

Günther Görz, Josef Schneeberger,
Ute Schmid (Hrsg.)

Handbuch der Künstlichen Intelligenz

5. Auflage





Handbuch der Künstlichen Intelligenz

von

Prof. Dr. Günther Görz
Friedrich-Alexander-Universität
Erlangen-Nürnberg

Prof. Dr. Josef Schneeberger
Technische Hochschule Deggendorf

Prof. Dr. Ute Schmid
Otto-Friedrich-Universität Bamberg

5., überarbeitete und aktualisierte Auflage

Oldenbourg Verlag München

Lektorat: Johannes Breimeier
Herstellung: Tina Bonertz
Grafik: Irina Apetrei
Einbandgestaltung: hauser lacour

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

Library of Congress Cataloging-in-Publication Data

A CIP catalog record for this book has been applied for at the Library of Congress.

Dieses Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, des Vortrags, der Entnahme von Abbildungen und Tabellen, der Funksendung, der Mikroverfilmung oder der Vervielfältigung auf anderen Wegen und der Speicherung in Datenverarbeitungsanlagen, bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Eine Vervielfältigung dieses Werkes oder von Teilen dieses Werkes ist auch im Einzelfall nur in den Grenzen der gesetzlichen Bestimmungen des Urheberrechtsgesetzes in der jeweils geltenden Fassung zulässig. Sie ist grundsätzlich vergütungspflichtig. Zu widerhandlungen unterliegen den Strafbestimmungen des Urheberrechts.

© 2014 Oldenbourg Wissenschaftsverlag GmbH
Rosenheimer Straße 143, 81671 München, Deutschland
www.degruyter.com/oldenbourg
Ein Unternehmen von De Gruyter

Gedruckt in Deutschland

Dieses Papier ist alterungsbeständig nach DIN/ISO 9706.

ISBN 978-3-486-71307-7
eISBN 978-3-486-71979-6

Vorwort

Die vorliegende Ausgabe des „Handbuchs der Künstlichen Intelligenz“ ist die fünfte, wesentlich überarbeitete und erweiterte Auflage eines Werks, das zuerst 1993 als „Einführung in die Künstliche Intelligenz“¹ erschienen war.

Mit diesem Handbuch wird eine repräsentative Übersicht über die wissenschaftliche Disziplin der „Künstlichen Intelligenz“ (KI) vorgelegt, deren Autoren ausschließlich dem deutschsprachigen Raum entstammen. Als sich abzeichnete, dass die Lagerbestände der vierten Auflage zu Ende gehen, stellte sich wiederum die Frage, welche Anforderungen an eine Neuauflage angesichts der Fortschritte in der Wissenschaft und der Veränderungen im wissenschaftsorganisatorischen, technischen und wirtschaftlichen Umfeld zu richten seien. Nach immerhin neun Jahren lag es nahe, dass es nicht nur um eine Aktualisierung der Beiträge aus der letzten Auflage gehen konnte, sondern vielmehr um eine in mehrfacher Hinsicht thematisch ergänzte und abgerundete Neufassung. Hierbei verdanken wir wichtige Impulse der neuen Mitherausgeberin Ute Schmid, Bamberg, die an die Stelle von Claus-Rainer Rollinger trat; er musste aufgrund der Arbeitsbelastung als Präsident der Universität Osnabrück seine Mitarbeit einstellen. Nach wie vor trägt die Fachgruppe 1 „Künstliche Intelligenz“ der Gesellschaft für Informatik (GI) e.V. die Herausgabe dieses Werks mit. Der Oldenbourg-Verlag, München, hat die gesamten Vorbereitungsarbeiten konstruktiv begleitet, wofür ihm ein herzlicher Dank gebührt.

