machine translation systems and computer-aided translation support tools*. The European Association for Machine Translation. <http://www.hutchinsweb.me.uk/Compendium.htm>.
- Hutchins, J. und W. Hartmann (2002). Compendium of Translation Software. Commercial Machine Translation Systems and Computer-Aided Support Tools. *European Association for Machine Translation*. Version 1.5.
- Hutchins, W. J. (1986). *Machine Translation, Past, Present, Future*. Chichester: Ellis Horwood.
- Ide, N. und J. Véronis (1994). MULTEXT: Multilingual Text Tools and Corpora. In *COLING 94 – 15th Int. Conf. on Comp. Ling.*, Kyoto, Japan, S. 588–592. ACL.
- IPA (1999). *Handbook of the IPA*. Cambridge: Cambridge University Press.
- ISO (2001-2004). Software engineering – Product Quality, Part 1-4. In International Organization for Standardization und International Electrical Commission (Hrsg.), *ISO-IEC*. Geneva, Switzerland: ISO.

- ISO 8879 (1986). Information Processing – Text and Office Information Systems – Standard Generalized Markup Language. Internationaler Standard, ISO, Genf.
- ISO/IEC (1998). *International Standard ISO/IEC 9241-11:1998. Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability*. Genf: International Organization for Standardization, International Electrotechnical Commission.
- ISO/IEC (2001). International standard ISO/IEC 9126-1:2001. Software engineering – Product Quality, Part 1: Quality Model. In International Organization for Standardization und International Electrical Commission (Hrsg.), *ISO-IEC*. Geneva, Switzerland: ISO.
- Israel, D., M. Kameyama, M. Stickel und M. Tyson (1996). Fastus: A cascaded finite-state transducer for extracting information from natural language text. In *in Roche and Schabes (eds.) Finite State Devices for Natural Language Processing*, S. 383–406. MIT Press.
- Jacobs, J., A. von Stechow, W. Sternefeld und T. Vennemann (Hrsg.) (1993). *Syntax. Ein internationales Handbuch zeitgenössischer Forschung*. Berlin u.a.: Walter deGruyter.
- Jameson, A. (2008). Adaptive interfaces and agents. In A. Sears und J. A. Jacko (Hrsg.), *Human-computer interaction handbook: Fundamentals, Evolving Technologies and Emerging Applications* (2 Ausg.), S. 433–458. New York: Lawrence Erlbaum Associates.
- Janssen, T. (1997). Compositionality. In J. van Benthem und A. ter Meulen (Hrsg.), *Handbook of Logic and Language*, S. 417–473. Elsevier.
- Jekat, S. und L. Tessiore (2000). End-to-end evaluation of machine interpretation systems: A graphical evaluation tool. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation, LREC 2000*, S. o.S. Athens, Greece: LREC.
- Jekat, S., L. Tessiore, B. Lause und M. Glockemann (1999). End-to-End Evaluation 3/99 of VERBMOBIL 0.7. Verbmobil memo 144, University of Hamburg.
- Jekat, S. und W. v. Hahn (2000). Multilingual Verbmobil-dialogs: Experiments, data collection and data analysis. In W. Wahlster (Hrsg.), *Verbmobil: Foundations of Speech-to-Speech Translation*, S. 577–584. Berlin: Springer.
- Jelinek, F. (1976). Continuous speech recognition by statistical methods. *Proceedings of the IEEE* 64(4), 532–556.
- Jensen, K., G. E. Heidorn und S. D. Richardson (Hrsg.) (1993). *Natural Language Processing: The PLNLP Approach*. Boston, MS: Kluwer.
- Johnson, C. (1972). *Formal Aspects of Phonological Description*. The Hague: Mouton.
- Johnson, K. (2008). *Quantitative Methods in Linguistics*. Malden / Oxford / Victoria: Blackwell Publishing.
- Johnson, M. (1988). *Attribute-value Logic and the Theory of Grammar*. Number 16 in CSLI Lecture Notes. Stanford University.
- Johnson, M. (1998). PCFG models of linguistic tree representations. *Computational Linguistics* 24(4), 613–632.
- Jones, K. S. und J. Galliers (1996). *Evaluating Natural Language Processing Systems*. Berlin: Springer.
- Joshi, A. (1985). *Tree adjoining grammars. How much context-sensitivity is required to provide reasonable structural descriptions?*, S. 206–250. Cambridge: Cambridge University Press.
- Joshi, A. (1990). Phrase structure grammar. In S. Shapiro (Hrsg.), *Encyclopedia of Artificial Intelligence*. New York: John Wiley & Sons.
- Junqua, J.-C. und L.-P. Haton (1996). *Robustness in automatic speech recognition. Fundamentals and applications*. Boston, USA: Kluwer Academic Publishers.
- Jurafsky, D. und J. H. Martin (2009). *Speech and Language Processing* (Zweite Ausg.). Prentice Hall.
- Kameyama, M. (1998). Intrasentential centering: A case study. In M. Walker, A. Joshi und E. Prince (Hrsg.), *Centering Theory in Discourse*, S. 89–112. Oxford, U.K.: Oxford University Press.

- Kamp, H. (1981). A theory of truth and semantic interpretation. In J. Groenendijk, T. Janssen und M. Stokhof (Hrsg.), *Formal Methods in the Study of Language*. Amsterdam: Mathematical Centre.
- Kamp, H. und U. Reyle (1993). *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Dordrecht: Kluwer.
- Kaplan, R. M. (1973). A general syntactic processor. In R. Rustin (Hrsg.), *Natural Language Processing*, S. 193–241. New York: Algorithmics Press.
- Karttunen, L. (1973). Presuppositions of compound sentences. *Linguistic Inquiry* 4, 167–193.
- Karttunen, L. (1974). Presuppositions and linguistic context. *Theoretical Linguistics* 1, 181–94.
- Kasami, T. (1965). An efficient recognition and syntax analysis algorithm for context-free languages. Technischer Bericht, Air Force Cambridge Research Laboratory, Bedford Mass.
- Kay, M. (1973). The mind system. In R. Rustin (Hrsg.), *Natural Language Processing*, S. 155–188. New York: Algorithmics Press.
- Kay, M. (1979). Functional grammar. In *Proc. of the 5th Meeting of the Berkeley Linguistics Society*, S. 142–158.
- Kay, M. (1980). Algorithm schemata and data structures in syntactic processing. Technischer Bericht, Xerox PARC, Palo Alto.
- Kay, M. (1996). Chart generation. In *Proc. of ACL-96*, S. 200–204.
- Kehler, A. (1997). Current theories of centering for pronoun interpretation: A critical evaluation. *Computational Linguistics* 23(3), 467–475.
- Kehler, A., D. Appelt, L. Taylor und A. Simma (2004). The (non)utility of predicate-argument frequencies for pronoun interpretation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, Boston, Mass., 2–7 May 2004, S. 289–296.
- Keller, F. und M. Lapata (2003). Using the web to obtain frequencies for unseen bigrams. *Computational Linguistics* 29, 459–484.
- Kennedy, C. und B. Boguraev (1996). Anaphora in a wider context: Tracking discourse referents. In *Proceedings of the 12th European Conference on Artificial Intelligence*, Budapest, Hungary, 11–16 August 1996, S. 582–586.
- Kern, F. (1883). *Zur Methodik des deutschen Unterrichts*. Berlin.
- Kernigham, M. D., K. W. Church und W. A. Gale (1990). A spelling correction program based on a noisy channel model. In *Proceedings of 13th International Conference on Computational Linguistics (COLING 1990)*, Helsinki, S. 205–210.
- Kilgarriff, A. und P. Edmonds (Hrsg.) (2003). *Special Issue on Senseval-2*, Volume 9(1) of *Journal of Natural Language Engineering*.
- Kilgarriff, A. und M. Palmer (Hrsg.) (2000). *Computing and the Humanities: Special Issue on SENSEVAL*, Volume 34. Kluwer.
- Kilgarriff, A. und J. Rosenzweig (2000). Framework and results for English SENSEVAL. *Computers and the Humanities: Special Issue on SENSEVAL* 34(1–2), 15–48.
- Kilgarriff, A., P. Rychly, P. Smrz und D. Tugwell (2004). The Sketch Engine. In *Proceedings of the 11th EURALEX International Congress*, Lorient, France, S. 105–116.
- King, M., B. Maegard, J. Schutz und L. des Tombes (1996). EAGLES – evaluation of natural language processing systems. Technischer Bericht.
- King, M., A. Popescu-Belis und E. Hovy (2003). FEMTI: Creating and using a framework for MT evaluation. In *Proceedings of the Machine Translation Summit IX, New Orleans, LA, September 2003*. New Orleans, LA.
- King, P. (1994). An expanded logical formalism for head-driven phrase structure grammar. Arbeitspapiere des SFB 340, Universität Tübingen.

- Kipp, M., M. Neff und I. Albrecht (2007). An annotation scheme for conversational gestures: How to economically capture timing and form. *Language Resources and Evaluation* 41(3), 325–339.
- Kirch-Prinz, U. und P. Prinz (2007). *C++ lernen und professionell anwenden* (4., aktual. u. erw. Aufl. Ausg.). Heidelberg: mitp, Redline.
- Kiss, T. und J. Strunk (2002). Viewing sentence boundary detection as collocation identification. In S. Busemann (Hrsg.), *KONVENS 2002 Tagungsband*, Saarbrücken, S. 75–82.
- Kiss, T. und J. Strunk (2006). Unsupervised multilingual sentence boundary detection. *Computational Linguistics* 32(4), 485–525.
- Klein, D. und C. D. Manning (2003). Accurate unlexicalized parsing. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, Morristown, NJ, USA, S. 423–430. Association for Computational Linguistics.
- Klesen, M. und P. Gebhard (2007). Affective multimodal control of virtual characters. *International Journal of Virtual Reality* 6(4), 43–53.
- Knight, K. und S. Luk (1994). Building a large knowledge base for machine translation. In *Proceedings of American Association of Artificial Intelligence Conference (AAAI-94)*, Seattle, WA, S. 773–778.
- Knuth, D. E. (1969). *The Art of Computer Programming*. Reading, MA: Addison-Wesley.
- Knuth, D. E. (1998). *Sorting and Searching* (2nd Ausg.), Volume 3 of *The Art of Programming*. Reading, MS: Addison-Wesley.
- Kobsa, A. (2001). Generic user modeling systems. *User modelling and user-adapted interaction* 11, 49–63.
- Kobsa, A. und W. Wahlster (Hrsg.) (1989). *User Models in Dialog Systems*. Berlin Heidelberg New York Tokyo: Springer Verlag, Symbolic Computation Series.
- Koehn, P. (2005). Europarl: A Parallel Corpus for Statistical Machine Translation. In *Machine Translation Summit X*, S. 79–86.
- Koehn, P. (2009). *Statistical Machine Translation*. Cambridge University Press.
- Koehn, P. et al. (2007). Moses: open source toolkit for statistical machine translation. In *Annual meeting of the Association for Computational Linguistics: Demonstration session*, Prague, S. 177–180.
- Koehn, P., F. Och und D. Marcu (2003). Statistical phrase based translation. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT/NAACL)*.
- Kohler, K., M. Pätzold und A. Simpson (1994). Handbuch zur Segmentation und Etikettierung von Spontansprache - 2.3. Technischer Bericht, Verbmobil Technisches Dokument 16, Kiel: IPDS.
- Koller, A. (2004). *Constrained-Based And Graph-Based Resolution of Ambiguities in Natural Language*. Ph. D. thesis, Universität des Saarlandes.
- König, E. und W. Lezius (2000). A description language for syntactically annotated corpora. In *Proceedings of COLING 2000*, Saarbrücken.
- Koskenniemi, K. (1983). Two-level model for morphological analysis. In *Proc. 8th International Joint Conference on Artificial Intelligence, IJCAI-1983*, Karlsruhe, FRG, S. 683–685.
- Kotelly, B. (2003). *The Art and Business of Speech Recognition. Creating the Noble Voice*. Amsterdam: Addison Wesley.
- Krieger, H.-U. und U. Schäfer (1994). TDL - a type description language for HPSG. Technical Report RR-94-37, DFKI Saarbrücken.
- Krötzsch, M., D. Vrandecic und M. Völkel (2005). Wikipedia and the Semantic Web – the missing links. In *Proceedings of First International Wikimedia Conference – Wikimania 2005*, Frankfurt, Germany.
- Kuhn, J. und C. Rohrer (1997). Approaching ambiguity in real-life sentences – the application of an optimality theory-inspired constraint ranking in a large-scale lfg grammar. In *DFGS-CL-Jahrestagung 1997*, <http://www.ims.uni-stuttgart.de/projekte/pargram>.

- Kukich, K. (1992). Techniques for automatically correcting words in text. *ACM Computing Surveys* 24(4), 377–439.
- Kunze, C. und L. Lemnitzer (2007). *Computerlexikographie. Eine Einführung*. Tübingen: Narr.
- Kunze, C. und A. Wagner (1999). Anwendungsperspektiven des GermaNet, eines lexikalisch-semantischen Netzes für das Deutsche. In B. Schröder, A. Storrer und I. Lemberg (Hrsg.), *Probleme und Perspektiven computergestützter Lexikographie*. Tübingen: Niemeyer.
- Kwok, C. C. T., O. Etzioni und D. S. Weld (2001). Scaling question answering to the web. In *WWW*.
- Ladd, D. R. (2008). *Intonational Phonology*. Cambridge: Cambridge University Press.
- Lang, E., K.-U. Carstensen und G. Simmons (1991). *Modelling Spatial Knowledge on a Linguistic Basis*. Springer-Verlag. Lecture Notes in Artificial Intelligence No. 481.
- Langer, H. (1996). Analyse-orientierte Zugriffsstrategien für default-orientierte Vererbungslexika. In *Forschungsstelle für Artikulationsprozesse. Sprachstrukturen und Sprachprozesse*, S. 92–117. secolo Verlag Osnabrück.
- Langer, H. (2001). *Parsing-Experimente. Praxisorientierte Untersuchungen zur automatischen Analyse des Deutschen*. Frankfurt/M.: Peter Lang.
- Lapata, M. und F. Keller (2005). Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing* 2, 1–31.
- Lappin, S. und H. J. Leass (1994). An algorithm for pronominal anaphora resolution. *Computational Linguistics* 20(4), 535–561.
- Larsen, L. B. (2003). *On the usability of spoken dialogue systems*. Phdthesis, Aalborg University.
- Lassila, O. und R. R. Swick (1999). Resource Description Framework (RDF). Model and Syntax Specification. Technische Spezifikation, W3C.
- Lebert, M. (2008). Project Gutenberg (1971-2008). <http://www.gutenberg.org/etext/27045>.
- Leech, G. und E. Eyes (1997). Syntactic annotation: Treebanks. In R. Garside, G. Leech und T. McEnery (Hrsg.), *Corpus Annotation: Linguistic Information from Computer Text Corpora*, S. 34–52. New York: Addison Wesley Longman.
- Leech, G., A. Wilson et al. (1996). EAGLES Guidelines: Recommendations for the Morphosyntactic Annotation of Corpora. <http://www.ilc.cnr.it/EAGLES96/annotate/annotate.html>.
- Leidner, J. L. (2003). Current issues in software engineering for natural language processing. In *SEALT'S '03: Proceedings of the HLT-NAACL 2003 workshop on Software engineering and architecture of language technology systems*, Morristown, NJ, USA, S. 45–50. Association for Computational Linguistics.
- Lemnitzer, L., P. Gupta und H. Wunsch (2008). Acquisition of a new type of lexical-semantic relation from german corpora. In A. Zgrzywa, K. Choros und A. Sieminski (Hrsg.), *New Trends in Multimedia and Network Information Systems*. Amsterdam: IOS Press.
- Lemnitzer, L. und C. Kunze (2002). Adapting GermaNet for the web. In *Proceedings of 1st International WordNet Conference*. Mysore, India.
- Lemnitzer, L. und K. Naumann (2002). A virtual course in applied computational linguistics. <http://milca.sfs.uni-tuebingen.de/ACL/release/>. Online Textbook.
- Lemnitzer, L. und H. Zinsmeister (2006). *Korpuslinguistik. Eine Einführung*. narr studienbücher. Tübingen: Narr.
- Lenat, D. und R. Guha (1989). *Building large knowledge-based systems. Representation and inference in the Cyc project*. Reading, Mass.: Addison-Wesley.
- LEO. URL: <http://dict.leo.org/>. Stand: 02.07.2009.
- Leopold, E. (2002). Das Zipfsche Gesetz. *Künstliche Intelligenz* 2/02, 34.
- Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *SIGDOC '86: Proceedings of the 5th annual international conference on Systems documentation*, New York, NY, USA, S. 24–26. ACM.