Das Vorwort zur ersten Auflage charakterisierte das Werk folgendermaßen:

„Seine Herausgeber und Autoren haben sich das Ziel gesetzt, damit eine weithin als schmerzlich empfundene Lücke auf dem Lehrbuchsektor zu schließen. (...) Dass ein solches Lehrbuch ein dringendes Desiderat ist, ist unbestritten, denn zum einen ist die KI inzwischen an vielen unserer Universitäten – zumeist als Teilgebiet der Informatik – vertreten, andererseits decken die zum Teil hervorragenden Lehrbücher aus dem angelsächsischen Sprachraum das Gebiet nicht in allen Aspekten umfassend ab. Angesichts des raschen Fortschritts der Forschung lassen manche dieser Werke in ihrer Aktualität Wünsche offen. Zudem hat die KI in Deutschland und Europa durchaus eigenständige Sichtweisen und Ansätze entwickelt, die auch den vorliegenden Band geprägt haben.“

Dieses Buch verdankt sein Entstehen einer Serie von Frühjahrsschulen zum Thema Künstliche Intelligenz (KIFS), die seit 1982 jährlich – von einer Ausnahme abgesehen – bis 1996 von der Fachgruppe 1 „Künstliche Intelligenz“ der Gesellschaft für Informatik (GI) e.V. durchgeführt wurden. Ziel der Frühjahrsschulen war es, in der Form von Kursen, die jeweils in etwa einer zweistündigen Vorlesung entsprachen, eine breit angelegte moderne Einführung in das Fach sowie einen Überblick über aktuelle Forschungsgebiete zu bieten. Zu einigen der Frühjahrsschu-

¹ ursprünglich im Addison-Wesley Verlag, Bonn.

len wurden Tagungsbände vorgelegt, die die dort gehaltenen Kurse dokumentieren². Durch den Ausbau der KI an den deutschen Universitäten und Fachhochschulen hatte die Nachfrage nach der zentralen Bildungsaufgabe, die die KIFS erfüllt hatte, spürbar nachgelassen. Ihre Nachfolge hat seitdem ein von mehreren wissenschaftlichen Gesellschaften am selben Ort durchgeführtes „Interdisziplinäres Kolleg Kognitionswissenschaft“ angetreten.

Die Erfahrung mit den ersten Auflagen zeigte, dass viele der Kapitel in der Lehre eingesetzt wurden, sei es als Hauptreferenz oder als ergänzende Lektüre, selten aber das Werk in seiner Gesamtheit als Lehrbuch. Dies lag zum einen an seinem Umfang, zum anderen aber auch an der Vielfalt der eher einführenden und eher vertiefenden Kapitel, die es doch von der Geschlossenheit eines klassischen Lehrbuchs unterscheidet, das von einem einzigen Autor oder einem kleinen Autorenteam verfasst wurde. Zudem wurde vielfach angeregt, die thematische Breite des Werks zu vergrößern. Die Herausgeber hoffen, mit der Überarbeitung und der Weiterentwicklung in Richtung eines „Handbuchs“ dem veränderten Anforderungsprofil weitestgehend gerecht zu werden.

Für alle Beiträge dieses Werks war maßgeblich, dass sie eine straffe und qualitativ hochstehende Darstellung des jeweiligen Themengebiets geben und Hinweise auf notwendige und sinnvolle Vertiefungsmöglichkeiten, u.a. auch in der Form einer Auswahlbibliographie, bieten. Es ist offensichtlich, dass bei einem angestrebten Umfang von ca. 50 Seiten pro Beitrag nicht *alles*, was thematisch wichtig ist, behandelt werden kann.

Gegenüber der letzten Auflage sind folgende Änderungen erwähnenswert:

Das Handbuch ist in drei Teile gegliedert, *Grundlagen, Weiterführenden Theorien und Methoden* und *Anwendungen*, in die die Kapitel neu eingeordnet wurden. Neu sind die Kapitel Multigagentensysteme, Verkörperte Kommunikation mit kognitiven virtuellen Agenten, Universelle Spiele und Semantic Web. Komplett neu verfasst – z.T. auch von neuen Autoren – oder stark überarbeitet wurden die Kapitel Kognition, Wissensrepräsentation, automatische Inferenz (früher: Automatisches Beweisen), Nichtmonotoner Schließen, Constraints, Planen, Fallbasiertes und modellbasiertes Schließen, Neuronale Netze, Maschinelles Lernen und Data Mining. Die restlichen Kapitel wurden überarbeitet. Eine größere Veränderung hat sich im Themengebiet der Sprachverarbeitung ergeben, das nunmehr mit einem repräsentativen Übersichtskapitel vertreten ist. Der hauptsächliche Grund liegt darin, dass – inzwischen in der dritten – Auflage ein ausgezeichnetes umfassendes Handbuch „Computerlinguistik und Sprachtechnologie“ (Hg. K.-U. Carstensen et al., Heidelberg: Spektrum Akademischer Verlag, 2010) vorliegt, das von der Konzeption her durchaus dem vorliegenden Werk ähnelt. Andererseits hätte eine Aktualisierung des Themas Sprachverarbeitung im vorliegenden Werk eine Erweiterung um neue Teilgebiete bedingt, was zu einer deutlichen Steigerung des Umfangs und damit zu einer Aufteilung in zwei Bände geführt hätte. Weiterhin musste eine gesonderte Darstellung des Wissens über Raum und Zeit wegfallen, u.a. auch krankheitsbedingt in einer sehr späten Planungsphase des Werks. Anstelle einer Überarbeitung des Raumwissen-Kapitels verweisen die bisherigen Autoren auf ein im Erscheinen begriffenes „Handbook of Spatial Cognition“ (Hg. D. Waller und L. Nadel). Daraufhin haben sich bei einigen Kapiteln Veränderungen in der Zusammensetzung der Autorengruppe ergeben.

² Erschienen in der Reihe Informatik-Fachberichte (IFB) im Springer-Verlag, Berlin: Teisendorf 1982: IFB Nr. 59, Dassel 1984: IFB Nr. 93, Dassel 1985: IFB Nr. 159, Günne 1987: IFB Nr. 202, Günne 1989: IFB Nr. 203.

Wir danken ganz herzlich allen Autorinnen und Autoren, dass sie so engagiert an dieser nicht einfachen Aufgabe mitgewirkt haben. Neben den hohen qualitativen Anforderungen dieses ambitionierten Buchprojekts stellten die vielen notwendigen inhaltlichen Absprachen bis hin zu einer einheitlichen Layoutgestaltung und die – im großen und ganzen vorbildlich eingehaltene terminliche Disziplin eine nicht alltägliche Herausforderung dar. Leider konnten einige wenige geplante Artikel und Überarbeitungen krankheitshalber nicht realisiert werden, womit bei einem derart komplexen Unternehmen zu rechnen ist, was wir aber umso mehr bedauern.

Zur Qualitätssicherung wurde eine Referierungsprozedur durchgeführt: Jeder Beitrag wurde von externen Referenten und anderen Buchmitarbeitern gelesen und die kritischen Anmerkungen wurden den Autoren zur endgültigen Überarbeitung zugesandt. Besonderer Dank für die externe Begutachtung gilt folgenden Kolleginnen und Kollegen: Kai-Uwe Carstensen (Siegen), Stefan Edelkamp (Hamburg), Uli Furbach (Koblenz), Hans Guesgen (Neuseeland), Barbara Hammer (Bielefeld), Joachim Hertzberg (Osnabrück), Pascal Hitzler (Dayton, Ohio), Eyke Hüllermeier (Marburg), Frank Jäckel (Osnabrück), Martin Lauer (Karlsruhe), Torsten Schaub (Potsdam), und Daniel Sonntag (Saarbrücken).

Mirjam Minor veröffentlichte in der Zeitschrift „*Künstliche Intelligenz*“³ eine ausführliche Rezension mit einer Reihe wertvoller Anregungen die teilweise in der neuen Auflage umgesetzt werden konnten.

Erlangen, Nürnberg und Bamberg, im Mai 2013

Günther Görz
Ute Schmid
Josef Schneeberger

³ KI, Nr. 4, Oktober 2002.

Inhaltsverzeichnis

Vorwort	V
1 Einleitung	1
1.1 Zum Begriff Künstliche Intelligenz	2
1.2 Die Entwicklung der KI	4
1.3 Grundsätzliche Herangehensweisen	6
1.3.1 Symbolische Repräsentation – die Wissensebene	6
1.3.2 Verteilung und Situiertheit.....	9
1.3.3 Nicht-Symbolische Ansätze und Maschinelles Lernen.....	11
1.3.4 Verkörperung (Embodiment)	12
1.4 Teilbereiche und Anwendungsbereiche der KI	13
Literaturverzeichnis	16
I Grundlagen	
2 Kognition	21
2.1 Kognitionswissenschaft	22
2.1.1 Charakterisierung und historische Entwicklung	22
2.1.2 Methoden kognitionswissenschaftlicher Forschung	24
2.2 Menschliche Kognition	34
2.2.1 Wahrnehmung, Aufmerksamkeit, Bewusstsein, Handlungskontrolle	35
2.2.2 Mentale Repräsentation, Begriffe und mentale Modelle	38
2.2.3 Gedächtnis und Lernen	39
2.2.4 Denken und Problemlösen	44
2.2.5 Soziale Kognition	53
2.2.6 Sprache.....	57
2.2.7 Schlusswort.....	63
Literaturverzeichnis	64
3 Suche	75
3.1 Problemlösen als Suche	75
3.1.1 Problemrepräsentation mit Zuständen und Operatoren.....	75