- Levelt, W. (2008). *An Introduction to the Theory of Formal Languages and Automata*. Amsterdam: John Benjamins.
- Levinson, S. (1983). *Pragmatics*. Cambridge, Mass.: Cambridge University Press.
- Levinson, S. (2000). *Presumptive Meanings*. Cambridge, Mass.: MIT Press.
- Levy, R. und G. Andrew (2006). Tregex and Tsurgeon: Tools for querying and manipulating tree data structures. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, Genua, Italien.
- Lewis, D. (1969). *Convention*. Cambridge, MA: Harvard University Press.
- Li, F. und Y. Yang (2003). A loss function analysis for classification methods in text categorization. In *ICML*, S. 472–479.
- Li, M., M. Zhu, Y. Zhang und M. Zhou (2006). Exploring distributional similarity based models for query spelling correction. In *Proceedings of the 21st International Conference on Computational Linguistics (COLING 2006) and 44th Annual Meeting of the ACL*, Sydney, Australien, S. 1025–1032.
- Li, W. und A. McCallum (2003). Rapid development of hindi named entity recognition using conditional random fields and feature induction. *ACM Trans. Asian Lang. Inf. Process.* 2(3), 290–294.
- Lidstone, G. J. (1920). Note on the general case of the Bayes-Laplace formula for inductive or a posteriori probabilities. *Transactions of the Faculty of Actuaries* 8, 182–192.
- Lin, D. (1998). An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, Madison, WI, S. 296–304.
- Lita, L. und J. Carbonell (2004). Instance-based question answering: A data-driven approach. In *EMNLP 2004*, Barcelona, Spain.
- Lizogat, G. (2000). From language to motion, and back: generating and using route descriptions. In D. Christodoulakis (Hrsg.), *Natural Language Processing 2000*, Number 1835 in LNCS, S. 328–345. Berlin und Heidelberg: Springer-Verlag.
- Lobin, H. (1999a). Intelligente Dokumente – Linguistische Repräsentation komplexer Inhalte für die hypermediale Wissensvermittlung. In H. Lobin (Hrsg.), *Text im digitalen Medium*, S. 155–178. Wiesbaden: Westdeutscher Verlag.
- Lobin, H. (Hrsg.) (1999b). *Text im digitalen Medium – Linguistische Aspekte von Textdesign, Texttechnologie und Hypertext Engineering*. Wiesbaden: Westdeutscher Verlag.
- Lobin, H. (2000). *Informationsmodellierung in XML und SGML*. Berlin, Heidelberg, New York etc.: Springer.
- Lobin, H. (2003). Textauszeichnung und Dokumentgrammatiken. In H. Lobin und L. Lemnitzer (Hrsg.), *Texttechnologie*, S. 51–82. Tübingen: Stauffenburg. Erscheint.
- Lobin, H. und L. Lemnitzer (2004a). Text(e) technologisch. In H. Lobin und L. Lemnitzer (Hrsg.), *Texttechnologie*, S. 1–9. Tübingen: Stauffenburg.
- Lobin, H. und L. Lemnitzer (Hrsg.) (2004b). *Texttechnologie – Anwendungen und Perspektiven*. Tübingen: Stauffenburg.
- Lowrance, R. und R. A. Wagner (1975). An extension of the string-to-string correction problem. *Journal of the ACM* 22(2), 177–183.
- Lüdeling, A. (2008). Mehrdeutigkeiten und Kategorisierung: Probleme bei der Annotation von Lerner korpora. In P. Grommes und M. Walter (Hrsg.), *Fortgeschrittene Lernervarietäten*, S. 119–140. Tübingen: Niemeyer.
- Lüdeling, A. und M. Kytö (Hrsg.) (2008). *Corpus Linguistics. An International Handbook*. Handbücher zur Sprache und Kommunikationswissenschaft / Handbooks of Linguistics and Communication Science 29.1. Berlin/New York: Mouton de Gruyter.
- Luo, X. (2005). On coreference resolution performance metrics. In *Proceedings of the Human Language Technology Conference and the 2005 Conference on Empirical Methods in Natural Language Processing*, Vancouver, B.C., Canada, 6–8 October 2005, S. 25–32.
- Luo, X., A. Ittycheriah, H. Jing, N. Kambhatla und S. Roukos (2004). A mention-synchronous coreference resolution algorithm based on the Bell Tree. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain, 21–26 July 2004, S. 136–143.

- Lutz, M. (2008). *Learning Python* (3rd ed Ausg.). Beijing: O'Reilly.
- MacWhinney, B. (1995). *The CHILDES-Project: Tools for Analyzing Talk* (2nd Ausg.). Hillsdale, NJ: Erlbaum.
- Maegard, B. (1999). Chapter 4 machine translation. In E. Hovy, N. Ide, R. Frederking, J. Mariani und A. Zampolli (Hrsg.), *Multilingual Information Management: Current Levels and Future Abilities*, <http://www.cs.cmu.edu/~ref/mlim/index.html>. US National Science Foundation.
- Magerman, D. M. (1995). Statistical decision-tree models for parsing. In *Proc. 33rd Annual Meeting of the Association of Computational Linguistics, ACL-1995*, Cambridge, MA.
- Mahesh, K. (1996). Ontology development for machine translation: Ideology and methodology. Technischer Bericht MCCS-96-292, NMSU CRL.
- Mahesh, K. und S. Nirenburg (1995). A situated ontology for practical NLP. In *Proc. Workshop on Basic Ontological Issues in Knowledge Sharing, International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada.
- Mahesh, K., S. Nirenburg, J. Cowie und D. Farwell (1996). An assessment of Cyc for natural language processing. Technischer Bericht MCCS-96-302, Computing Research Laboratory, New Mexico State University, Las Cruces, NM.
- Malenke, M., M. Bäumler und E. Paulus (2000). Speech recognition performance assessment. In W. Wahlster (Hrsg.), *Verbmobil: Foundations of Speech-to-speech Translation*, S. 583–591. Berlin: Springer.
- Maler, E. und J. E. Andaloussi (1996). *Developing SGML DTDs – From Text to Model to Markup*. Upper Saddle River: Prentice Hall.
- Mann, W. und C. Matthiessen (1985). Demonstration of the nigel text generation computer program. In Benson und Greaves (Hrsg.), *Systemic Perspectives on Discourse*, S. 50–83. Norwood Ablex.
- Mann, W. und S. Thompson (1988). Rhetorical Structure Theory: Toward a functional theory of text organization. *Text* 8(3), 243–281.
- Manning, C. und H. Schütze (2003). *Foundations of Statistical Natural Language Processing* (Sechste Ausg.). Cambridge, Massachusetts: The MIT Press.
- Manning, C. D. und B. Carpenter (1997). Probabilistic parsing using left corner language models. In *Proc. 5th International Workshop on Parsing Technologies*, Boston.
- Manning, C. D., P. Raghavan und H. Schütze (2008). *Introduction to Information Retrieval*. Cambridge University Press.
- Marcu, D. (2000). The rhetorical parsing of unrestricted texts: A surface-based approach. *Computational Linguistics* 26(3), 395–448.
- Marcus, M., G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz und B. Schasberger (1994). The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the ARPA Human Language Technology Workshop, HLT 94*, Plainsboro, NJ, S. 114–119.
- Marcus, M., B. Santorini und M. Marcinkiewicz (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19. <ftp://ftp.cis.upenn.edu/pub/treebank/doc/c193.ps.gz>.
- Marcus, M. P. (1980). *A Theory of Syntactic Recognition for Natural Language*. Cambridge: MIT Press.
- Markman, A. B. (1999). *Knowledge representation*. Mahwah, NJ: Lawrence Erlbaum Associates.
- Martell, C. (2002). FORM: An extensible, kinematically-based gesture annotation scheme. In *Proceedings of ICSLP*, S. 353–356.
- Mason, O. (2000). *Programming for corpus linguistics : how to do text analysis with Java*. Edinburgh textbooks in empirical linguistics. Edinburgh: Edinburgh University Press.
- Massion, F. (2005). *Translation Memory Systeme im Vergleich*. Reutlingen: doculine.
- Matsumoto, Y. (2003). Lexical knowledge acquisition. In R. Mitkov (Hrsg.), *The Oxford Handbook of Computational Linguistics*, S. 395–413. Oxford University Press.

- Mays, E., F. J. Damerau und R. L. Mercer (1991). Context based spelling correction. *Information Processing & Management* 27(5), 517–522.
- McDonald, D. D. (2000). Natural Language Generation. In R. Dale, H. Moisl und H. Somers (Hrsg.), *A handbook of natural language processing: techniques and applications for the processing of language as text*, S. 147–179. New York: Marcel Dekker.
- McDonald, R., K. Crammer und F. Pereira (2005). Online large-margin training of dependency parsers. In *Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics, ACL-2005*, S. 91–98.
- McDonald, R., F. C. N. Pereira, S. Kulick, R. S. Winters, Y. Jin und P. S. White (2005). Simple Algorithms for Complex Relation Extraction with Applications to Biomedical IE. In *ACL*.
- McEnery, T. und A. Wilson (2001). *Corpus Linguistics* (2nd Ausg.). Edinburgh: Edinburgh University Press.
- McGlashan, S. (2004, March). Voice extensible markup language (VoiceXML) version 2.0. <http://www.w3.org/TR/voicexml20/>.
- McGuinness, D. und F. van Harmelen (2004). OWL web ontology language overview. Technischer Bericht, W3C Recommendation.
- McKelvie, D., C. Brew und H. Thompson (1997). Using SGML as a Basis for Data-Intensive NLP. In *Proc. of Applied Natural Language Processing (ANLP) 97*, Washington D.C. ACL.
- McKeown, K. (1985a). Discourse strategies for generating natural-language text. *Artificial Intelligence* 27, 1–41.
- McKeown, K. (1985b). *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge, U.K.: Cambridge University Press.
- McKeown, K., S. Pan, J. Shaw, D. Jordon und B. Allen (1997). Language generation for multimedia healthcare briefings. In *Proc. of the 5th Conference on Applied Natural Language Processing (ANLP-1997)*, S. 277–282.
- McNeill, D. (2005). *Gesture and Thought*. Chicago: University of Chicago Press.
- McTear, M. F. (2004). *Spoken Dialogue Technology: Toward the Conversational User Interface*. London: Springer.
- Meadow, C. T. (1992). *Text Information Retrieval Systems*. San Diego: Academic Press.
- Meggle, G. (Hrsg.) (1979). *Handlung, Kommunikation, Bedeutung*. Suhrkamp.
- Mehler, A. und H. Lobi (Hrsg.) (2003). *Automatische Textanalyse – Systeme und Methoden zur Annotation und Analyse natürlichsprachlicher Texte*. Wiesbaden: Westdeutscher Verlag. Erscheint.
- Melcuk, I. (1982). Lexical functions in lexicographic descriptions. In *Proc. of the 8th Annual Meeting of the Berkeley Linguistic Society*.
- Melcuk, I. und A. Polguere (1987). A formal lexicon in the meaning-text theory. *Computational Linguistics* 13(3–4), 261–275.
- Melli, G., M. Ester und A. Sarkar (2007). Recognition of multi-sentence n-ary subcellular localization mentions in biomedical abstracts. In *LBM (Short Papers)*.
- Mengel, A. und W. Lezius (2000). An XML-based encoding format for syntactically annotated corpora. In *Proceedings of the Second International Conference on Language Resources and Engineering (LREC 2000)*, Athen, Griechenland, S. 121–126.
- Merhav, N. und Y. Ephraim (1991). Hidden markov modeling using a dominant state sequence with application to speech recognition. *Computer Speech & Language* 5(4), 327–339.
- Meyers, A., R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young und R. Grishman (2004). Annotating Noun Argument Structure for NomBank. In *Proceedings of LREC-2004*, Lisbon, Portugal, S. 803–806.
- Mikheev, A. (2003). Text segmentation. In R. Mitkov (Hrsg.), *Oxford Handbook of Computational Linguistics*, S. 201–218. Oxford et al.: Oxford University Press.

- Miller, G., R. Beckwith, C. Fellbaum, D. Gross und K. Miller (1990). Five papers on wordnet. Technischer Bericht, CSL Report 43, Cognitive Science Laboratory, Princeton University.
- Miltzakaki, E., R. Prasad, A. Joshi und B. Webber. (2004). Annotating discourse connectives and their arguments. In *Proceedings of the HLT/NAACL Workshop on Frontiers in Corpus Annotation*, Boston, MA, S. 9–16.
- Minsky, M. (1975). A framework for representing knowledge. In P. H. Winston (Hrsg.), *The Psychology of Computer Vision*, S. 211–277. New York: McGraw Hill.
- Mintert, S. (Hrsg.) (2002). *XML & Co. Die W3C-Spezifikationen für Dokumenten- und Datenarchitektur*. München, Boston, San Francisco etc.: Addison-Wesley.
- Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.
- Mitkov, R. (1998). Robust pronoun resolution with limited knowledge. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics*, Montréal, Québec, Canada, 10–14 August 1998, S. 869–875.
- Mitkov, R. (Hrsg.) (2003). *The Oxford Handbook of Computational Linguistics*. Oxford: Oxford University Press.
- Mitton, R. (1996). *English Spelling and the Computer*. London: Longman. <http://eprints.bbk.ac.uk/archive/00000469;2008/12/26>.
- Möbius, B. (1999). The Bell Labs German text-to-speech system. *Computer Speech and Language* 13, 319–358.
- Moeller, S., J. Krebber, A. Raake, P. Smeele, M. Rajman, M. Melichar, V. Pallotta, G. Tsakou, B. Kladis, A. Vovos, J. Hoonhout, D. Schuchardt, N. Fakotakis, T. Ganchev und I. Potamitis (2004). INSPIRE: Evaluation of a smart-home system for infotainment management and device control. In *Proceedings of the 4th International Conference on Language Resources and Evaluation, LREC 2004*, Lissabon, Portugal, S. 1603–1606.
- Mohri, M. (1996). On some applications of finite-state automata theory to natural language processing. *Natural Language Engineering* 2(1), 61–68.
- Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational Linguistics* 23(2), 269–311.
- Moldovan, D., S. Harabagui, C. Clark, M. Bowden, J. Lehmann und J. Williams (2004). Experiments and analysis of lcc's two qa systems over trec 2004. In *TREC 2004*, Gaithersburg, USA.
- Mönnoch, U. und F. Morawietz (2003). Formale Grundlagen. In H. Lobin und L. Lemnitzer (Hrsg.), *Texttechnologie*, S. 109–141. Tübingen: Stauffenburg.
- Montague, R. (1970a). English as a Formal Language. In B. e. a. Visentini (Hrsg.), *Linguagi nella Societ'a e nella Tecnica*, S. 189–224. Milan: Edizioni di Comunità. Wieder abgedruckt in: Montague (1974), 188–221.
- Montague, R. (1970b). Universal Grammar. *Theoria* 36, 373–398. Wieder abgedruckt in: Montague (1974), 222–246.
- Montague, R. (1973). The Proper Treatment of Quantification in Ordinary English. In J. Hintikka, J. Moravcsik und P. Suppes (Hrsg.), *Approaches to Natural Language*, S. 221–242. Dordrecht: Reidel. Wieder abgedruckt in: Montague (1974), 247–270.
- Montague, R. (1974). *Formal Philosophy. Selected Papers of Richard Montague*. New Haven and London.
- Morik, K. (1989). User models and conversational settings: modeling the user's wants. In A. Kobsa und W. Wahlster (Hrsg.), *User Models in Dialog Systems*, S. 364–385. Berlin Heidelberg New York Tokyo: Springer Verlag, Symbolic Computation Series.
- Moschitti, A., D. Pighin und R. Basili (2008). Tree kernels for semantic role labeling. *Computational Linguistics* 34(2), 193–224.
- Moser, L. (1992). Simulating turing machines in DATR. Brighton: University of Sussex, Cognitive Science Research Paper CSRP 241.,
- Mukherjee, J. (2002). *Korpuslinguistik und Englischunterricht: Eine Einführung*. Frankfurt am Main: Peter Lang.

- Müller, C. (2007). Resolving it, this, and that in unrestricted multi-party dialog. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, Prague, Czech Republic, 23–30 June 2007, S. 816–823.
- Müller, C. und M. Strube (2006). Multi-level annotation of linguistic data with MMAX2. In S. Braun, K. Kohn und J. Mukherjee (Hrsg.), *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*, S. 197–214. Peter Lang: Frankfurt a.M., Germany.
- Müller, S. (2008). *Head-Driven Phrase Structure Grammar: Eine Einführung*. 2. Aufl. Stauffenburg.
- Möller, S. (2002). A new taxonomy for the quality of telephone services based on spoken dialogue systems. In *Proceedings of the Third SIGdial Workshop on Discourse and Dialogue*, Philadelphia, S. 142–153. Association for Computational Linguistics.
- Mühlhäuser, M. und I. Gurevych (2007). *Handbook of Research on Ubiquitous Computing Technology for Real Time Enterprises*. Hershey PA, USA: IGI Global.
- Müller, C. (2008). JWKT1: Java-based Wiktionary API. <http://www.ukp.tu-darmstadt.de/software/jwkt1/>.
- Müller, C. und I. Gurevych (2008). Using Wikipedia and Wiktionary in domain-specific information retrieval. In F. Borri, A. Nardi und C. Peters (Hrsg.), *Working Notes for the CLEF 2008 Workshop*.
- Nadeau, D. und S. Sekine (2007). A survey of named entity recognition and classification. *Linguisticae Investigationes* 30(1), 3–26.
- Naumann, K. (2006). Manual of the annotation of in-document referential relations. http://www.sfs.uni-tuebingen.de/resources/tuebadz_relations_man.pdf.
- Naumann, S. und H. Langer (1994). *Parsing. Eine Einführung in die maschinelle Analyse natürlicher Sprache*. Stuttgart: Teubner.
- Navarro, G. (2001). A guided tour to approximate string matching. *ACM Computing Surveys* 33(1), 31–88.
- Nesselhauf, N. (2004). Learner Corpora and their Potential for Language Teaching. In J. Sinclair (Hrsg.), *How to use corpora in Language Teaching*, S. 125–152. Amsterdam: John Benjamins.
- Neumann, G. (2006). *A Hybrid Machine Learning Approach for Information Extraction from Free Texts.*, S. 390–397. Studies in Classification, Data Analysis, and Knowledge Organization. Springer-Verlag Berlin, Heidelberg, New-York.
- Neumann, G. (2008). Strategien zur webbasierten multilingualen Fragebeantwortung. *Computer Science - Research and Development* 22(2), 71–84.
- Neumann, G., R. Backofen, J. Baur, M. Becker und C. Braun (1997). An information extraction core system for real world german text processing. In *5th Applied Natural Language Processing Conference*, Washington, USA.
- Neumann, G. und F. Xu (2004). Mining natural language answers from the web. *International Journal of Web Intelligence and Agent Systems*.
- Newell, A. und H. Simon (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the ACM* 19, 113 – 126.
- Ng, V. (2005). Machine learning for coreference resolution: From local classification to global ranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, Ann Arbor, Mich., 25–30 June 2005, S. 157–164.
- Ng, V. und C. Cardie (2002). Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Penn., 7–12 July 2002, S. 104–111.
- Nirenburg, S. und V. Raskin (2004). *Ontological Semantics*. The MIT Press.
- Nissim, M., S. Dingare, J. Carletta und M. Steedman (2004). An annotation scheme for information status in dialogue. In *Proceedings of the 4th Conference on Language Resources and Evaluation (LREC2004)*, Lisbon.
- NIST (2008). Machine translation evaluation (Open MT-08) Official evaluation results. S. 188–221. NIST. zuletzt aktualisiert 09.12.2008, zuletzt besucht 05.01.2009.

- Nivre, J. (2008). Algorithms for deterministic incremental dependency parsing. *Computational Linguistics* 34(4), 513–553.
- Nivre, J., J. Hall, J. Nilsson, G. Eryiğit und S. Marinov (2006). Labelled pseudo-projective dependency parsing with support vector machines. In *Proc. 10th Conference on Computational Natural Language Learning, CoNLL-2006*, S. 221–225.
- Nivre, J. und R. McDonald (2008). Integrating graph-based and transition-based dependency parsers. In *Proc. 46th Annual Meeting of the Association for Computational Linguistics, ACL-HLT-2008*, S. 950–958.
- Norvig, P. (2007). How to write a spelling corrector. <http://norvig.com/spell-correct.html>; 2008/12/26.
- Numberg, G. (1979). The nonuniqueness of semantic solutions: Polysemy. *Linguistics and Philosophy* 3, 143–184.
- Oberhofer, W. (1979). *Wahrscheinlichkeitstheorie*. München; Wien: Oldenbourg.
- Oberle, D., A. Ankolekar, P. Hitzler, P. Cimiano, M. Sintek, M. Kiesel, B. Mougiouie, S. Vembu, S. Baumann, M. Romanelli, P. Buitelaar, R. Engel, D. Sonntag, N. Reithinger, B. Loos, R. Porzel, H.-P. Zorn, V. Micelli, C. Schmidt, M. Weiten, F. Burkhardt und J. Zhou (2007). DOLCE ergo SUMO: On foundational and domain models in the smart-web integrated ontology (SWIntO). *Journal of Web Semantics: Science, Services and Agents on the World Wide Web* 5(3), 156–174.
- Och, F. J. und H. Ney (2000). Improved statistical alignment models. In *Proc. of the 38th Annual Meeting of the Association for Computational Linguistics*, Hongkong, China, S. 440–447.
- Och, F. J. und H. Ney (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics* 29(1), 19–51.
- Och, F. J. und H. Ney (2004). The alignment template approach to statistical machine translation. *Computational Linguistics* 30(4), 417–449.
- Oepen, S., K. Toutanova, S. Shieber, C. Manning, D. Flickinger und T. Brants (2002). The LinGO Redwoods treebank: Motivation and preliminary applications. In *In Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, Taipei, S. 1253–1257.
- Oflazer, K. (1996). Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics* 22(1), 73–89.
- Okuda, T., E. Tanaka und T. Kasai (1976). A method for the correction of garbled words based on the Levenshtein metric. *IEEE Transactions* 25(2), 172–178.
- Oppenheim, A. N. (1979 (1966)). *Questionnaire design and attitude measurement*. London: Heinemann.
- Oviatt, S., R. Lunsford und R. Coulston (2005). Individual differences in multimodal integration patterns: what are they and why do they exist? In *Proceedings of ACM CHI 2005 Conference on Human Factors in Computing Systems*, S. 241–249. ACM Press.
- Pala, K. et al. (2002). Balkanet: A multilingual semantic network for balkan languages. In *Proceedings of 1st International WordNet Conference*. Mysore, India.
- Pallett, D., J. Fiscus, W. Fisher, J. Garafolo, B. Lund, A. Martin und M. Przybocki (1994). 1994 benchmark tests for the ARPA spoken language program. In *Proceedings of the ARPA Workshop on Spoken Language Technology*, S. 5–36.
- Palmer, D. und M. Hearst (1997). Adaptive multilingual sentence boundary disambiguation. *Computational Linguistics* 23(2), 241–267.
- Palmer, M., D. Gildea und P. Kingsbury (2005). The Proposition Bank: A corpus annotated with semantic roles. *Computational Linguistics* 31(1), 71–106.
- Papineni, K., S. Roukos, T. Ward und W.-J. Zhu (2001). Bleu: a method for automatic evaluation of machine translation. Technischer Bericht RC22176 (W0109-022), IBM Research Division, Thomas J. Watson Research Center, Almaden.
- Paris, C. (1993). *User modelling in text generation*. London: Pinter Publishers.
- Paris, C., M. Wu, A.-M. Vercoustre, S. Wan, P. Wilkins und R. Wilkinson (2003). An Empirical Study of the Effect of Coherent and Tailored Document Delivery as an Interface

- to Organizational Websites. In *The Proceedings of the Adaptive Hypermedia Workshop at the 2003 User Modelling Conference*, Pittsburgh, USA, S. 133–144.
- Partee, B. (1992). Syntactic categories and semantic type. In M. Rosner und R. Johnson (Hrsg.), *Computational Linguistics and Formal Semantics*. Cambridge: Cambridge University Press.
- Partee, B., A. ter Meulen und R. Wall (1990). *Mathematical Methods in Linguistics*. Dordrecht: Kluwer Academic Publishers.
- Passonneau, R. J. (2004). Computing reliability for co-reference annotation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, Lisbon, Portugal, 26–28 May 2004, S. 1503–1506.
- Paul, D. B. und J. M. Baker (1992). The design for the wall street journal-based CSR corpus. In *HLT '91: Proceedings of the workshop on Speech and Natural Language*, Morristown, NJ, USA, S. 357–362. Association for Computational Linguistics.
- Peckham, J. (1993). A new generation of spoken dialogue systems: Results and lessons from the SUNDIAL project. In *Proceedings of the 3rd European Conference on Speech Communication and Technology*, Volume 1, S. 33–40.
- Pellegrini, T. und A. Blumauer (Hrsg.) (2006). *Semantic Web: Wege zur vernetzten Wissensgesellschaft*. Berlin: Springer Verlag.
- Pemberton, S. (2002). XHTML 1.0: The Extensible Hypertext Markup Language (Second Edition). Technische Spezifikation, W3C.
- Pepper, S. und G. Moore (2001). XML Topic Maps (XTM) 1.0. Technische Spezifikation, TopicMaps.Org. Online verfügbar: <http://www.topicmaps.org/xtm/1.0/>.
- Pereira, F. (2000). Formal grammar and information theory: Together again? *Philosophical Transactions of the Royal Society* 358(1769), 1239–1253.
- Pereira, F. und S. Shieber (1987). *Prolog and Natural-Language Analysis*. CSLI Lecture Notes. Stanford, California: Center for the Study of Language and Information.
- Pereira, F. und D. Warren (1980). Definite clause grammars for natural language analysis. *Artificial Intelligence* 23, 231–278.
- Peterson, J. L. (1980). Computer programs for detecting and correcting spelling errors. *Communications of the ACM* 23(12), 676–687.
- Peterson, J. L. (1986). A note on undetected typing errors. *Communications of the ACM* 29(7), 633–637.
- Pfanzagl, J. (1988). *Elementare Wahrscheinlichkeitsrechnung*. Berlin; New York: de Gruyter.
- Pinkal, M. (1985). *Logik und Lexikon. Die Semantik des Unbestimmten*. Berlin/New York: de Gruyter.
- Poesio, M. (2000). The GNOME Annotation Scheme Manual. http://cswww.essex.ac.uk/Research/nle/corpora/GNOME/anno_manual_4.htm.
- Poesio, M. und M. A. Kabadjanov (2004). A general-purpose, off-the-shelf anaphora resolution module: Implementation and preliminary evaluation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, Lisbon, Portugal, 26–28 May 2004, S. 663–666.
- Polguère, A. (2009). Lexical systems: graph models of natural language lexicons. *Language Resources and Evaluation* 43(1), 41–55. Multilingual Language Resources and Interoperability.
- Pollard, C. und I. Sag (1987). *Information-based Syntax and Semantics. Vol 1: Fundamentals*. Stanford, CA: CSLI Lecture Notes 13.
- Pollard, C. und I. A. Sag (1994). *Head-Driven Phrase Structure Grammar*. Chicago: The University of Chicago Press.
- Pompino-Marschall, B. (2003). *Artikulatorische und akustische Phonetik*. Berlin: Walter de Gruyter.
- Ponzetto, S. P. und M. Strube (2006). Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, New York, N.Y., 4–9 June 2006, S. 192–199.

- Popescu-Belis, A. (2008). Reference-based vs. task-based evaluation of human language technology. In *Proceedings of the ELRA Workshop on Evaluation. Looking into the Future of Evaluation: When automatic metrics meet task-based and performance-based approaches*, Marrakesh, S. 12–16.
- Popescu-Belis, A., P. Estrella, M. King und N. Underwood (2006). A model for context-based evaluation of language processing systems and its applications to machine translation evaluation. In *Proceedings of the 5th International Conference on Language Resources and Evaluation, LREC 2006*, Genua, Italy, S. 691–696.
- Porzel, R., M. Jansche und R. Klabunde (2002). The Generation of Spatial Descriptions from a Cognitive Point of View. In K. Coventry und P. Olivier (Hrsg.), *Spatial Language. Cognitive and Computational Aspects*. Dordrecht: Kluwer.
- Povlsen, C., A. Sågvall Hein und K. de Smedt (1999). Scarrie Final Report. <http://ling.uib.no/~desmedt/scarrie/final-report.html>; 2008/12/26.
- Power, R., D. Scott und R. Evans (1998). What you see is what you meant: direct knowledge editings with natural language feedback. In H. Prade (Hrsg.), *13th European Conference on Artificial Intelligence (ECAI'98)*, S. 677–681. Chichester, England: John Wiley and Sons.
- Prince, A. und P. Smolensky (1993). Optimality Theory. Constraint Interaction in Generative Grammar. Technischer Bericht, Rutgers University Center for Cognitive Science.
- Pustejovsky, J. (1991). The generative lexicon. *Computational Linguistics* 17(4), 409–441.
- Pustejovsky, J. (1995). *The Generative Lexicon*. Cambridge: The MIT Press.
- Pustejovsky, J. und S. Bergler (Hrsg.) (1992). *Lexical Semantics and Knowledge Representation*. Berlin: Springer.
- Pustejovsky, J., P. Hanks, R. Saurí, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D. Day, L. Ferro und M. Lazo (2003). The TIMEBANK Corpus. In *Proceedings of Corpus Linguistics*, S. 647–656.
- Qiu, L., M.-Y. Kan und T.-S. Chua (2004). A public reference implementation of the RAP anaphora resolution algorithm. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, Lisbon, Portugal, 26–28 May 2004, S. 291–294.
- Quillian, M. R. (1968). Semantic memory. In M. Minsky (Hrsg.), *Semantic Information Processing*, S. 227–270. Mass.: MIT Press.
- Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings IEEE* 77(2), 257–285.
- Rabiner, L. und B. Juang (1993). *Fundamentals of speech recognition*. Englewood Cliffs NJ: PTR Prentice Hall.
- Radev, D. (2003). Panel on web-based question answering. In *AAAI Spring Symposium on New Directions in Question Answering*.
- Radev, D. R., E. Hovy und K. McKeown (2002). Introduction to the special issue on summarization. *Comput. Linguist.* 28(4), 399–408.
- Raggett, D., A. L. Hors und I. Jacob (1999). HTML 4.01 Specification. Technische Spezifikation, W3C.
- Ramakrishnan, G., D. Paranjape, S. Chakrabarti und P. Bhattacharyya (2004). Is question answering an acquired skill? In *WWW04*.
- Ramírez Bustamente, F. und F. Sánchez León (1996). GramCheck: A grammar and style checker. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING 1996)*, Kopenhagen, Dänemark.
- Ravichandran, D. und E. H. Hovy (2002). Learning surface text patterns for a question answering system. In *ACL*, S. 41–47.
- Ravin, Y. (1993). Grammar errors and style weaknesses in a text-critiquing system. See Jensen, Heidorn und Richardson (1993), S. 65–76.
- Recanati, F. (2004). *Literal Meaning*. Cambridge, UK: Cambridge University Press.
- Reetz, H. (2003). *Artikulatorische und akustische Phonetik*. Trier: Wissenschaftlicher Verlag.

- Rehm, G. (1999). Automatische Textannotation – Ein SGML- und DSSSL-basierter Ansatz zur angewandten Textlinguistik. In H. Lobin (Hrsg.), *Text im digitalen Medium*, S. 179–195. Wiesbaden: Westdeutscher Verlag.
- Reidsma, D. und J. Carletta (2008). Reliability measurement without limits. *Computational Linguistics* 34(3), 319–326.
- Reimer, U. (1991). *Einführung in die Wissensrepräsentation*. Stuttgart: B.G. Teubner.
- Reinke, U. (2004). *Translation Memories. Systeme - Konzepte - Linguistische Optimierung*. Frankfurt am Main: Peter Lang.
- Reiter, E. (1994). Has a Consensus NL Generation Architecture Appeared, and is it Psycholinguistically Plausible? In *Proc. of the 7th International Workshop on Natural Language Generation (INLGW-94)*, S. 163–170.
- Reiter, E. (1995). NLG vs. Templates. In *Proceedings of the Fifth European Workshop on Natural Language Generation*, Leiden, The Netherlands, S. 95–105. Faculty of Social and Behavioural Sciences, University of Leiden.
- Reiter, E. und R. Dale (2000). *Building Natural Language Generation Systems*. Cambridge: Cambridge University Press.
- Reiter, E., C. Mellish und J. Levine (1995). Automatic generation of technical documentation. *Applied Artificial Intelligence* 9, 259–287.
- Reiter, E., R. Robertson und L. Osman (2003). Lessons from a Failure: Generating Tailored Smoking Cessation Letters. *Artificial Intelligence* 144, 41–58.
- Reithinger, N., G. Herzog und A. Blocher (2007). Smartweb - mobile broadband access to the semantic web. *KI Zeitschrift* (2).
- Resnik, P. (1995, August). Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada, S. 448–453.
- Reyde, U. (1993). Dealing with ambiguities by underspecification: Construction, representation and deduction. *Journal of Semantics* 10(2), 123–179.
- Rich, E. (1979). User modeling via stereotypes. *Cognitive Science* 3, 329–354.
- Richter, F. (2004). *A Mathematical Formalism for Linguistic Theories with an Application in Head-Driven Phrase Structure Grammar*. Dissertation (2000), Eberhard-Karls-Universität Tübingen.
- Richter, H. (1966). *Wahrscheinlichkeitstheorie*. Berlin; Heidelberg; New York: Springer-Verlag.
- Richter, H. (2004). Reguläre Sprachen, reguläre Ausdrücke. Technischer Bericht, Leibniz-Rechenzentrum der bayerischen Akademie der Wissenschaften. <http://www.lrz-muenchen.de/services/schulung/unterlagen/regul/>, zuletzt geprüft am 19.02.2009.
- Riester, A. (2008). A semantic explication of *Information Status* and the underspecification of the recipients' knowledge. In A. Grønn (Hrsg.), *Proceedings of SuB-12*, Oslo, S. 508–522.
- Riloff, E. und R. Jones (1999). Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI/IAAI*, S. 474–479.
- Rohde, D. (2001). Tgrep2. Technischer Bericht, Carnegie Mellon University.
- Rosch, E. (1978). Principles of categorization. In E. Rosch und B. Lloyd (Hrsg.), *Cognition and Categorization*. Hillsdale: Lawrence Erlbaum Associates.
- Rosenfeld, B. und R. Feldman (2006). Ures : an unsupervised web relation extraction system. In *ACL*.
- Rudolph, E., R. Pegam und S. Polubinski (2001). Probleme bei der Erstellung eines deutschen Referenzkorpus für automatisches Tagging. Unveröffentlichtes Manuskript, Sprachwissenschaftliches Institut, Ruhr-Universität Bochum.
- Ruiz-Casado, M., E. Alfonseca und P. Castells (2005). Automatic Assignment of Wikipedia Encyclopedic Entries to WordNet Synsets. *Advances in Web Intelligence*, 380–386.

- Ruoff, A. (1984). *Alltagstexte I. Transkriptionen von Tonbandaufnahmen aus Baden-Württemberg und Bayrisch-Schwaben mit zwei Karten*. IDIOMATICA 10. Veröffentlichungen der Tübinger Arbeitsstelle "Sprache in Südwestdeutschland". Tübingen: Niemeyer.
- Ruppenhofer, J., M. Ellsworth, M. Petrucc, C. Johnson und J. Scheffczyk (2006). *FrameNet II, Extended Theory and Practice*. Berkeley: ICSI FrameNet Project.
- Sacaleanu, B. und G. Neumann (2006). A cross-lingual german-english framework for open-domain question answering. In *CLEF*, S. 328–338.
- Saeed, J. I. (2008). *Semantics* (Dritte Ausg.). Wiley-Blackwell.
- Sag, I. A., T. Wasow und E. M. Bender (2003). *Syntactic Theory: A Formal Introduction* (Zweite Ausg.). Number 152 in CSLI Lecture Notes. Stanford: Center for the Study of Language and Information.
- Sampson, G. (1983). Context-free parsing and the adequacy of context-free languages. In M. King (Hrsg.), *Parsing Natural Language*, S. 151–170. London: Academic Press.
- Sampson, G. (2003). Thoughts on two decades of drawing trees. In A. Abeillé (Hrsg.), *Treebanks: Building and Using Parsed Corpora*. Kluwer.
- Sasaki, F. und A. Witt (2003). Linguistische Korpora. In H. Lobin und L. Lemnitzer (Hrsg.), *Texttechnologie*, S. 195–216. Tübingen: Stauffenburg. Erscheint.
- Schäfer und Weyrath (1997). Retrospective thinking-aloud study of inferences by firemen about emergency callers. In A. Jameson, C. Paris und C. Tasso (Hrsg.), *Proceedings of the Sixth International Conference on User Modeling (UM97)*, Berlin, S. 377–388. Springer Verlag. (Chia Laguna, Sardinia, Italy).
- Schank, R. C. (1975). *Conceptual Information Processing*. Amsterdam: North-Holland.
- Scheid, H. (1992). *Wahrscheinlichkeitsrechnung*. Mannheim; Leipzig; Wien; Zürich: BI-Wissenschafts-Verlag.
- Schiel, F., S. Burger, A. Geumann und K. Weilhammer (1997). The Partitur Format at BAS. FIPKM report #35, Institut für Phonetik und Sprachliche Kommunikation, Universität München.
- Schiel, F., C. Draxler, A. Baumann, T. Ellbogen und A. Steffen (2003). *The Production of Speech Corpora*. Institut für Phonetik und Sprachliche Kommunikation, Universität München.
- Schiel, F., S. Steininger und U. Türk (2002). The martKom Multimodal Corpus at BAS. In *Proceedings of Second International Conference on Language Resources and Evaluation (LREC2002)*, S. 200–206.
- Schiller, A., S. Teufel und C. Stückert (1999). Guidelines für das Tagging deutscher Textkorpora mit STTS. Technischer Bericht, University of Stuttgart, University of Tübingen, Stuttgart.
- Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing (NeMLaP)*, S. 44–49.
- Schmid, H. (1995, March). Improvements in part-of-speech tagging with an application to German. In *Proceedings of the ACL SIGDAT Workshop*.
- Schmid, H. (2000). Unsupervised learning of period disambiguation for tokenisation. Internal Report, IMS, Stuttgart.
- Schmidt, I. (2003). Modellierung von Metadaten. In H. Lobin und L. Lemnitzer (Hrsg.), *Texttechnologie*, S. 143–164. Tübingen: Stauffenburg.
- Schmidt, S. (1968). *Sprache und Denken als sprachphilosophisches Problem von Locke bis Wittgenstein*. Den Haag.
- Schmidt, T., C. Chiarcos, T. Lehmburg, G. Rehm, A. Witt und E. Hinrichs (2006). Avoiding Data Graveyards: From Heterogeneous Data Collected in Multiple Research Projects to Sustainable Linguistic Resources. In *Proceedings of the E-MELD 2006 Workshop on Digital Language Documentation: Tools and Standards – The State of the Art*, East Lansing, Michigan.