3.1.2	Generische Suche	78
3.1.3	Suchstrategien und deren Bewertung	81
3.2	Uninformierte Suchverfahren	82
3.2.1	Breitensuche	82
3.2.2	Gleiche-Kosten-Suche.....	84
3.2.3	Tiefensuche.....	85
3.2.4	Schrittweise vertiefende Suche.....	88
3.3	Heuristische Suche	89
3.3.1	Heuristische Schätzfunktionen	89
3.3.2	Suche mit schrittweiser lokaler Verbesserung.....	91
3.3.3	Bestensuche	94
	Literaturverzeichnis	103
4	Wissensrepräsentation und -verarbeitung	105
4.1	Einleitung und Motivation	105
4.1.1	Wissen – wozu?.....	105
4.1.2	Wissensformen	106
4.1.3	Repräsentation	109
4.1.4	Wissensverarbeitung = Schlussfolgern	110
4.2	Deklarative Wissensrepräsentation	111
4.2.1	Wissensbasierte Systeme	111
4.2.2	Die Rolle der Logik	113
4.2.3	Schlussfolgerungstypen	114
4.3	Ein Beispiel: Beschreibungslogiken	115
4.3.1	Der Formalismus.....	115
4.3.2	Semantik	117
4.3.3	Inferenzdienste	118
4.3.4	Inferenzalgorithmen	119
4.3.5	Berechenbarkeitseigenschaften	123
4.3.6	Neuere Entwicklungen	125
4.4	Ausblick.....	126
	Literaturverzeichnis	127
5	Automatische Inferenz	129
5.1	Einleitung	129
5.2	Entwurf automatischer Inferenzsysteme	132
5.3	Prädikatenlogik erster Stufe	134
5.4	Normalformen	137
5.5	Das DPLL Verfahren für die Aussagenlogik	142

5.6	Aussagenlogische Resolution	145
5.6.1	Ein einfacher Resolutionskalkül	146
5.6.2	A-geordnete Resolution	149
5.6.3	Verfeinerungen des Resolutionsverfahrens	154
5.7	Kalküle für die Prädikatenlogik	154
5.7.1	Herbrand-Theorie	155
5.7.2	Prädikatenlogische Resolution	157
5.8	Weitere Betrachtungen	163
	Literaturverzeichnis	164

II Theorie und Methoden

6	Nichtmonotonen Schließen	171
6.1	Einführung	171
6.2	Formalisierungen nichtmonotonen Schließens	176
6.2.1	Default Logik	177
6.2.2	Autoepistemische Logik	183
6.2.3	Zirkumskription	185
6.3	Default-Schließen als Behandlung von Inkonsistenz	187
6.3.1	Ein Rahmen für nichtmonotone Systeme	187
6.3.2	Pooles System	189
6.3.3	Zuverlässigkeitssstufen	191
6.4	Nichtmonotonie und Logikprogrammierung	193
6.4.1	Stabile Modelle	194
6.4.2	Wohlfundierte Semantik	195
6.4.3	Antwortmengenprogrammierung	197
6.5	Argumentation	198
6.6	Ausblick	200
	Literaturverzeichnis	201
7	Constraints	205
7.1	Einführung	205
7.2	Finite-Domain-Constraints	207
7.2.1	Constraint-Satisfaction-Probleme	207
7.2.2	Lokale und globale Konsistenz	209
7.2.3	Suchtechniken	212
7.2.4	Globale Constraints	215
7.3	Constraint-basierte Programmierung	220
7.3.1	Constraint-logische Programmierung	221
7.3.2	Constraint-basierte Modellierungssprachen	222