- Schmidt, T. und K. Wörner (2005). Erstellen und Analysieren von Gesprächskorpora mit EXMARAlDA. *Gesprächsforschung* Vol. 6, 171–195.
- Schmidt-Wigger, A. (1998). Grammar and style checking for German. In *Proceedings of the 2nd International Workshop on Controlled Language Applications (CLAW 98)*, Pittsburgh, PA.
- Schmitz, U. (1992). *Computerlinguistik. Eine Einführung*. Opladen: Westdeutscher Verlag.
- Schölkopf, B. und A. J. Smola (2002). *Learning with Kernels*. MIT Press.
- Schöning, U. (1995). *Logik für Informatiker*. Heidelberg: Spektrum Akademischer Verlag.
- Schöning, U. (1999). *Theoretische Informatik – kurzgefaßt*. Heidelberg: Spektrum-Verlag.
- Schukat-Talamazzini, E. G. (1995). *Automatische Spracherkennung*. Braunschweig: Vieweg.
- Schulte im Walde, S. (2009). The Induction of Verb Frames and Verb Classes from Corpora. In A. Lüdeling und M. Kyö (Hrsg.), *Corpus Linguistics. An International Handbook*, Volume 2, S. 952–971. Berlin: Mouton de Gruyter.
- Schulz, K. U. und S. Mihov (2001). Fast string correction with levenshtein-automata. Technischer Bericht 01-127, Universität München, Centrum für Informations- und Sprachverarbeitung. <http://citeseervx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.652>.
- Schwartz, R., T. Phoenix und B. Foy (2008). *Learning Perl: [covers Perl 5.10]*. Beijing: O'Reilly.
- Searle, J. (1990). Artificial intelligence: A debate—is the brain's mind a computer program? *Scientific American*.
- Sedgewick, R. (1992). *Algorithmen in C*. Bonn: Addison-Wesley.
- Seibel, P. (2004). *Practical Common Lisp*. Berkeley, Calif.: Apress.
- Sekine, S. (2006). On-demand information extraction. In *ACL*.
- Sekine, S., K. Sudo und C. Nobata (2002). Extended named entity hierarchy. In *The Third International Conference on Language Resources and Evaluation*. Canary Island, Spain.
- Sells, P. (1985). *Lectures on Contemporary Syntactic Theories*. CSLI Lecture Notes 3. Stanford University Press.
- Senia, F. und J. van Velden (1997). Specification of Orthographic Transcription and Lexicon Conventions. Technischer Bericht SD1.3.2, SpeechDat-II LE-4001.
- SENSEVAL (2009). The SENSEVAL homepage. URL <http://www.senseval.org/>. Last checked: 20 August 2009.
- Shannon, C. und W. Weaver (1949). *The mathematical theory of communication*. Urbana: University of Illinois Press.
- Shannon, C. E. (1948, July, October). A mathematical theory of communication. *The Bell System Technical Journal* 27, 379–423, 623–656.
- Shaw, J. (1998a). Clause aggregation using linguistic knowledge. In *Proc. of the 9th International Workshop on Natural Language Generation (INLGW-98)*, S. 138–147.
- Shaw, J. (1998b). Segregatory coordination and ellipsis in text generation. In *Proc. of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational linguistics (COLING-ACL'98)*, S. 1220–1126.
- Shieber, S. (1985). Evidence against the context-freeness of natural languages. *Linguistics and Philosophy* 8, 362–366.
- Shieber, S., F. Pereira, G. van Noord und R. Moore (1990). Semantic-head-driven generation. *Computational Linguistics* 16, 30–42.
- Shieber, S. M. (1986). *An Introduction to Unification-Based Approaches to Grammar*. CSLI Lecture Notes. Stanford, CA.
- Shieber, S. M. (1992). *Constraint-Based Grammar Formalisms*. Cambridge, MA: Bradford.
- Shinyama, Y. und S. Sekine (2006). Preemptive information extraction using unrestricted relation discovery. In *HLT-NAACL*.
- Sidner, C. L. (1983). Focusing in the comprehension of definite anaphora. In M. Brady und R. Berwick (Hrsg.), *Computational Models of Discourse*, S. 267–330. Cambridge, Mass.: MIT Press.

- Sierra, K. und B. Bates (2008). *Java von Kopf bis Fuß : behandelt Java 5.0* (3., korr. Nachdr. Ausg.). Beijing: O'Reilly.
- SIGWAC, A. (2008). The Special Interest Group of the Association for Computational Linguistics (ACL) on Web as Corpus. <http://www.sigwac.org.uk/>.
- Simov, K. und P. Osenova (2003). Practical annotation scheme for an HPSG Treebank of Bulgarian. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora (LINC-2003)*, Budapest, Ungarn.
- Skut, W., T. Brants, B. Krenn und H. Uszkoreit (1998). A linguistically interpreted corpus of German newspaper texts. In *ESSLLI Workshop on Recent Advances in Corpus Annotation*, Saarbrücken.
- Sloetjes, H., A. Russel und A. Klassmann (2007). ELAN: a free and open-source multimedia annotation tool. In *Proc. of Interspeech*, Antwerpen, S. 4015–4016.
- Smith, B. (2003). Ontology. In L. Floridi (Hrsg.), *Blackwell Guide to the Philosophy of Computing and Information*, S. 155–166. Oxford: Blackwell.
- Smith, B. C. (1982). Prologue to ‘reflection and semantics in a procedural language’ [reprinted in Brachman and Levesque (1985)]. Technischer Bericht 272, MIT.
- Soderland, S. (1997). *Learning Text Analysis Rules for Domain Specific Natural Language Processing*. Ph. D. thesis, University of Massachusetts Amherst.
- Somers, H. (2003). Machine translation: Latest developments. In R. Mitkov (Hrsg.), *The Oxford Handbook of Computational Linguistics*, Chapter 28, S. 512–528. Oxford University Press.
- Somtag, G. (1999). Evaluation der Prosodie. In *Berichte aus der Kommunikationstechnik*. Aachen: Shaker.
- Soon, W. M., H. T. Ng und D. C. Y. Lim (2001). A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics* 27(4), 521–544.
- Sowa, J. F. (1984). *Conceptual Structures : Information Processing in Mind and Machine*. Reading, Mass.: Addison-Wesley.
- Sowa, J. F. (2000). *Knowledge representation: logical, philosophical, and computational foundations*. Pacific Grove, CA.: Brooks/Cole.
- Spencer, A. (1991). *Morphological Theory*. Oxford: Blackwell.
- Sperber, D. und D. Wilson (1986). *Relevance. Communication and Cognition*. Basil Blackwell.
- Sperberg-McQueen, C. M. und L. Burnard (Hrsg.) (2002). *TEI P4: Guidelines for Electronic Text Encoding and Interchange*. Text Encoding Initiative Consortium.
- Sproat, R. (1992). *Morphology and Computation*. Cambridge: MIT Press.
- Sproat, R. (Hrsg.) (1998). *Multilingual Text-to-Speech Synthesis. The Bell Labs Approach*. Dordrecht, Boston, London: Kluwer Academic Publishers.
- Sripada, S. G., E. Reiter, J. Hunter, J. Yu und I. P. Davy (2001). Modelling the Task of Summarising Time Series Data using KA Techniques. In *Applications and Innovations in Intelligent Systems IX: Proceedings of the International Conference on Knowledge Based Systems and Applied Artificial Intelligence (ES2001)*, Cambridge, S. 183–196. The British Computer Society SGAI: The Specialist Group on Artificial Intelligence.
- Stalnaker, R. C. (1973). Presuppositions. *Journal of Philosophical Logic* 2, 447–457.
- Stalnaker, R. C. (1974). Pragmatic presuppositions. In M. K. Kunitz und P. K. Unger (Hrsg.), *Semantics and Philosophy*, S. 197–230. New York University Press.
- Steude, M. (1996). Lexical options in multilingual generation from a knowledge base. In G. Adorni und M. Zock (Hrsg.), *Trends in Natural Language Generation: An Artificial Intelligence Perspective*, S. 222–237. Springer-Verlag.
- Stegmann, R., H. Telljohann und E. W. Hinrichs (2000). Stylebook for the German Treebank in VERBMOBIL. Technischer Bericht 239, Verbmobil.
- Stellman, A., J. Greene und L. Schulten (2008). *C# von Kopf bis Fuß : ein praxisorientiertes Arbeitsbuch für die Programmierung mit C# und .NET* (1. Aufl. Ausg.). Beijing: O'Reilly.

- Stephens, D. R., C. Diggins, J. Turkanitis und J. Cogswell (2006). *C++ Cookbook*. Beijing: O'Reilly.
- Stetina, J., S. Kurohashi und M. Nagao (1998). General word sense disambiguation method based on a full sentential context. In *Proceedings of COLING-ACL-98 Workshop Usage of WordNet in Natural Language Processing*. Montreal, Canada.
- Strube, M. (1998). Never look back: An alternative to centering. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics*, Montréal, Québec, Canada, 10–14 August 1998, Volume 2, S. 1251–1257.
- Strube, M. und U. Hahn (1999). Functional centering: Grounding referential coherence in information structure. *Computational Linguistics* 25(3), 309–344.
- Strube, M. und C. Müller (2003). A machine learning approach to pronoun resolution in spoken dialogue. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan, 7–12 July 2003, S. 168–175.
- Strube, M., S. Rapp und C. Müller (2002). The influence of minimum edit distance on reference resolution. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, Philadelphia, Penn., 6–7 July 2002, S. 312–319.
- Suchanek, F. M., G. Kasneci und G. Weikum (2007). Yago: a core of semantic knowledge. In *WWW*, S. 697–706.
- Sudhoff, S., D. Lenertová, R. Meyer, S. Pappert, P. Augurzky, I. Mleinek, N. Richter und J. Schließer (Hrsg.) (2006). *Methods in Empirical Prosody Research*. Berlin: Walter de Gruyter.
- Sudo, K., S. Sekine und R. Grishman (2003). pre-CODIE–Crosslingual On-Demand Information Extraction. In *HLT-NAACL*.
- Sudo, K., S. Sekine und R. Grishman (2004). Cross-lingual information extraction system evaluation. In *Proceedings of Coling 2004*, Geneva, Switzerland, S. 882–888. COLING.
- Suri, L. Z., K. F. McCoy und J. D. DeCristofaro (1999). A methodology for extending focusing frameworks. *Computational Linguistics* 25(2), 173–194.
- Sutton, C. und A. McCallum (2006). An introduction to conditional random fields for relational learning. In L. Getoor und B. Taskar (Hrsg.), *Introduction to Statistical Relational Learning*. MIT Press.
- Taylor, P. (2009). *Text-to-Speech Synthesis*. Cambridge: Cambridge University Press.
- TEI AI1W2 (1991). List of Common Morphological Features for Inclusion in TEI Starter Set of Grammatical-Annotation Tags. <http://www.w3.org/People/csmcq/1991/ai1w02.html>.
- Telljohann, H., E. Hinrichs, S. Kübler und H. Zinsmeister (2006). *Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z)*. Germany: Seminar für Sprachwissenschaft, Universität Tübingen.
- Telljohann, H., E. W. Hinrichs und S. Kübler (2004). The TüBa-D/Z treebank: Annotating German with a context-free backbone. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, Lisbon, Portugal, 26–28 May 2004, S. 2229–2235.
- Tesnière, L. (1959, dt. Grundzüge der strukturellen Syntax, Stuttgart 1980). *Éléments de syntaxe structurale*. Paris.
- Tetreault, J. R. (2001). A corpus-based evaluation of centering and pronoun resolution. *Computational Linguistics* 27(4), 507–520.
- Thompson, H. S. und D. McKelvie (1997). Hyperlink Semantics for Standoff Markup of Read-Only Documents. In *Proceedings of SGML Europe '97: The next decade – Pushing the Envelope*, Barcelona, S. 227–229.
- Thouin, B. (1982). The meteo system. In V. Lawson (Hrsg.), *Practical experience of machine translation*, S. 39–44. Amsterdam: North-Holland.
- Thümmel, W. (1993). *Geschichte der Syntaxforschung. Westliche Entwicklungen*, S. 131–199. Berlin u.a.: Walter de Gruyter.

- Tillmann, H. G. (1997). Eight Main Differences between Collections of Written and Spoken Language Data. FIPKM report #35, Institut für Phonetik und Sprachliche Kommunikation, Universität München.
- Tomlinson, M., R. Winski und W. Barry (1988). Label file format proposal. Technischer Bericht, ESPRIT Project 1542 (SAM), Extension Phase, Final Report.
- Trawinski, B., J.-P. Söhn, M. Sailer und F. Richter (2008). A multilingual electronic database of distributionally idiosyncratic items. In E. Bernal und J. DeCesaris (Hrsg.), *Proceedings of the XIII Euralex International Congress*, Volume 20 of *Activitats*, Barcelona, Spain, S. 1445–1451.
- Trommer, J. (1999). Natural language allomorphy in mo_lex. In *Proceedings of Vextal Conference*.
- Turing, A. (1936). On computable numbers with an application to the entscheidungsproblem. *Proc. London Math. Soc.* 2, 230–265.
- Turing, A. M. (1950). Computing machinery and intelligence. *Mind* 59, 433–560.
- Turner, R., S. Sripada, E. Reiter und I. P. Davy (2007). Selecting the content of textual descriptions of geographically located events in spatio-temporal weather data. *Applications and Innovations in Intelligent Systems XV*, 75–88.
- Turney, P. D. (2001). Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *ECML*, S. 491–502.
- Ukkonen, E. (1992). Approximate string-matching with q -grams and maximal matches. *Theoretical Computer Science* 92, 191–211.
- Ule, T. und E. Hinrichs (2003). Linguistische Annotation. In H. Lobin und L. Lemnitzer (Hrsg.), *Texttechnologie*, S. 217–243. Tübingen: Stauffenburg. Erscheint.
- Ule, T. und F. H. Müller (2003). KaRoPars: Ein System zur linguistischen Annotation großer Text-Korpora des Deutschen. In A. Mehler und H. Lobin (Hrsg.), *Automatische Textanalyse*. Wiesbaden: Westdeutscher Verlag. Erscheint.
- Ullensboom, C. (2009). *Java ist auch eine Insel : programmieren mit der Java Platform, Standard Edition Version 6* (8., aktual. u. erw. Aufl. Ausg.). Galileo Computing. Bonn: Galileo Press.
- Uszkoreit, H. (1986). Categorial unification grammars. In *Coling*, S. 187–194.
- Uszkoreit, H. (1987). The Stuttgart Type Unification Formalism. Lilog report, IBM Germany.
- van Deemter, K. und S. Peters (Hrsg.) (1996). *Semantic Ambiguity and Underspecification*. Stanford: CLSI Publications.
- van der Linden, K. (2000). Natural language generation. In D. Jurafsky und J. Martin (Hrsg.), *Speech and Language Processing: an introduction to speech recognition, computational linguistics and natural language processing*, Chapter 20. New Jersey: Prentice-Hall.
- van der Sandt, R. (1992). Presupposition projection as anaphora resolution. *Journal of Semantics* 9, 333–377.
- van der Sluis, I., M. Theune, E. Reiter und E. Krahmer (Hrsg.) (2007). *Proceedings of the Workshop on Multimodal Output Generation (MOG 2007)*. Centre for Telematics and Information Technology (CTIT), University of Twente.
- van Eynde, F. und D. Gibbon (2000). *Lexicon Development for Speech and Language Processing*. Boston & Dordrecht: Kluwer Academic Publishers.
- van Oirschot, R. (1987). *The Syntax of Coordination*. Beckenham: Croom Helm.
- van Santen, J. P. H. (1997). Segmental duration and speech timing. In Y. Sagisaka, N. Campbell und N. Higuchi (Hrsg.), *Computing Prosody—Computational Models for Processing Spontaneous Speech*, S. 225–249. New York: Springer.
- Van Valin, R. D. und R. J. LaPolla (1997). *Syntax: Structure, Meaning and Function*. Cambridge, England: Cambridge university press.
- Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer.