7.3.3	Constraints als Objekte	223
7.3.4	Nebenläufige Constraint-Programmierung	224
7.4	Soft-Constraints.....	225
7.5	Temporale Constraints	227
7.6	Zusammenfassung	229
	Literaturverzeichnis	230
8	Unsicheres und vages Wissen	235
8.1	Begriffe	236
8.1.1	Wissen	236
8.1.2	Impräzision, Unsicherheit und Vagheit	236
8.1.3	Schlussfolgern	238
8.1.4	Wahrscheinlichkeit.....	240
8.1.5	Fuzzy-Menge	246
8.2	Sicherheitsfaktoren	250
8.2.1	Grundlagen des Sicherheitsfaktoransatzes	250
8.2.2	Rechenregeln für Sicherheitsfaktoren	253
8.2.3	Inkonsistenz der Originaldefinition	255
8.2.4	Korrekte probabilistische Interpretation	257
8.3	Probabilistische Schlussfolgerungsnetze	259
8.3.1	Ein einfaches Beispiel	261
8.3.2	Bedingte Unabhängigkeit	266
8.3.3	Darstellung durch Graphen	269
8.3.4	Evidenzpropagation	275
8.3.5	Lernen aus Daten	280
8.4	Fuzzy-Regelsysteme	281
8.4.1	Einführung	281
8.4.2	Fuzzy-Regelsysteme nach Mamdani	283
8.4.3	Defuzzifizierung	285
8.4.4	Fuzzy-Regelung auf der Basis von Gleichheitsrelationen	287
8.4.5	Fuzzy-Regelung und Relationalgleichungen	290
	Literaturverzeichnis	292
9	Fallbasiertes Schließen	297
9.1	Motivation und etwas Historie	297
9.2	Einige Charakteristika von CBR	299
9.3	Grundbegriffe und ein einfaches Modell	299
9.3.1	Fälle als Erfahrungen	299
9.3.2	CBR Methodologie	301
9.3.3	Das Prozessmodell	301
9.3.4	Die Wissenscontainer und ihre Diskussion	303

9.4	Eine Erweiterung	303
9.5	Repräsentationssprachen	305
9.5.1	Attribut-Wert Darstellungen	305
9.5.2	Weitere Darstellungen.....	306
9.6	Ähnlichkeiten	307
9.6.1	Generelles	307
9.6.2	Semantik der Ähnlichkeitsmaße	310
9.6.3	Das lokal-global Prinzip für Ähnlichkeitsmaße	313
9.6.4	Spezielle Ähnlichkeitsmaße	314
9.7	Spezielle Retrievalfragen	317
9.8	Fallbasisprobleme	321
9.9	Adaptionsfragen	321
9.10	Ein paar typische Anwendungen	322
9.10.1	Aufwands-Prognose.....	322
9.10.2	E-commerce	323
9.10.3	Skizzen von Bildern.....	323
9.11	Methodologie zum Aufbau eines CBR-Systems und Integrationsfragen	324
9.11.1	Generelles	324
9.11.2	Integration in übergeordnete Problemlöser.....	325
	Literaturverzeichnis	326
10	Planen	329
10.1	Repräsentation von Planungsproblemen	330
10.1.1	Mengenbasiertes Planen: STRIPS	331
10.1.2	Die Planungsaufgabe	333
10.1.3	Propositionale Repräsentationen	333
10.2	Planen als Suche im Zustandsraum.....	334
10.2.1	Planungsheuristiken	336
10.2.2	Hierarchische Abstraktion	338
10.3	Planen im Planraum	339
10.3.1	Partiell geordnete Pläne	340
10.3.2	Planen mit partiell geordneten Plänen	340
10.3.3	Transformationsplanen	344
10.4	Graphbasiertes Planen	345
10.5	Erweiterungen und Ausblick	350
10.5.1	Erweiterungen des Planungsproblems	350
10.5.2	Was haben wir ausgelassen?	352
10.6	Literatur und Verweise	353
	Literaturverzeichnis	353