- Vasilescu, F., P. Langlais und G. Lapalme (2004). Evaluating variants of the Lesk approach for disambiguating words. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, Lisbon, Portugal, S. 633–636.
- Véronis, J. (1998). A study of polysemy judgements and inter-annotator agreement. In *Proceedings of SENSEVAL-1*, Herstmonceux Castle, Sussex, UK.
- Versley, Y. (2007). Antecedent selection techniques for high-recall coreference resolution. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Language Learning*, Prague, Czech Republic, 28–30 June 2007, S. 496–505.
- Versley, Y., S. P. Ponzetto, M. Poesio, V. Eidelman, A. Jern, J. Smith, X. Yang und A. Moeschitti (2008). BART: A modular toolkit for coreference resolution. In *Companion Volume to the Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, Columbus, Ohio, 15–20 June 2008, S. 9–12.
- Vieira, R. und M. Poesio (2000). An empirically-based system for processing definite descriptions. *Computational Linguistics* 26(4), 539–593.
- Vilain, M., J. Burger, J. Aberdeen, D. Connolly und L. Hirschman (1995). A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Message Understanding Conference (MUC-6)*, San Mateo, Cal., S. 45–52. Morgan Kaufmann.
- Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* 13, 260–269.
- Volk, M. (1995). *Einsatz einer Testsatzsammlung im Grammar Engineering*. Tübingen: Niemeyer.
- Volk, M. (1998). Markup of a Test Suite with SGML. In J. Nerbonne (Hrsg.), *Linguistic Databases*, S. 59–76. Cambridge University Press.
- Volk, M. (2008). The automatic translation of film subtitles. a machine translation success story? In J. Nivre, M. Dahllöf und B. Megyesi (Hrsg.), *Resourceful Language Technology: Festschrift in Honor of Anna Sägåvall Hein*, Volume 7 of *Studia Linguistica Upsaliensia*, S. 202–214. Uppsala University, Humanistisk-samhällsvetenskapliga vetenskapsområdet, Faculty of Languages.
- Vossen, P. (1999). *EuroWordNet: a multilingual database with lexical–semantic networks*. Kluwer Academic Publishers.
- WAC (2008). The Web as Corpus Website. <http://webascorpus.sourceforge.net/>.
- Wagner, A. und C. Kunze (1999). Integrating GermaNet into EuroWordNet, a multi-lingual lexical–semantic database. *Sprache und Datenverarbeitung* 23.2, 5–20.
- Wagner, R. A. (1974). Order-n correction for regular languages. *Communications of the ACM* 17(5), 265–268.
- Wagner, R. A. und M. J. Fischer (1974). The string-to-string correction problem. *Journal of the ACM* 21(1), 168–173.
- Wahlster, W. (1993). Verbmobil: Translation of face-to-face dialogues. In *Proc. of the 3rd European Conference on Speech Communication and Technology*, Berlin.
- Wahlster, W. (Hrsg.) (2000). *Verbmobil: Foundations of Speech-To-Speech Translation*. Artificial Intelligence. Berlin, Heidelberg, New York: Springer Verlag.
- Wahlster, W. (Hrsg.) (2006). *SmartKom - Foundations of Multimodal Dialogue Systems*. Berlin: Springer Verlag.
- Wahlster, W. (2007). Smartweb: multimodal web services on the road. In *ACM Multimedia*, S. 16.
- Waibel, A. und C. Fügen (2008). Spoken language translation. enabling cross-lingual human–human communication. In *IEEE Signal Processing Magazine May 2008*, S. 70–79. IEEE Signal Processing Magazine.
- Walker, D. (1999). Taking Snapshots of the Web with a TEI Camera. *Computers and the Humanities* (33), 185–192.
- Walker, M., L. Hirschmann und J. Aberdeen (2000). Evaluation for DARPA COMMUNICATOR spoken dialogue systems. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation, LREC 2000*, Athen, Greece, S. 735–741.

- Walker, M., M. Iida und S. Cote (1994). Japanese discourse and the process of centering. *Computational Linguistics* 20(2), 193–233.
- Walker, M., A. K. Joshi und E. Prince (1998). *Centering Theory in Discourse*. Oxford University Press.
- Walker, M., D. Litman, C. Kamm und A. Abella (1997). PARADISE: A general framework for evaluating spoken dialogue agents. In *Proc. ACL/EACL*, S. 271U–280.
- Walker, M., R. Passonneau und J. Boland (2001). Quantitative and qualitative evaluation of DARPA COMMUNICATOR spoken dialogue systems. In *Meeting of the Association of Computational Linguistics*, S. 515–522.
- Walther, M. (1999). One-level prosodic morphology. Marburger Arbeiten zur Linguistik.
- Wang, R. und G. Neumann (2007). Recognizing textual entailment using a subsequence kernel method. In *AAAI*.
- Way, A. und N. Gough (2005). Comparing example-based and statistical machine translation. *Natural Language Engineering* 11(3).
- Weaver, W. (1949). Translation. In W. N. Locke und A. D. B. (1955) (Hrsg.), *reprinted in: Machine Translation of Languages*, S. 15–23. Cambridge, MA: MIT Press.
- Weaver, W. (1955). Foreword: the new tower. In W. Locke und A. Booth (Hrsg.), *Machine translation of languages*, S. v–vii. Cambridge, Mass.: Technology Press of M.I.T.
- Weber, H.-J. (1997). *Dependenzgrammatik. Ein interaktives Arbeitsbuch*. Günter Narr.
- Wells, J. (1997). Standards, Assessment, and Methods: Phonetic Alphabets. Technischer Bericht, University College, London.
- White, M. (2004). Reining in ccg chart realization. In *Proc. of the 3rd International Conference on Natural Language Generation (INLG-04)*, S. 182–191.
- Whitelock, P. (1992). Shake-and-bake translation. In *Proc. of COLING-92*, S. 610–616.
- Wiebe, J., T. Wilson, R. Bruce, M. Bell und M. Martin (2004). Learning subjective language. *Computational Linguistics* 30(3), 277–308.
- Wierzbicka, A. (1996). *Semantics. Primes and Universals*. Oxford: Oxford University Press.
- Wiese, R. (1994). Phonological vs. morphological rules: on German Umlaut and Ablaut. *Journal of Linguistics* 32, 113–135.
- Williams, M. (2004). *Translation quality assessment: An argumentation-centred approach*. Ottawa: University of Ottawa Press.
- Williams, S. und E. Reiter (2005). Appropriate Microplanning Choices for Low-Skilled Readers. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, Edinburgh, S. 1704–1705.
- Winograd, T. und F. Flores (1986). *Understanding computers and cognition: a new foundation for design*. Norwood, New Jersey: Ablex.
- Wirth, N. (1983). *Algorithmen und Datenstrukturen*. Stuttgart: Teubner.
- Witt, A. (1999). SGML und Linguistik. In H. Lobin (Hrsg.), *Text im digitalen Medium*, S. 121–154. Wiesbaden: Westdeutscher Verlag.
- Witt, A. (2003). Linguistische Informationsmodellierung mit XML. In A. Mehler und H. Lobin (Hrsg.), *Automatische Textanalyse*. Wiesbaden: Westdeutscher Verlag. Erscheint.
- Wittgenstein, L. (1922). *Tractatus logico-philosophicus*. Routledge.
- Wolff, C. (2003). Systemarchitekturen – Aufbau texttechnologischer Anwendungen. In H. Lobin und L. Lemitzer (Hrsg.), *Texttechnologie*, S. 165–192. Tübingen: Stauffenburg. Erscheint.
- Woods, W. A. (1970). Transition network grammars for natural language analysis. *Communications of the ACM* 13(10), 591–606.
- Woods, W. A. (1975). What's in a link: foundations for semantic networks. In D. Bobrow und A. Collins (Hrsg.), *Representation and Understanding*, S. 35–82. New York: Academic Press.
- Xu, F., H. Uszkoreit und H. Li (2007). A seed-driven bottom-up machine learning framework for extracting relations of various complexity. In *ACL*.

- Xu, J. (2000). Multilingual search on the World Wide Web. In *Presentation to HICSS-33*, Maui, Hawaii.
- Xue, N., F. Xia, F.-D. Chiou und M. Palmer (2005). The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering* 11(2), 207–238.
- Yang, X., J. Su und C. L. Tan (2005). Improving pronoun resolution using statistics-based semantic compatibility information. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, Ann Arbor, Mich., 25–30 June 2005, S. 165–172.
- Yang, X., J. Su und C. L. Tan (2008). A twin-candidate model for learning-based anaphora resolution. *Computational Linguistics* 34(3), 327–356.
- Yangarber, R., R. Grishman, P. Tapanainen und S. Huttunen (2000). Unsupervised discovery of scenario-level patterns for information extraction. In *Proceedings of the 6th ANLP*, Seattle, USA.
- Yangarber, R., W. Lin und R. Grishman (2002). Unsupervised learning of generalized names. In *COLING*.
- Yarowsky, D. (1995, June). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, Cambridge, MA, S. 189–196.
- Yates, A. und O. Etzioni (2007). Unsupervised resolution of objects and relations on the web. In *HLT-NAACL*.
- Young, S., M. Adda-Decker, X. Aubert, C. Dugast, J. Gauvain, D. Kershaw, L. Lamel, D. Leeuwen, D. Pye, A. Robinson, H. Steeneken und P. Woodland (1997). Multilingual large vocabulary speech recognition: The European SQALE project. In *Computer Speech and Language*, Volume 11, S. 73–89.
- Younger, D. (1967). Recognition and parsing of context-free languages in time n^3 . *Information and Control* 10, 189–208.
- Yu, J., E. Reiter, J. Hunter und C. Mellish (2005). Choosing the content of textual summaries of large time-series data sets. *Natural Language Engineering* 11.
- Zadeh, L. (1965). Fuzzy sets. *Inf. Control* 8, 338–353.
- Zeevat, H. (1988). *Combining Categorial Grammar and Unification*, S. 202–229. Reidel.
- Zelenko, D., C. Aone und A. Richardella (2003). Kernel methods for relation extraction. *Journal of Machine Learning Research* 3, 1083–1106.
- Zesch, T. (2008). JWPL: Java-based Wikipedia API. <http://www.ukp.tu-darmstadt.de/software/jwpl/>.
- Zesch, T., I. Gurevych und M. Mühlhäuser (2007). Analyzing and accessing Wikipedia as a lexical semantic resource. In *Data Structures for Linguistic Resources and Applications*, Tuebingen, Germany, S. 197–205. Gunter Narr, Tübingen.
- Zesch, T., C. Müller und I. Gurevych (2008a). Extracting lexical semantic knowledge from wikipedia and wiktionary. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, electronic proceedings, Marrakech, Morocco.
- Zesch, T., C. Müller und I. Gurevych (2008b). Using Wiktionary for computing semantic relatedness. In *Proceedings of Twenty-Third AAAI Conference on Artificial Intelligence*, Chicago, Illinois, S. 861–867.
- Zhao, S. und R. Grishman (2005). Extracting relations with integrated information using kernel methods. In *ACL*.
- Zinsmeister, H., J. Kuhn und S. Dipper (2002). TIGER TRANSFER – Utilizing LFG Parses for Treebank Annotations. In *Proceedings of the LFG 2002 Conference*, CSLI Publications.
- Zinsmeister, H., A. Witt, S. Kübler und E. Hinrichs (2008). Linguistically Annotated Corpora: Quality Assurance, Reusability and Sustainability. In A. L. üdelen und M. Kytö (Hrsg.), *Corpus Linguistics. An International Handbook*, Handbücher zur Sprache und Kommunikationswissenschaft / Handbooks of Linguistics and Communication Science 29.1, Chapter 37. Berlin/New York: Mouton de Gruyter.
- Zipf, G. K. (1935). *The Psycho-Biology of Language*. Boston, MA: Houghton Mifflin.

- Zipf, G. K. (1949). *Human Behavior and the Principle of Least Effort*. Cambridge, MA: Addison-Wesley.
- Zobel, J. und P. W. Dart (1995). Finding approximate matches in large lexicons. *Software - Practice and Experience* 25(3), 331–345.
- Zukerman, I. und D. Litman (2001). Natural language processing and user modeling: synergies and limitations. *User modeling and user-adapted interaction* 11, 129–158.

Index

Symbolen

α -Konversion	63
β -Normalform	63
β -Redex	63
β -Reduktion	63
η -Reduktion	63
λ -Konversion, <i>siehe</i> Lambda-Konversion	

A

A-Box	538
Abdeckung	490
von Grammatiken	311
Abkürzung	265
Ablaufkontrolle	627
Ablaut	239
Ableitbarkeit	43
Ableitung	68
direkte Ableitung	68
Linksableitung	80
Abschlusseigenschaften	
kontextfreie Sprachen	84
kontextsensitive Sprachen	85
reguläre Sprachen	79
rekursiv aufzählbare Sprachen	92
rekursive Sprachen	92
TAG	87
Abtastfrequenz	197, 199
Abtastrate	525
Abtrennbarkeit	414
ACE, <i>siehe</i> Automatic Content Extraction	
Evaluation	
ACL (Association for Computational Linguistics)	17
ad hoc query	591
additive Konstante	198
Adjazenz	502
Adjektiv	
attributives	345
intersektives	346
prädikatives	342
Adjunktion	86
Affix	238
Aggregation	437, 454
Ähnlichkeitsfunktion	590
ähnlichster String	
Suche	557
Akronym	265
Akustisches Modell	665

Akzent	189
Tonakzent	187
Akzentplatzierung	188
Akzentuierung	226
Algorithmus	67
Aliasing	197
Alignierung	487
Alignment	630
Allgemeingültigkeit	37, 44
Allomorph	238
Allomorphie	
partielle	238
phonologisch konditionierte	239
uneigentliche	239
Allophon	179, 180
Allquantor, <i>siehe</i> Quantor, Allquantor	
Alphabet	66
Ambiguität	280, 351, 371
syntaktische	308
Amplitude	197, 198, 525
Analyse	
lexikalische, <i>siehe</i> Dekodierung	218
semantische	218
syntaktische	217
Analyserichtung	305
Anapher	399
Anaphernresolution	399
Annotation	407
Evaluierung	407
Heuristik	403
maschinelles Lernen	404, 406
Anfechtbarkeit	414, 416
Angemessenheitsfunktion	109
Angewandte NLG-Systeme	634
Annotate	493
Annotation	159, 190, 196, 483, 484, 488, 526
diskursbezogen	484
Editor	529
Fehler	484
Konsistenz	484, 493
konstituenzbasierte	493
Kriterien	484
Label	484
Qualität	484
Schema	492, 493, 498
semantische	484
standoff	408

- syntaktische 484
- theorie neutrale **494**, 495
- theoriespezifische 494
- Antonym 549
- Antonymie 505
- Antwortextraktion **584**, 608
- Antworttyp 583
- Anwendungen
 - der Computerlinguistik 553
- Applikation 625
- Approximate String Matching, *siehe* ähnlichster String, Suche
- Äquijunktion 35
- Äquivalenz **38**, 52, 70
 - starke 290
- Architektur
 - integrative 309
 - sequentielle 309
- Argumentstruktur 387
- Artikel-Graph 548
- Aspekt 484
- atomarer Typ 109
- atomarer Wert 98
- Attribut **534**, 581
- Attribut-Wert-Logik **171**, 204
- Attribut-Wert-Struktur, *siehe* Merkmalsstruktur
- attribute-value matrix, *siehe* Merkmalsstruktur
- Aufmerksamkeitsfokus 447
- Aufnahmephase 193
- Aufnahmesoftware 529
- Auskunftsysteem 280, **424**
- Aussage 34
- äußerer Knoten 96
- Aussprachelexikon 528
- Ausspracheregeln 226
- Aussprachevariante **171**, 194
- Auszeichnung **159**, 160, 167
 - manuelle 167
 - maschinelle 167
- Auszeichnungselement 160–162
- Auszeichnungssprache, *siehe* Markup-Sprache
- Autokorrelation 201
- Automat 70
 - deterministischer endlicher **74**, 93
 - deterministischer Kellerautomat **81**, 93
 - endlicher .. **73**, 171, 182, 186, 205, 227, 236, 244
 - Kellerautomat 80
 - linear beschränkter **85**, 93
 - nichtdeterministischer endlicher **75**, 93
 - nichtdeterministischer Kellerautomat **81**, 93
 - stochastischer **6**
- Automatentheorie 5
- Automatic Content Extraction Evaluation **407**, 409
- Autorenystem 657
- B**
- Backtrack-Parsing 313
- Backtracking 313, 318
- Bank of England Corpus 489
- Bar-Level 292
- Base Concept 508
- Baseline, *siehe* Vergleichsmaßstab
- Basic Level Concept 509
- Baum **96**, 159
 - elementarer 86
 - binärer 97
- Baum-Welch-Algorithmus 144–146
- Baum-Welch-Training, *siehe* Baum-Welch-Algorithmus
- Baumbank 492
- Baumgrammatik
 - lokale 163
 - reguläre 163
- Baumgraph 281
- Baumstruktur 161
- Bayerischen Archiv für Sprachsignale 488
- Bayes-Formel **115**, **122**–**123**, 131, 141, 559, 665
- Beam-Suche 307
- Bedeutung
 - konventionelle 330
 - literale 330
- Bedeutungsvokabular 548
- Begriffshierarchien 569
- Belegsammlung 482
- Benutzerkontrolle 628
- Benutzermodell 422
- Benutzermodellierung **422**, 634, 636
- Benutzerschnittstelle 626
- Berechenbarkeit 67
- Bernoulli-Effekt 174
- Beschreibungslogik 538
- Best-first-Suche 307
- Betonung 185, 187
- Bewegung 497
- Beweisbarkeit 43
- Beweisverfahren 40
- BFP-Algorithmus 401
- Bigramm 270, 545
- Biimplikation 35
- BLLIP Korpus 493
- Blockierung 242
- BNC 569
- boilerplates 545
- Bootstrapping 597
- Bottom 105
- Breitensuche 306
- British National Corpus **488**, 489
- Brown Corpus **487**, 488

- Browser 164
C
C 469
C++ 470
C-Struktur 298
C# 472
Cascading Style Sheet, *siehe* CSS 298
CAT, *siehe* Computer-Aided Translation 400
Centering-Modell 400
 backward-looking center 401
 forward-looking center 401
 preferred center 401
CGI, *siehe* Common Gateway Interface 31, 91
charakteristische Funktion 31, 91
Chart 317
Chart-Generierung 460
Chart-Parser 317
 aktiver 318
Chart-Parsing
 Packing 318
Chomsky-Hierarchie 93
Chooser 463
Chunk 276
Classifier 539
Clustering 593
 agglomeratives 593
 divisives 593
 flaches 594
 hierarchisches 593
Cocke-Kasami-Younger-Algorithmus 318
Common Gateway Interface 479
comparable corpus 568
Compiler 470
Computational Lexicography 566, 569
computational linguistics, *siehe* Computerlinguistik 1-2
Computer-Aided Translation, CAT 654
Computer-Spiele 620
Computerlinguistik 1-2
 korpusorientierte 6
 praktische 6
 theoretische 6
Computerphonetik 170, 196
Computerphonologie 170, 196, 204, 205
Computersemantik 331
concept-to-speech 228
Confusion Sets, *siehe* Verwechlungsmen-
gen 185-186, 209
Constraint 185-186, 209
Constraint Relaxation 562
Constraintlogik 209
Corpus Encoding Standard 486
Cross-Media Publishing 159, 641
crosslinguale Verfahren 613
CSS 164
CTS, *siehe* concept-to-speech 54
Currying 54
customisation 434
Cyc 542
D
Daten 533
Datenbanken 636
Datenmodell 495
Datenrate 525
DATR 204-205, 236, 251-257
DCG, *siehe* Definite Clause Grammar 39
Deduktionstheorem 39
Default-Unifikation 204
Default-Vererbung 204
Defaultlogik 171, 176, 208, 209
Definite Clause Grammar 113, 469
definite Kennzeichnung 335, 349
Deklarativität 298
Dekodierung
 lexikalische 217
DeMorgansche Regel 32, 39
Denotat 332
Dependenz 281-282, 484, 497
Dependenzannotation 494
Dependenzgrammatik 281
Derivation 226, 237
description logics, *siehe* Beschreibungslo-
gik 193
Designphase 281
Determinationssyntax 201
DFT 395
Dialog 484
Dialogakt 677
Dialogerfolgsrate 218, 630
Dialogsystem 15
 multimodales 624
Dice-Koeffizient 385
Digitalisierung 197, 525
Diktiersystem 222, 617
Diphon 195
Disambiguierung 226, 280, 309, 332, 371
 monotone 372
 von Eigennamen 598
Disambiguierungsseite 548
discourse entity, *siehe* Diskursentität
discourse model, *siehe* Diskursmodell
disjunkt 31
Disjunktion 35, 101
Diskrete Fourier-Transformation 201
Diskurs 361, 396
Diskursentität 399
Diskursinterpretation 629
Diskursmodell 399, 425
Diskursplanung 436, 437, 446, 634
Diskursprosodie 186
Diskursreferent 362
Diskursrelation 396, 442, 449, 484
Diskursrepräsentationsstruktur 361, 363

- Kondition 363
 Konstruktionsregeln 366
 Subordination 366
 Universum 363
 unterspezifizierte 375
 Zugänglichkeit 366
 Diskursrepräsentationstheorie 361
 Diskursemantik 330
 DocBook 164
 document classification 580
 Document Object Model, *siehe* DOM 361
 document retrieval 579
 Dokumentation 528
 Dokumentgrammatik, *siehe* DTD 361
 Dokumentklassifikation 580
 Dokumenttyp 159
 Dokumenttyp-Definition, *siehe* DTD 159
 DOM 165
 DOM-Prozessor 165
 Domäne 47, 335, 535
 eines Typs 57
 domänenoffene Informationsextraktion 594
 Dominance Constraints 377
 Dominanz 96, 502
 Domänen Ontologie 508
 donkey sentence, *siehe* Eselssätze 57
 Dortmunder CHAT-Korpus 488
 DRS, *siehe* Diskursrepräsentationsstruktur 488
 DRT, *siehe* Diskursrepräsentationstheorie 488
 DSP 199
 DTD 160–164, 167
 DTW-Verfahren 143
 Dublin Core 162
 Dublin Core Metadata Initiative 486
 Duplexbedingung 370
 Dynamik 525
 Dynamische Programmierung 317
- E**
- e-Accessibility 426
 e-Inclusion 426, 427
 e-Learning 620
 EAGLES 486
 Earley-Algorithmus 318, 319
 ECESS – European Center of Excellence for Speech Synthesis 675
 Echtzeit 626
 Echtzeitfaktor 527
 Effizienz
 von Parsern 312
 Eigennamenerkennung 581
 Einzelworterkennung 616
 Element 29
 ELIZA 466
 EM-Algorithmus 146, 230
 Emotion 484
 Emotionserkennung 619
 empirische linguistische Forschung 492, 495
 Endknoten 96
 Energie 177, 197, 200
 Entailment 506
 Entropie 126, 127
 Gibbs-Ungleichung 127, 151
 KL-Divergenz 127
 Kreuzentropie 127
 Entscheidbarkeit 67, 91
 der Prädikatenlogik 53
 Wortproblem 91
 Entwicklungskorpus 156, 490
 Ereignis 116
 atomares 116
 Informationsgehalt 125
 sicheres 117
 unabhängige Ereignisse 121
 unmögliches 117
 unvereinbare Ereignisse 117
 Ereignisextraktion 602
 Ereignisstruktur 388
 Erkenner 313
 error anticipation, *siehe* Fehlerantizipation 313
 Eselsanapher 362
 Eselssätze 362
 Etikettierung 526
 Europarl Corpus 487
 EuroWordNet 504, 508
 Evaluation 7, 492, 659
 benutzerorientierte 660
 entwicklerorientierte 660
 Entwicklungskorpus 156
 F-Maß 155
 Genauigkeit 154
 Kreuzvalidierung 154, 490
 Precision 155
 Recall 155
 referenz-basierte 666
 Testkorpus 154
 Trainingskorpus 154
 Überanpassung 154
 Existenzquantor, *siehe* Quantor, Existenzquantor 319
 Expansion 305
 Expansionswahrscheinlichkeit 300
 experimentelle Methode 192
 Expertensystem 423, 532
 Explikatur 415
 Extensible Markup Language, *siehe* XML 338
 Extensible Stylesheet Language, *siehe* XSL 338
 Extension 338
- F**
- F-Maß 155, 586
 F-Score, *siehe* F-Maß 586

- F-Struktur 298
F-Wert 490
FAHQQT 643
Fast Fourier Transformation 201
FCR, *siehe* feature co-occurrence restriction 292
feature co-occurrence restriction 292
feature structure, *siehe* Merkmalsstruktur
Fehlerantizipation 562
FFT 201
Filter 416
Finite State Transducer, *siehe* Transduktor 292
Finite-State-Maschine, *siehe* Transduktor
Fission 626, 630
Flexion 225, 237
Fokus-Modell 400
Folgerung 38, 52, 410
Folksonomien 544
Formale Sprache 66
Formant 177, 201
Formantsynthese 622
Formel
 allgemeingültige 37, 50
 atomare 46
 aussagenlogische 34
 erfüllbare 37, 50
 komplexe 46
 prädikatenlogische 46
 typenlogische 56
 unerfüllbare 37, 50
Fortsetzungsklassen 225
forward-backward-Algorithmus, *siehe* Baum-Welch-Algorithmus
Fourier-Transformation 200
Fourieranalyse 200
Fouriersynthese 200
Frageanalyse 584
Fragebeantwortung 550, 584
 textbasierte 606
 webbasierte 608
FrageTyp 583
Frames 535
Frequenz 197–199, 525
Frequenzdomäne 177, 200
FST, *siehe* Transduktor
FUG, *siehe* Functional Unification Grammar
Functional Unification Grammar 298
Funktion
 syntaktische 298
 grammatische 484, 494, 497, 498, 500
Funktionalapplikation 341
Funktionalismus 185
Fusion 626, 629
- G**
Garbage Collection 470
- Gazetteer 596
gemischte Initiative 628
Generalisierte Phrasenstrukturgrammatik 290
Generalisierung 106
generative Kapazität 290
Generatives Lexikon 379, 387
Generic Contextual Quality Model 664
Generierung 228, 613, 630, 633
 im engeren Sinn 13
 kopfgesteuerte 459
 semilexikalische 460
 shake-and-bake 460
 Template-Generierung 638
Generierungsarchitektur 438
Generierungslücke 439
Generierungssysteme 635
Genre 545
Geräusch 173, 175, 177, 200, 223
gerichteter Graph 95
GermaNet 504
gesprochene Sprache 502
Gestik 484
Gesundheitswesentexte, Generierung von 635
Gibbs-Ungleichung, *siehe* Entropie, Gibbs-Ungleichung
GIS (Geographical Information System) 637
Gleichverteilung 119, 126
Glottis 173
Goldstandard, *siehe* Referenzkorpus
Good-Turing-Schätzer 230
Government and Binding-Theorie 494, 496
GPSG, *siehe* Generalisierte Phrasenstrukturgrammatik
Gradientenabstieg 144
Grammatik 67, 218
 allgemeine Regelgrammatik 68, 87, 93
 Baumadjunktions-Grammatik9, 85–87, 93
 einseitig-linear 73
 formale 280
 kontextfreie 80, 161–163, 167–282
 kontextsensitive 85, 93
 lexikalisierte 110
 links-linear 73
 lokale Baumgrammatik 163
 probabilistische 300
 rechts-linear 73
 reguläre 205
 reguläre Baumgrammatik 163
 Typ-0 87
 Typ-1 85
 Typ-2 80
 Typ-3 73
 X-Bar-Grammatik 292
Grammatikkorrektur 562

- Grammatikmodell 493
 Graph 94, 95
 markierter 96
 Graphem 184
 Graphentheorie 5
 Grounding 630
 Grundfrequenz 175, 177, 186–188, 201
 Grundmorphem 237
 Guidelines 484
 Gutenberg Projekt 482
- H**
- Häsitationssignal 195
 Häufigkeitsverteilungen 229
 Hasse-Diagramm 104
 Head-Driven Phrase Structure Grammar9,
 113, 292, 293, 490, 494, 645
 head feature convention 292
 head feature principle 292, 294, 296
 Head Switching 644
 Head-Corner-Parsing 305
 Heldout Data 490
 Hidden-Markov-Modell, *siehe* HMM
 hineinquantifizieren 352
 HMM .. 124, 130–133, 171, 201, 206, 218,
 221, 280
 Übergangswahrscheinlichkeit 133
 Ausgabewahrscheinlichkeit 133
 Bakis 135
 Beobachtungswahrscheinlichkeit .. 135–
 140
 Definition 132–133
 diskret 220
 Emission 220
 Entwicklung 130
 Erkennungsproblem 219
 kontinuierlich 220
 lineares 135
 Links-Rechts 135, 220
 Normierungsbedingung 132, 133
 optimale Zustandsfolge .. 136, 140–143
 Parameteroptimierung .. 136, 143–147
 Rückwärtsprozedur 139–140
 Startwahrscheinlichkeit 132
 Trainingsproblem 219
 Transition 220
 Trigramm-HMM 152
 Vorwärtsprozedur 137, 139
 Hobbs-Algorithmus 400
 Hole 416
 Hole Semantics 373
 Homograph 380
 Homonymie 379, 380
 Homophon 380
 HPSG, *siehe* Head-Driven Phrase Struc-
 ture Grammar
 HTML 160–162, 165
 HTML-Dokument 160
- Hyperonymie 505
 Hypertext Markup Language, *siehe*
 HTML
 Hyponymie 505, 506, 548
 Hyponymierelation 506
- I**
- IAA, *siehe* manuelle Annotierung, Über-
 einstimmung
 ID-Regel 290
 ID/LP-Grammatik 290
 Implikation 35, 113, 505
 Implikatur 410
 eingebettete 415
 generalisierte 413
 klausale 413
 konventionelle 410
 konversationelle 410, 411
 partikularisierte 413
 skalare 412
 Indexierung 588
 Individualisierung 423
 Individuenbereich 48
 Infimum 104
 Inferenz 4, 535
 Inferenzmechanismus 538
 Information 533, 577
 information extraction, *siehe* Informati-
 onsextraktion
 Information Retrieval, *siehe* Informations-
 erschließung
 Information-Retrieval-Systeme 674
 Informationserschließung 510, 563
 Informationsextraktion 533, 550, 584, 592,
 594
 domänenoffene 594
 on-demand 604
 Informationsmanagement
 textbasiertes 577
 Informationsmodellierung 159
 Informationsrecherche
 natürlichsprachliche 550
 Informationsstatus 484
 Informationsstruktur 484
 Informationstechnologie 553
 Inhaltsfestlegung 437, 634
 Inhaltsorganisation 437, 634
 Initialwort, *siehe* Abkürzung
 Inkrementalität 362
 inkrementell 304
 Inline-Format 485
 Inschriftion 520
 Instanz 534
 Instrumentalphonetik 191, 192
 Intension 338
 Intensionalität 337
 Intensität 173, 197, 198
 Intention 424

Inter-Annotator Agreement, <i>siehe</i> manuelle Annotierung, Übereinstimmung	
Interlingua	645-646
Interlingualer Index	508
Internationale Phonetische Assoziation	176
Interpretation	
der Lambda-Typenlogik	62
der Aussagenlogik	36
der Prädikatenlogik	49
der Typenlogik	58
direkte Deutung	340
indirekte Deutung	340
phonetische	180 , 207, 209
typgetriebene	359
Interval	198
Intonation	171, 178, 185, 188 , 210
Intonationsmodell	233
Intonationsphrase	188
INTSINT	196
Institut für Deutsche Sprache	489
inverse Dokumentenfrequenz	588
invertierter Index	588
IPA	176, 194, 213
ISLE Meta Data Initiative	486
Item	317, 318
J	
Java	472
Junktor	34
K	
K-means	594
k-Nearest-Neighbor	592
k-NN, <i>siehe</i> k-Nearest-Neighbor	
Künstliche Intelligenz	4 , 532, 533
Kante	94 , 317
in semantischem Netz	535
Kappa-Koeffizient	153
Kardinalität	29
Kategorie	
syntaktische	283 , 292, 357, 461, 494, 498
Kategorialgrammatik	299
Kausation	505
Kerncorpus des DWDS	489
Kernel	600
Keyword-in-context	568
Klang	173 , 177, 200
Klasse	591
Klassifikation	382
Kleene-Plus	66 , 269
Kleene-Stern	66 , 269
Klitikum	266
Knoten	94
innerer	96
in semantischem Netz	535
Kodierung	180
Kognitionswissenschaft	4, 533
Kohärenz	396 , 400
Kollokation	490
Kommandowortsysteme	627
komunikative Absicht	436
komunikativer Sinn	330
komplexe Kategorie	97
komplexer Typ	109
komplexer Wert	99
Kompositazerlegung	226
Komposition	226, 237
Kompositionalität	333 , 391
Kompositionalitätsprinzip	333
Kompression	525
Kondition	
atomare	362
komplexe	363
Konfiguration	83
Konfigurationsübergang	83
Kongruenz	288
Konjunktion	35
Konkatenation	67
Konklusion	38
Konkordanz	517 , 568, 656
Konsistenz	495
Konsonant	175 , 176, 182
Konstituente	281
Konstituentenstruktur	281 , 484
Kontextanalyse	225
Kontextfenster	548
Kontextänderungspotential	331
Kontraktion	549
Kontraposition	38
Kontrollierte Sprache	643
Konversationsmaximen	411
Konzept	534
Konzept-Effizienz	670
Konzeptakkurtheit	669
Kooperationsprinzip	411
Kopf	292
Koreferenz	100, 297, 336 , 399, 484
Koreferenzbestimmung	582
Korpus	129, 190, 196, 204, 482, 492, 584
multilinguale	544
Annotation	167, 488
Designkriterien	489
empirische Evidenz	491
geschriebene Sprache	488
gesprochene Sprache	483 , 488
Größe	488
Medium	487
Monitorkorpus	489
monolinguale	487
Multifunktionalität	487
multilinguale	487
multimodale	488
opportunistisches	489
Persistenz	489