11	Neuronale Netze	357
11.1	Motivation	357
11.2	Natürliche neuronale Netze	359
11.2.1	Das Nervensystem besteht aus diskreten Zellen	359
11.2.2	Nervenzellen sind erregbar	360
11.2.3	Synaptische Übertragung	361
11.2.4	Lernen und synaptische Plastizität	363
11.3	Künstliche neuronale Netze	365
11.3.1	Elemente neuronaler Netze	365
11.3.2	Erregungsdynamik	366
11.3.3	Grundtypen von neuronalen Netzen	369
11.3.4	Gewichts- und Strukturdynamik	375
11.3.5	Überwachtes Lernen als Fehlerminimierung	376
11.3.6	Unüberwachtes Lernen	380
11.3.7	Generalisierung und Komplexität	386
11.4	Modellierung biologischer Systeme	390
11.4.1	Neuroanatomie des visuellen Systems	390
11.4.2	Rezeptive Felder	392
11.4.3	Visuelle Informationsverarbeitung	394
11.5	Mustererkennung mit neuronalen Netzen	397
11.6	Schlussbemerkung	401
11.7	Weiterführende Literatur	401
	Literaturverzeichnis	401
12	Maschinelles Lernen und Data Mining	405
12.1	Was ist maschinelles Lernen	406
12.1.1	Intensionale Definitionsversuche	406
12.1.2	Extensionale Definition über Lernaufgaben	407
12.1.3	Motivationen und Anwendungen	408
12.1.4	Wissensentdeckung	409
12.2	Funktionslernen aus Beispielen	410
12.3	Entscheidungsbäume	413
12.3.1	Stutzen des Baumes	418
12.3.2	Boosting and Bagging: Ensemble-Methoden	419
12.3.3	Erweiterungen des Basis-Entscheidungsbaumverfahrens	420
12.4	Instanzbasiertes Lernen	421
12.4.1	Die Ähnlichkeitsfunktion	424
12.4.2	Parameterbestimmung durch Kreuzvalidierung	426
12.4.3	Weitere Verfahrensvarianten	427
12.5	Stützvektormethode	427
12.5.1	SVMs und die optimale Hyperebene	428

12.5.2	Wie berechnet man die optimale Hyperebene	429
12.5.3	Statistische Eigenschaften der optimalen Hyperebene	430
12.5.4	Nicht-lineare SVMs durch Kernfunktionen	432
12.5.5	SVMs mit „weicher“ Trennung	433
12.6	Lernbarkeit in wahrscheinlich annähernd korrektem Lernen (PAC)	433
12.6.1	Stichprobenkomplexität	435
12.7	Lernen aus strukturierten Daten: Logik	438
12.7.1	Repräsentation	439
12.7.2	Algorithmus FOIL	440
12.8	Assoziationsregeln	442
12.8.1	Der Apriori-Algorithmus	444
12.8.2	Erweiterungen	447
12.9	Subgruppenentdeckung	449
12.9.1	Qualitätsfunktionen	450
12.9.2	Effiziente Suche	453
12.9.3	Assoziationsregeln vs. Subgruppen	456
12.9.4	Erweiterungen	456
12.10	Clusteranalyse	457
12.10.1	Das k -Means-Verfahren	458
12.10.2	Hierarchische Clustering-Verfahren	460
12.11	Verstärkungslernen	461
12.11.1	Wann handelt ein Agent optimal?	462
12.11.2	Dynamische Programmierung	463
12.11.3	Q-Learning – Lernen in unbekannter Umgebung	465
12.11.4	Erweiterungen	466
12.12	Weiterführende Themen	466
	Literaturverzeichnis	467
13	Sprachverarbeitung	473
13.1	Sprache und sprachliche Beschreibungsebenen	473
13.2	Sprache und KI	476
13.3	Anwendungen der Sprachtechnologie	480
13.3.1	Werkzeuge für die zwischenmenschliche Kommunikation	482
13.3.2	Werkzeuge für die Textproduktion	483
13.3.3	Werkzeuge für das Informationsmanagement	483
13.3.4	Mensch-Maschine-Kommunikation	485
13.4	Modelle und Verfahren zur Sprachverarbeitung	485
13.4.1	Strukturbeschreibungen	485
13.4.2	Wissensrepräsentation	491
13.4.3	Strukturanalyse	499
13.4.4	Robuste Verfahren	504

13.4.5	Maschinelles Lernen zur Wissensakquisition	513
13.4.6	Generierung	514
13.5	Architekturen für die Sprachverarbeitung.....	516
13.5.1	Modularisierung	516
13.5.2	Inkrementelle Verarbeitung.....	518
13.5.3	Multimodale Kommunikation.....	519
	Literaturverzeichnis	520

14 Multiagentensysteme 527

14.1	Vom Agenten zum Multiagentensystem	528
14.1.1	Begriff und Charakteristika	528
14.1.2	Wichtige Agentenarchitekturen	529
14.1.3	Multiagentensystem	533
14.2	Interaktion, Kommunikation, Organisation	534
14.2.1	Kommunikation und Koordinationsinfrastruktur.....	534
14.2.2	Interaktionsprotokolle	537
14.2.3	Organisation	537
14.2.4	Koordinierte Aktivitäten.....	539
14.3	Von der Kooperation zum Wettbewerb	540
14.3.1	Idee des rationalen Agenten	541
14.3.2	Voting.....	542
14.3.3	Auktionen	544
14.3.4	Verhandlungen	545
14.3.5	Bildung von Koalitionen	546
14.4	Entwicklung und Praxis	546
14.4.1	Agentenorientiertes Software Engineering.....	546
14.4.2	Werkzeuge und Wettbewerbe	548
14.4.3	(Zu) kurzer Blick auf die Anwendungen	549
14.5	Aktuelle Trends	550
	Literaturverzeichnis	551

III Anwendungen

15 Verkörperte Kommunikation mit kognitiven virtuellen Agenten 559

15.1	Einleitung	559
15.2	Grundlagen	560
15.2.1	Begriffe der Kommunikation	560

15.3	Technische Ansätze	563
15.3.1	Dialogmanagement	563
15.3.2	Multimodale Verhaltensverarbeitung	564
15.3.3	Multimodale Verhaltengenerierung	565
15.3.4	Emotionen	566
15.3.5	Kognitive Architektur	566
15.4	Virtueller Kommunikationspartner Max	567
15.4.1	Szenario	567
15.4.2	Kognitive Architektur: Beispiel	568
15.4.3	Interaktionssteuerung	571
15.4.4	Sprach- und Gestenverarbeitung	572
15.4.5	Turn-Taking	573
15.4.6	Multimodale Verhaltengenerierung	575
15.4.7	Physis, Emotionen, Bewegungsgenerierung	577
15.5	Zusammenfassung und Ausblick	578
	Literaturverzeichnis	579
16	Semantic Web	585
16.1	Einleitung	585
16.2	Semantic Web Architektur	586
16.3	Verteilte semantische Daten im Web	590
16.3.1	Verknüpfte Daten	590
16.3.2	Anfragen mit SPARQL	591
16.3.3	Anfragen auf verknüpfte und verteilte Daten	593
16.4	Wissensrepräsentation und -integration	595
16.4.1	Analyse des einführenden Beispiels	595
16.4.2	Verschiedene Arten von Ontologien	596
16.4.3	Verteiltes Netzwerk von Ontologien im Web	597
16.5	Inferenz im Web	599
16.6	Identität und Verknüpfung von Objekten und Begriffen	600
16.7	Herkunft und Vertrauenswürdigkeit von Daten	602
16.8	Semantic Web Anwendungen und Benutzerschnittstellen	603
16.8.1	Vokabulare und Schemas	603
16.8.2	Semantic Web Browser und Semantische Suche	604
16.8.3	Zugriff auf soziale Netzwerke	605
16.8.4	Visualisierung semantisch heterogener und verteilter Daten	605
16.9	Zusammenfassung und Ausblick	606
	Literaturverzeichnis	608

17	Universelle Spielprogramme	613
17.1	Spielregeln beschreiben: Wissensrepräsentation	614
17.1.1	Spielzustände und Züge	614
17.1.2	Spielregeln	615
17.1.3	GDL: Zusammenfassung	618
17.1.4	Kommunikationsprotokoll für GDL	619
17.2	Spielregeln verstehen: Inferenz	619
17.2.1	Unifikation/Grundinstanzierung	622
17.2.2	Ableitungsschritt (ohne Negation)	622
17.2.3	Ableitungen	623
17.2.4	Regeln mit Negation	623
17.2.5	Regeln mit Disjunktion	624
17.3	Spielbaumsuche	624
17.3.1	Minimax-Verfahren	624
17.3.2	Optimierungen	626
17.3.3	Gegenspielermodelle	627
17.4	Stochastische Baumsuche	628
17.4.1	MCT-Suche	629
17.4.2	UCT-Bonus	629
17.4.3	Optimierungen	631
17.4.4	Grenzen	631
17.5	Heuristische Suche	632
17.5.1	Mobilitätsheuristik	633
17.5.2	Zielheuristiken	634
17.5.3	Optimierungen	636
17.6	Wissen	637
17.6.1	Domänenanalyse	637
17.6.2	Regelstrukturanalyse	639
17.7	Spiele mit unvollständiger Information	642
17.7.1	GDL-II	642
17.7.2	Hypothetische Spielstellungen	645
17.8	Weiterführende Literatur	647
	Literaturverzeichnis	647
Index		651