- Repräsentativität 487
 Sprachbezug 489
 Sprachenauswahl 487
 statischer 489
 Verwendungszweck 487
 Wiederverwendbarkeit 487
 Korpusfilter 269
 korpusphonetische Methode 192
 Korpusstatistisches Verfahren 21
 Korpustypologie 486
 Korrektheit 44
 Korrektursysteme 555
 kreuzende Kanten 495, 500
 Kreuzklassifikation 507
 Kreuzvalidierung, *siehe* Evaluation,
 Kreuzvalidierung
 Kullback-Leibler-Divergenz, *siehe* Entro-
 pie, KL-Divergenz
- L**
- Längenmonotonie 85
 Lücke
 lexikalische 644
 Label 526
 Lambda-Abstraktion 60
 Lambda-Kalkül 60
 Lambda-Konversion 63
 Lambda-Operator 60
 Lambda-Typenlogik 61
 Landessprachenerkennung 619
 language model, *siehe* Sprachmodell
 Laplace-Raum 119
 Lappin & Leass-Algorithmus 403
 Latent Semantic Analysis 610
 Lautdauermodell 230, **231**, 233
 Left-Corner-Strategie 305
 Lemma 484
 Lemmatisierung 383
 Lernerkorpus 484
 Lernverfahren
 überwachtes 592, 596, 599
 Bootstrapping 386
 maschinelles **386**, 584
 semi-überwachtes 601
 unüberwachtes 593
 Lesart 351
 Lesartendisambiguierung 379, 387, 510
 Indikatorwort 384
 Kontextwort 383
 Lesk-Algorithmus 384–387
 Zielwort 382
 Levenshtein-Distanz 558, 653
 Lexem 236
 Lexical Markup Framework 572
 Lexical Functional Grammar . 9, 298, 490,
 645
 Lexical Markup Framework 572, 573
 Lexikalische Akquisition 568
 lexikalische Funktionen 461
 lexikalischer Anker 600
 Lexikalisierung 437, **453**
 im eigentlichen Sinn **440**, 462
 im engeren Sinn 440
 Lexikalismus 290
 Lexikographie 15, 490, **517–519**, 566
 Lexikographie-Arbeitsplatz 574
 Lexikologie 518, **519**
 Lexikon . 185, 187, 190, 195, 196, 205, 225,
 391, **515**, 538, 556
 endosystemisches 518
 exosystemisches 518
 Hyperlexikon 516
 Makrostruktur 516, 520
 Megastruktur 516
 Mesostuktur 521
 Mikrostruktur 516, 520
 Lexikonimplementierung 515
 Lexikoninhalt 515
 Lexikonstruktur 515
 Lexikontheorie 518, **519**
 LFG, *siehe* Lexical Functional Grammar
 Likeability 674
 Likelihood-Funktion 143
 Likert-Skala 672
 LIMA-Korpus 487
 Linearisierung 634
 Linguistic Data Consortium 497
 Linguistische Datenverarbeitung . 2, 554
 Link 160, 165
 bidirektionaler 165
 typisierter 165
 Linker 470
 Links-rechts-Verarbeitung 304
 Linksrekursivität 306
 Lisp 467
 Lizenz 529
 LOB Corpus 487
 Logik 4, **28**, 535
 Lokale Baumgrammatik 163
 Lookahead 310
 LP-Statement 290
- M**
- manuelle Annotierung
 Übereinstimmung 152
 Fehlerarten 153
 Richtlinien 152
 systematische Fehler 153
 Markov-Prozess 132
 Markup-Sprache . **159–161**, 164, 165, 228,
 484
 Maschine 70
 Maschinelle Übersetzung, *siehe* Überset-
 zung, maschinelle
 maschinelle Sprachverarbeitung 2

- machine translation, *siehe* Übersetzung,
maschinelle
maximale Clique 603
Maximum-Likelihood-Prinzip 130, 151
Mean Reciprocal Rank 587
Meaning Text Theorie 440, 461
Medienallokation 630
Medizintechnik 619
Mehrbandautomat 207
Mehrfachvererbung 111
Mehrwortausdrücke
 Datenextraktion 568
Meinung 484
Menge 28
 leere 29
 regulär 70
 reguläre 182
 rekursive 92
 rekursiv aufzählbare 92
 ungeordnete 29
Mengendifferenz 31
Mengenkomplement 31
Mengenlehre 28
Mengenoperationen 30
Mengenschmitt 31
Mengenvereinigung 31
Mensch-Maschine-Interaktion 554
Merkmal 97, 179, 185, 191
 distinktives 204
Merkmalsstruktur 97, 286
 leere 102
 Tag 297
 Unterspezifikation 289
Merkmalsvektor 590, 592
Meronymie 505, 506, 548
Meronymierelation 506
Message Understanding Conference 404,
 409
Metadaten 162, 166, 167, 190, 193, 485,
 521
 Dublin Core 162
Metafunktion 463
 ideationale 463
 interpersonale 463
 textuelle 463
Metalexikographie 566
Metapher 381
Metonymie 380
Mikrofon 526
Mikrokosmos 543
Mimik 484
Minimal Edit Distance, *siehe* Levenshtein-
 Distanz
Minimal Recursion Semantics 377
MLE 151
Modalität 624
Modell 28
 der Prädikatenlogik 47
 der Typenlogik 58
Modelltheorie 28
Modifikation 343
modus ponens 38, 469
modus tollens 38
monostratal 298
monotone Operation 107
Montague-Semantik 339
Morph 238
Morphem 172, 186, 189
 abstrakt 241
 freies 237
 gebundenes 237
 peripheres 237
Morphemalternante 238
Morphemstrukturregel 207
Morphologie 12, 226, 236, 557
 autosegmentale 243
 prosodische 243
Morphophonem 181, 194, 195, 204
morphophonemisch 239
MRR, *siehe* Mean Reciprocal Rank
MRS, *siehe* Minimal Recursion Semantics
MTT, *siehe* Meaning Text Theorie
MUC, *siehe* Message Understanding Con-
ference
Multi-Nucleus 396
multimediales System 515
Multimenge 29
multimodales System 515
Multimodalität 425
Musterabgleich 563
Mutterknoten 96
mögliche Welten 333
- N**
- .Net 472
N-Gramm125, 131, 222, 270, 546, 558, 560
Naive Bayes 592
Named Entity Recognition 596
Namespaces 165
natürliche Klasse 183, 205
natürlichsprachliche Systeme 554
natural language processing, *siehe* maschi-
nelle Sprachverarbeitung
Natural Language Toolkit 478
Negation 34–35, 112
Negationstest 417
Negra Baumbank 498
NER, *siehe* Named Entity Recognition
Neuronales Netz 143, 534
Neutralisierung 240
Nichtterminalsymbol 67, 283
Nichtwort-Korrektur, *siehe* Rechtschreib-
korrektur von Nichtwörtern
NIST – National Institute of Standards
and Technology 676
NLG-Systeme, *siehe* Generierungssysteme

- NLTK, *siehe* Natural Language Toolkit
 Noisy Channel Model 559
 NomBank 488
 Nominalphrase
 quantifizierende 346
 Normalisierung 589
 Normung von Sprachressourcen 572
 Nucleus 396
 Nuklearskopos 348
 Nykvist-Frequenz 199
 Nykvist-Theorem 199, 525
- O**
 O-Notation 139
 Oberflächenrealisierung 437, **458-461**
 lexemgesteuerte 459
 strukturgesteuerte 459
 Obermenge 30
 Obliqueness-Hierarchie 298
 Onomastikon 538
 Onomatopoeia 549
 Ontologie. 5, 166, 191, **532**, 539, 550, 584, 631, 637
 Ontology learning 541
 OOV-Rate 668
 opaker Kontext 338
 Open Language Archive Community 486
 Optimalitätstheorie 3, 171, **185-186**, 206, 209
 Optimum
 globales 146
 lokales 146
 Orthographie 183-184
 Oszillogramm 177, 198
 Overlay 631
 Overstemming 589
 OWL 540, 637
- P**
 Pairwise Variability Index 202
 Paradigma 236
 paradigmatische Relation 180
 PARADISE-Framework 673
 Parallelkorpus **487**, 568, 571
 Parser 167
 Chunk-Parser 167
 statistischer **326-328**, 492, 495, 496
 XML-Parser 163, 164, 167
 Parsing 13, 149, 226, **303**
 Chunk-Parsing 275
 deterministisches 313
 partielles 276
 shallow 276
 statistisches 326-328
 Part of Speech 484
 Part-of-Speech-Tagger, *siehe* Tagger
 passage retrieval 580
 PATR-II 287
- pattern matching, *siehe* Musterabgleich
 Paula-Format 485
 Pause 195
 Penn Arabic Treebank 497
 Penn Chinese Treebank 497
 Penn Discourse Bank 488
 Penn Treebank 407, 488, 493, **496**
 Periode 197, **198**
 Perl 475
 Perplexität 667
 Persona 671
 Pfad 95
 Pfadgleichung 287
 Pfadäquivalenz 100
 Phase 197-199
 Phomen 189
 Phon 179
 Phonation 174
 Phonem **178-181**, 184, 186, 195
 Phonetik . 12, 172, 173, 178, 179, 189, 191
 akustische 172, **176**
 artikulatorische 172, **173**
 auditive 172, **178**
 klinische 178
 Ohrenphonetik 191
 Rezeptionsphonetik 172
 Phonologie . 12, 170, 172, 173, 178, 181, 183, 184, 186, 191, 226
 Autosegmentale 181, **185**
 Generative 181, **185-186**, 204, 208
 Lexikalische 185
 Metrische 185
 Natürliche 185
 Phonologien 186
 Phrase 281
 Phrasenstruktur, *siehe* Grammatik, kontextfreie
 Phrasierung **188**, 226
 Pivot 459
 Planoperatoren 396
 growth points 396
 Plug 416
 Polysemie 379, 548
 POS-Tagger, *siehe* Tagger
 POS-Tagging, *siehe* Wortartenannotation
 Potenzen
 Wort 67
 Potenzmenge 30
 PP-Anbindung 301
 Prädikat 45
 Prädikatenlogik 4, **45-53**
 Prädikation 341
 Präfix 238
 Prämissé 38
 Präsentationsplanung 630
 Präsupposition 394, 410, **415**, 430
 Präsuppositionsakkommmodation 419

- Präsupositionsauslöser 417
 Präsupositionsbindung 417
 Präsupositionsprojektion 416
 Präsupositionsresolution 419
 Präsupositionstest 417
 Präterimal 283
 Präzision 155, 490, 510, 563, **585**
 Prager Dependenzbaumbank 497
 Pragmatik 13, 188, 394
 Precision, *siehe* Präzision
 Pre-recorded prompts 621
 Primärdaten 483, 487
 Produktion
 geteilte 318
 Produktionsphonetik 172
 Programmentwicklung
 korpusbasierte 490
 Programmierschnittstelle 627
 Programmiersprache
 dynamische 475
 funktionale 468
 objektorientierte 470
 Projektion 292
 Prolog 113, 469
 Pronomen
 nicht-referentielle 403
 Reflexiv- 403
 PropBank 488
 Proposition 333
 Prosodie 172, 181, **186**, 189, 226, 229,
 233-234
 Satzprosodie 188
 prosodische Hierarchie 178
 Protologismus 549
 prozedural 298
 Pruning 307
 PVI 202
 Python 477
- Q**
- Qualiastruktur **388**, 392
 Qualitätstest 489
 Qualität
 Benutzungsqualität 663
 interne Qualität 663
 Softwarequalität 662
 qualitative Methode 191
 Qualitätsmerkmal
 Benutzungsqualität 663
 Usability
 Effektivität 664
 Produktivität 664
 Sicherheit 664
 Zufriedenheit 664
 Qualitätsmerkmal
 Übertragbarkeit 663
 Änderbarkeit 663
 Effizienz 663
- Funktionalität 663
 Usability 663
 Zuverlässigkeit 663
 Qualitätskriterien 495
 Quantifying In 352
 Quantisierung 525
 quantitative Methode 192
 Quantor 46, **346**
 Allquantor 47
 Existenzquantor 47
 generalisierter 357, 370
 konservativer 355
 monotoner 356
 Quelle-Filter-Modell 173
 Quellsprache 642
 Query Expansion 590
 Query Logs 561
 Query-Dichte 670
 Question Answering, *siehe* Fragebeantwortung
- R**
- RDF 162, **166**
 RDF-Vokabular 166
 RDF Schema, *siehe* RDFS
 RDFS 162, **166**
 Recall 155, 490, 510, 563, 586
 Rechtsableitung 80
 Rechtschreibkorrektur
 für Suchmaschinen 561
 kontextabhängige 559
 mit *Noisy Channel Model* 559
 statistikbasierte 560
 von Nichtwörtern 556
 Redehintergrund 415
 Reduktion 305
 Redundanzregel 204, 207
 Referent 399
 Referenz 335
 Referenzausdruck **399**, 437, 441, 456
 Referenzkorpus **150**, 489
 Regel 534
 phonologische 183
 Regelmenge 67
 Registertheorie **426**, 429
 regulärer Ausdruck 71, 205
 reguläres Modell 171, 205
 Relation 534, 581
 konverse 506
 konzeptuelle 505
 lexikalische 505
 semantische 595
 syntagmatische 180
 Relationsextraktion 599
 Relax NG 163
 Relevance Feedback 590
 Reliabilität 673
 RES – Remote Evaluation System 675

- Resource Description Framework, *siehe*
RDF
- Ressourcen 566
computerlinguistische 481
- Ressourcen-Interoperabilität 573
- Restriktor 348
- Retrodigitalisierung 570
- Rhetorical Structure Theory 161, **396**, 446
- Rhythmus 186, 188, **202-203**
- RMS-Amplitude 197, **198**
- Robustheit 626
- Rocchio 592
- RST, *siehe* Rhetorical Structure Theory
- Russells Mengenparadoxon **32**, 54
- S**
- Salienz 400
- Salienzfaktoren 403
- SAMPA 176, 177, 196
- Samplerate 525
- SAMPROSA 196
- Satellit 396
- Satz **47**, 172
 Akzent 188
- Satzalignment 571
- Satzmodus 226
- Satzplanung 437, **453**, 634
- Satzprosodie 188
- Satzsemantik 330
- SAX 164
- Scalable Vector Graphics, *siehe* SVG
- Schallfilter 173
- Schallquelle 173
- Schematron 163
- Schreibvariante 548
- Schriftsysteme
 nicht-segmentierte 264
 segmentierte 264
- Schwesterknoten 96
- Schönfinkel-Darstellung 53
- Seeds 597
- Segment 526
- Segmentierung 483, **526**
- selektive Bindung 391-392
- Seltene Ereignisse 229-233
- Semantic Web 532, **540**, 546, 637
- Semantik 12, **330**
 dynamische 331, **361**
 formale 331, **339**
 lexikalische 330
- Semantikkonstruktion 334
- semantische Rolle 484
- semantische Verwandtschaft 550
- semantischer Frame 484
- semantischer Typ 389
 komplexer Typ **389**, 390
- semantisches Netz 5, 166, **535**
- semi-überwachtes Verfahren 597
- semi-automatische Annotation 493
- Semiotik **179**, 189
- Sensus 542
- SGML **159**, 160, 166, 569
 SGML-Instanz 160
 SGML-Prozessor 160
 WebSGML 160
- SHRDLU 466
- Signal 524
- Signalanalyse 216
- Signalverarbeitung 199
- Silbe **178-182**, 187, 206, 208
- Silbenstruktur 227
- Simple API for XML, *siehe* SAX
- Simple Object Access Protocol, *siehe*
SOAP
- Single Source Publishing 159
- Sinn 338
- Skala 413
- Sketch Engine 574
- Skopus **46**, 348
- Skopusbeweglichkeit 337, 350, **353**
- Skriptsprache 196, **475**
- Slang 549
- Smartkom-Korpus 488
- Smoothing 155, 274
 Add-λ 156
 Add-One 155
 Back-Off 156
 Good-Turing 156
 Interpolation 156
 Laplace 155
 OOV-Methode 156
- Snippet 546, **579**, 608, 611
- SOAP 165
- Sozio-Semantisches Web 546
- Sparse-Data-Problem **270**, 328
- Speicherkomplexität 90
- Spektralanalyse 200
- Spektrogramm 177, **200**
- Spektrum 177, **200**
- Spezialkorpus 489
- Spezifikation 528
- Sprachaufnahme 190, **193**, 526
- Sprachdaten 483
- Sprachdatenbank 483, **524**
- Sprache
 schwach kontextsensitiv 84
 allgemeine Regelsprache 84, **87-88**
 deterministisch kontextfrei 81
 kontextfrei 80
 kontextsensitiv 84, **85**
 kontrollierte 657
 reguläre 70-71
 rekursive 92
 rekursiv aufzählbare **88**, 92
 schwach kontextsensitive 9, **86**
 Typ-3 71

- Spracherkennergrammatik 628
Spracherkennung 131–132, 184, 192, **215–223**, 628
 Merkmalsextraktion 216
 Merkmalsvektor 131
 Musterabgleich 216
 Problem der 131
 unit matching, *siehe* Untereinheitenvergleich
 Untereinheitenvergleich 217
 Vorverarbeitung 216
Sprachinterpretation 628
Sprachlaut 525
Sprachmodell131, 197, **221–222**, 628, 665
 backoff 222
 discounting 222
 statistisch 221
Sprachproduktion 223, 234
Sprachregister 545
Sprachressourcen 566
Sprachsignal 179, 199, 200, 483, **524**
Sprachsynthese 184, 192, **223–229**, 630
 Algorithmen 232
 Architektur 223
 canned speech 228
 endliche Automaten 227
 Formalismus 227
 Komponenten 223
 konzeptbasierte 228
 Korpusdesign 231
 lexikalische Analyse 225
 Lexikon 225
 linguistische Analyse **224**, 225, 230
 multilinguale 228
 Parsing 226
 Part-of-Speech 226
 Phonologie 226
 Prosodie 226, 229
 Qualität 223, 224, 228, 229
 Schnittstellen 229
 sliced speech 228
 Suche 233
 Tagging 226
 Textanalyse 224
 textbasierte 224, 228
 Textnormalisierung 225
 Textvorverarbeitung 225
 Unit Selection 232
 Wortmodell 226
Sprachtechnologie .. **2**, 178, 195, 196, 481, 553
Sprachtypologie 187
Sprachunterricht 491
Sprachverarbeitung
 ngindent10pt symbolische 19
Sprachverständigen 218
Sprechakttheorie 330
Sprecherkennung 192, **619**
Sprechergruppenerkennung 619
Sprechervielfalt 223
Sprecherzustandserkennung 619
Sprechstil 188
Sprichwort 549
SQL 636
Stamm 238
Stammformbildung 589
Standard Generalized Markup Language,
 siehe SGML
standing query 591
Standoff-Format 485
Startsymbol **67**, 163, 283
statistisches Modell
 Evaluation 154–155
 generatives Modell 147
 Glättung, *siehe* Smoothing
 Hidden-Markov-Modell, *siehe* HMM
 Meta-Parameter 156
 Modellwahrscheinlichkeit 130
 N-Gramm-Modell 125, 131, **222**
Stelligkeit 45
Stemming 589
Stereotyp 431
Stern von Sigma 66
Stopwort 383
Stopwortliste 588
Stratum 298
string distance function, *siehe* Stringabstandsfunction
Stringabstandsfunction 557
Strukturalismus 184
Strukturbau 281
Strukturteilung 100
STTS, *siehe* Stuttgart-Tübingen tagset
Stuttgart-Tübingen Tagset .. **150**, 272–273, 484, 498
Style Sheet 160
Stylebook 495, 496
Subkategorisierung **286**, 296
 Datenextraktion 568
Subkategorisierungsprinzip 297
Subsprache, Sublanguage 639
Subsumption **101**, 111
Subtyp 111
Suchanfrage 589
Suchmaschine 474, 479, 517, 523, 551, 561
Suchraum 303
Suchstrategie 306
Suchtool 502
Suffix 238
summarization 611
 multidocument 612
 query-biased 612
Supertyp 111
Support Vector Machine 592
Supremum 106
SVG 165

- SVM, *siehe* Support Vector Machine
 Syllabifizierung 227, 231
 Symbol
 diakritisches 243, 249
 Symbolsystem 524
 Symbolvorrat 45, 55
 synkategorematisch 359
 Synkretismus 236
 Synonymie 379, 505, 548
 Sympathy 493
 Synset 504
 Syntax3, 12, 34, 45, 55, 61, 163, 281, 280, 334, 340, 487, 665
 Syntaxregel 163, 167
 Systemisch Funktionale Grammatik 461, 463
 Systemkontrolle 628
 Systemnetzwerke 463
 Szenario-Template-Filling 602
- T**
- T-Box 538
 Tableaux
 aussagenlogisches 40
 Expansionsregel 41
 geschlossenes 43
 prädikatenlogisches 52
 Zweig 42
 TAG, *siehe* Baumadjunktions-Grammatik
 Tag
 semantischer 548
 Tagger 271
 Tagging ... 9, 22, 114, 149, 226, 271-275, 303, 404, 596, 600, 610
 part-of-speech 114, 149, 226, 271-275, 303, 404, 596, 600, 610
 Wortarten-, *siehe* part-of-speech
 Tagset 150, 271, 484
 Tailoring 422, 635
 Takt 178
 Tautologie 37, 44, 50
 Taxonomie 535
 Technische Dokumentation, Generierung von 636
 TEI, *siehe* Text Encoding Initiative
 Teilmenge 30
 Telefoniesysteme 617
 Telefonüberwachung 618
 Template 164
 Tempus 484
 Term 46, 588
 Termextraktion 570
 Termfrequenz 588
 Terminalsymbol 67, 163, 283
 Terminalton 188, 189
 Terminographie 566
 Terminologie 566
 Terminologiedatenbank 566
- Tertium non datur 37, 349
 Test
 Feldtest 662
 Labortest 662
 Pilottest 662
 Regressionstest 662
 Wizard-of-Oz Test 662
 Testkorpus 490
 Testverfahren
 dynamische Testverfahren 661
 Blackbox-Test 661
 Glass-Box-Test 661
 White-Box-Test 661
 Text 395
 Text Encoding Initiative 167, 484, 486, 488
 Textarchiv 482
 textbasierte Fragebeantwortung 606
 textbasiertes Informationsmanagement 577
 Textgenerierung, *siehe* Generierung
 Textgruppierung 584
 Textklassifikation 149, 550, 591
 Textkorpus 483
 Textplan 436
 Textplanung, *siehe* Diskursplanung
 textQA 606
 Textschema 432, 446
 Textsorte 167, 634, 635
 Textstruktur 483
 Texttechnologie 17, 159, 167, 554
 text-to-speech 223
 Textuelle Datenbanken 167
 Textzusammenfassung 583, 611
 tf-idf 588
 grep 502
 grep2 502
 Theorem 43
 Theorieformalismus 287
 Thesaurus 548
 Tiefensuche 306
 iterative 307
 TIGER
 Korpus 498
 Projekt 495
 TIGER-XML 495
 TIGERSearch 502
 TIM, *siehe* textbasiertes Informationsmanagement
 time-series data 635
 TimeBank 488
 ToBI 196
 Tochterknoten 96
 Token 520
 Tokenisierung 264
 Ton 186, 187, 189, 210
 Tonsprache 209, 210
 Top Ontologie 508
 Topic maps 166, 541

- Topik 591
 topologische Felder 484, 500
 Training
 überwachtes 149–151
 unüberwachtes 149
 Trainingskorpus 490
 Transducer, *siehe* Transduktör
 Transduktör 78, 171, 227, 246, 557
 endlicher 171, 205–207
 Transfer 645
 ~regel 646
 Transkription 190, **194–196**, 483
 Transkriptionssystem 618
 Tree Adjoining Grammar, *siehe* Grammatik, BaumadjunktionsGrammatik
 treebank, *siehe* Baumbank
 Treebanker 493
 tregex 502
 Trigramm **270**, 488
 Triphon 195
 TTS, *siehe* text-to-speech
 TüBa-D/S 502
 TüBa-D/Z 407, **409**, 485, 496, 500
 Turingmaschine 85, **88**, 93
 Typ 32, 54, **55**, 109
 Typ-0-Sprache, *siehe* Sprache, allgemeine
 Regelsprache
 Typ-1-Sprache, *siehe* Sprache, kontextsensitiv
 Typ-2-Sprache, *siehe* Sprache, kontextfrei
 Typ-3-Sprache, *siehe* Sprache, regulär
 Typ-Unifikation 112
 Typanhebung **347**, 391
 Type 520
 Type-Token-Ratio 520
 Typenlogik 53
 Typerzung 391
 typisierte Merkmalsstruktur 109
 Typkonstruktör 389
 Typverschiebung 391
- U**
 Überbeantwortung 424
 Übergangsfunktion **74**, 75, 81, 89
 ε -Übergang 76
 Übergangsnetzwerk 205–206
 Übersetzung 642
 ambiguitätserhaltend 371
 beispielbasierte 571
 computer-gestützte 643
 computergestützte 642
 gesprochener Sprache 620
 maschinelle 280, 550, 613, **642**, 675
 statistische 571, **647**
 Übersetzungsdivergenz 644
 Übertragungsphonetik 172
- UDRS, *siehe* Diskursrepräsentationsstruktur
 unärer Knoten 500
 Understemming 589
 Unicode 167
 Unifikation **104**, 171, 631
 Unifikationsgrammatik **110**, **286**
 Unigramm 270
 Unit-Selection 622
 Universalie 178
 Universum 48
 einer DRS 362
 Unterbaum 96
 Untergraph 95
 Unterricht 493
 Unterspezifikation 98, 332, **371**, 379
 Hole 373
 Label 373
 Subordination 374
 Urheberrecht 486, 529
- V**
 Validierung 529
 Validität 163, 164, 673
 Variable **45**, 67
 freie 47
 gebundene 46
 Variablenbelegung
 der Aussagenlogik 35
 der Prädikatenlogik 49
 der Typenlogik 57–58
 Vektorraummodell 589
 Verarbeitungsrichtung 304
 bottom-up 305
 top-down 305
 Verband 107
 Verbmobil 309, 487, 502, 642, 676
 Vererbung **111**, 166, 205, 535
 lexikalische 391
 Vererbungshierarchie 110
 Vergleichskorpus 487
 Vergleichsmaßstab **267**, 400
 Verkettung, *siehe* Konkatenation
 Verknüpfungstafel 36
 Verwechlungsmengen 560
 Visem 517
 Viterbi
 Algorithmus **140**, 143, 146, 233
 Bewertung 146
 Training 146–147
 Voice User Interface (VUI) 661
 VoiceXML 627
 Vokabular
 non-terminales 163
 RDF-Vokabular 166
 terminales, *siehe* Terminalsymbol
 XHTML-Vokabular 160
 Vokabularabdeckung 668

- Vokal 173, **175–176**, 200
 Vollformenlexikon 236
 vollständiger Pfad 99
 Vollständigkeit 44, 563, 585
 Volltextsuche 584, **587–590**
 Vorrangregel 71
 Vorwissen 424
- W**
- W3C **161**, 165, 168
 Standard 165
 WaCky 488
 Wahrheitsbedingung 28, **332**
 Wahrheitswert 33
 Wahrheitswertetafel 36
 Wahrscheinlichkeit
 bedingte 115, **120**
 Formel von Bayes, *siehe* Bayes-Formel
 Kettenformel 124
 Wahrscheinlichkeitsmaß 117
 bedingtes 120
 Wahrscheinlichkeitsraum
 diskreter 115, **117**
 Laplace-Raum 119
 Wall Street Journal Subkorpus 488–489
 Web Services 165
 webbasierte Fragebeantwortung 608
 webQA 608
 Webseite 167
 WebSGML 160
 Werkzeugformalismus 287
 Wettervorhersagen, Generierung von 635
 WHG-Dichte 667
 Widerlegungsverfahren 40
 Wiederverwertbarkeit 481, 494–496
 WikiMining 547
 Wikipedia **547**, 598
 Wikipedia-Dumps 489
 Wiktionary 547, **548**
 Wissen **533**, 534, 643
 assertionales 538
 Default- 537
 Fakten- 538
 nicht-sprachliches 532
 ontologisches 532
 taxonomisches 535
 Wissensakquisition 427, 430
 wissensbasierte Systeme 538
 Wissensbasis 538
 Wissensquellen 598
 Wissensrepräsentation 4, 252, 389, 505, **533**, 534, 536, 539
 allgemein 533
 Default 537
 epistemologische Ebene 536
 Frames 535
 konzeptuelle Ebene 536
 ontologische Ebene 540
 semantisches Netzwerk 389, **535**
 terminologische Logik 389, **540**
 wohlgeformter Ausdruck
 der Lambda-Typenlogik 61
 der Typenlogik 55
 Wohlgeformtheit 164
 word alignment 571
 Word Sense Disambiguation, *siehe* Lesartendisambiguierung
 WordNet382, 407, 478, **504**, 541, 566, 605
 World Wide Web, *siehe* WWW
 World Wide Web Consortium, *siehe* W3C
 Wort172, 178, 179, 181, 182, 195, 210, 236
 Länge eines Worts 66
 leeres Wort 66
 unbekanntes 223
 Wortakkuratur 666
 Wortalignment 571
 Wortart 484, 494
 Wortartenannotierung 150
 Wortbedeutung 484
 Variabilität von 379
 Wortbildung 230, 237
 Wörterbuch 566
 interaktives 570
 multilinguale 549
 web-basiertes 548
 Wortfehlerrate 666
 Wortform 236
 Wortgrenze 225
 Worthypothesengraph 665
 Wortnetz 550
 lexikalisch-semantisches 504
 Wurzel 96, 238
 WWW 24, 160, 165, 167, 553
 WYSIWYM 636
- X**
- X-Bar-Schema 292
 XHTML **160**, 164
 XML **161–163**, 167, 481, 484, 485, 569
 Betrachtung 164
 Containerelement 162, 163
 Datenelement 162, 163
 DocBook 164
 Dokumentinstanz 159–165, 167
 Inhaltsmodell 161
 Konnektor 161
 Okkurrenzindikator 161
 Relax NG 163
 Schematron 163
 Validität 163, 164
 Web Services 165
 Wohlgeformtheit 164
 Wurzelement 161, 163, 165
 XLink 165
 XML Query 165
 XML Schema 163, 164

XML Topic Maps 166
XML-Anwendung 159
XML-Datenbank 165
XML-Parser 163, 164, 167
XML-Werkzeug 163
XPath 164, 165
XSL 159, 164
XSL Transformations, *siehe* XSLT
XSL-FO 164
XSLT 159, 164, 165
 Template 164
 XSLT-Prozessor 164

Y

y-Abschnitt 198

Z

Zeichen 66, 179
Zeichensatzkodierung 167
Zeitbäume 203
Zeitdomäne 177, **197**
Zeitfenster 200
Zeitkomplexität 90
Zentroid 594
Zielsprache 642
Zipfsches Gesetz 129, 155, **230**
Zirkumfix 250
Zustandstafel 76
Zwei-Ebenen-Morphologie 208, 557
Zwei-Ebenen-Phonologie 208
Zwischenrepräsentation 340
Zyklus 95

Die Autorinnen und Autoren

Jan W. Amtrup
Kofax Image Products
16245 Laguna Canyon Road
Irvine, CA 92618-3603
jamtrup@cox.net

John Bateman
Universität Bremen
FB10, Sprach- und Literaturwissenschaften
28334 Bremen
bateman@uni-bremen.de

Tilman Becker
DFKI GmbH
Intelligente Benutzerschnittstellen
66123 Saarbrücken
tilman.becker@dfki.de

Kai-Uwe Carstensen
Ruhr-Universität Bochum
Sprachwissenschaftliches Institut
44780 Bochum
carstensen@linguistics.rub.de

Christoph Draxler
Ludwig-Maximilian Universität München
Abteilung für Phonetik und Sprachkommunikation
Schellingstr. 3
80799 München
draxler@phonetik.uni-muenchen.de

Christian Ebert
Universität Tübingen
Seminar für Sprachwissenschaft
Wilhelmstraße 19
72074 Tübingen
christian.ebert@uni-tuebingen.de

Cornelia Ebert
Universität Osnabrück
Institut für Kognitionswissenschaft
49076 Osnabrück
cornelia.ebert@uni-osnabueck.de

Stefan Evert
Universität Osnabrück
Institut für Kognitionswissenschaft
49076 Osnabrück
stefan.evert@uni-osnabueck.de

Volker Fischer
European Media Laboratory GmbH
Villa Bosch
Schloß-Wolfsbrunnenweg 33

69118 Heidelberg
volker.fischer@eml-d.villa-bosch.de

Gerhard Fliedner
Trappentreustr. 31
80339 München
gerd@fliedner.name

Bernhard Frötschl
Freie Universität Berlin
Institut für Informatik
Takustr. 9
10963 Berlin
froetsch@inf.fu-berlin.de

Dafydd Gibbon
Universität Bielefeld
Fakultät für Linguistik und Literaturwissenschaft
Postfach 100131
33501 Bielefeld
gibbon@uni-bielefeld.de

Iryna Gurevych
Technische Universität Darmstadt
Fachbereich Informatik
Hochschulstr. 10
D-64289 Darmstadt
gurevych@tk.informatik.tu-darmstadt.de

Udo Haiber
DaimlerChrysler AG
Sprachverstehende Systeme (FT3/AV)
Wilhelm-Runge-Str. 11
D-89081 Ulm
udo.haiber@daimlerchrysler.com

André Hagenbruch
Universitätsbibliothek Bochum
44801 Bochum
andre.hagenbruch@rub.de

Ulrich Heid
Universität Stuttgart
Institut für Maschinelle
Sprachverarbeitung
Azenbergstrasse 12
70174 Stuttgart
ulrich.heid@ims.uni-stuttgart.de

Helmut Horacek
Universität des Saarlandes
FB 14 - Informatik
66041 Saarbrücken
horacek@ags.uni-sb.de

- Hans-Peter Hutter
 ZHAW Zürcher Hochschule
 für Angewandte Wissenschaften
 Institut für Angewandte Informationstechnologie
 Steinberggasse 13
 CH-8400 Winterthur
 hans-peter.hutter@zhaw.ch
- Gerhard Jäger
 Universität Tübingen
 Seminar für Sprachwissenschaft
 Wilhelmstraße 19
 72074 Tübingen
 gerhard.jaeger@uni-tuebingen.de
- Susanne J. Jekat
 ZHAW Zürcher Hochschule
 für Angewandte Wissenschaften
 Departement Angewandte Linguistik
 IUED Institut für Übersetzen und Dolmetschen
 Theaterstr. 15c
 CH-8401 Winterthur
 susanne.jekat@zhaw.ch
- Ralf Klabunde
 Ruhr-Universität Bochum
 Sprachwissenschaftliches Institut
 44780 Bochum
 klabunde@linguistics.rub.de
- Peter Kolb
 Institut für Linguistik
 Universität Potsdam
 Karl-Liebknecht-Str. 24-25
 D-14476 Golm
 peter@linguatoools.de
- Sandra Kübler
 Indiana University
 Department of Linguistics
 Memorial Hall 322
 1021 E. Third Street
 Bloomington IN 47405
 skuebler@indiana.edu
- Claudia Kunze
 Universität Heidelberg
 Seminar für Computerlinguistik
 Im Neuenheimer Feld 325
 69120 Heidelberg
 kunze@cl.uni-heidelberg.de
- Hagen Langer
 Universität Bremen
 Technologie-Zentrum Informatik
- Bereich Intelligente Systeme
 Postfach 33 04 40
 28334 Bremen
 hagen.langer@tzi.de
- Wolf Lindstrot
 Wühlischstraße 42
 10245 Berlin
- Wolfgang Menzel
 Universität Hamburg
 Department Informatik
 Vogt-Kölln-Str. 30
 22527 Hamburg
 menzel@informatik.uni-hamburg.de
- Bernd Möbius
 Universität Bonn
 Institut für Kommunikationswissenschaften
 Poppelsdorfer Allee 47
 53115 Bonn
 moebius@ifk.uni-bonn.de
- Günter Neumann
 DFKI GmbH
 Sprachtechnologie
 66123 Saarbrücken
 neumann@dfki.de
- Elmar Nöth
 Universität Erlangen-Nürnberg
 Lehrstuhl für Mustererkennung
 Martensstr. 3
 91058 Erlangen
 noeth@informatik.uni-erlangen.de
- Cécile Paris
 Intelligent Interactive Technology
 CSIRO/Mathematical and
 Information Sciences
 Locked Bag 17, North Ryde, NSW 1670
 Building E6B
 Macquarie University
 North Ryde Australia
 cecile.paris@cmis.csiro.au
- Georg Rehm
 vionto GmbH
 Karl-Marx-Allee 90a
 10243 Berlin
 georg.rehm@vionto.com
- Michael Schiehlen
 Universität Stuttgart
 Institut für Maschinelle Sprachverarbeitung
 Azenbergstrasse 12
 70174 Stuttgart
 mike@ims.uni-stuttgart.de

Michael Strube
EML Research GmbH
Schloss-Wolfsbrunnenweg 33
69118 Heidelberg
strube@emi-research.de

Jochen Trommer
Universität Leipzig
Institut für Linguistik
Beethovenstraße 15
D-04107 Leipzig
jtrommer@uni-leipzig.de

Martin Volk
Universität Zürich
Institut für Computerlinguistik
Binzmühlestrasse 14
CH-8050 Zürich
volk@ifi.unizh.ch

Heike Zinsmeister
Universität Konstanz
Fachbereich Sprachwissenschaft
Fach D 185
D-78457 Konstanz
heike.zinsmeister@uni-konstanz.de