1 Einleitung

Günther Görz, Ute Schmid und Ipke Wachsmuth

Das Forschungsgebiet „Künstliche Intelligenz“ (KI) hat seit seiner Entstehung in der Mitte des letzten Jahrhunderts nichts von seiner Faszination verloren. Mit dem Begriff KI wird oft der Versuch assoziiert, auf Computern und Robotern mentale Prozesse und Verhaltensweisen zu realisieren, die denen von Menschen entsprechen. Unter den zahlreichen Definitionen der Disziplin, die von ihren Fachvertretern angegeben wurden, sei beispielhaft die von [57] formulierte genannt, die die Bestimmung des Gegenstands der KI in folgender Weise präzisiert:

„Künstliche Intelligenz ist die Untersuchung von Berechnungsverfahren, die es ermöglichen, wahrzunehmen, zu schlussfolgern und zu handeln.“

Das heißt, „Künstliche Intelligenz“ ist eine wissenschaftliche Disziplin, die das Ziel verfolgt, menschliche Wahrnehmungs- und Verstandesleistungen zu operationalisieren und durch Artefakte, kunstvoll gestaltete technische – insbesondere informationsverarbeitende – Systeme verfügbar zu machen¹.

Ingenieurwissenschaftliche Leistungen, wie der Bau von Flugzeugen, begeistern viele Menschen. Die Idee der Erschaffung einer „Künstlichen Intelligenz“ trifft dagegen auf gespaltene Gefühle. Dass Menschen versuchen, Menschenähnliches zu erschaffen, kann man vermessen finden – oder aber ein faszinierendes Unterfangen, das bereits früh in Geschichten wie vom Golem oder Frankenstein beschrieben wurde. Entsprechend werden in der Philosophie, aber auch von KI-Forschern selbst, immer wieder ethische und erkenntnistheoretische Fragen aufgeworfen.

Betrachtet man die Realität der KI-Forschung, so ist KI etwas viel Nüchterneres: Ziel ist es, Computerprogramme für Problembereiche zu entwickeln, die bislang nur von Menschen lösbar sind. KI ist als Teil der Informatik eine Ingenieurwissenschaft und als Teil der Kognitionswissenschaft auch Erkenntniswissenschaft. Entsprechend kommen zwei Zielsetzungen zum Tragen:

- Konstruktion „Intelligenter“ Systeme, die bestimmte menschliche Wahrnehmungs- und Verstandesleistungen maschinell verfügbar machen;
- Kognitive Modellierung, d.h. der Simulation kognitiver Prozesse durch Informationsverarbeitungsmodelle.

Eine weitere Orientierung der KI ist formalwissenschaftlich und überlappt stark mit der theoretischen Informatik. In diesem Bereich werden allgemeine Beschränkungen für KI-Algorithmen

¹ Mit Kant wollen wir unter „Verstand“ das Vermögen der Regeln verstehen – im Unterschied zur Vernunft als Vermögen der Prinzipien.

analysiert und insbesondere Fragen der Komplexität, Berechenbarkeit und Lernbarkeit von Problemen bearbeitet.

KI-Forschung hat einen stark interdisziplinären Charakter: Bezüge zur Philosophie ergeben sich über grundsätzliche Fragen über die Natur menschlichen Fühlens, Denkens und Handelns; Bezüge zur Linguistik ergeben sich über menschliche Leistungen bei Sprachverstehen und Sprachproduktion. Psychologie und Neurowissenschaften liefern wesentliche Grundlagen zur Umsetzung menschlicher Repräsentations- und Informationsverarbeitungsmechanismen [3, 18, 53].

In der ingenieurwissenschaftlich orientierten KI liefern Erkenntnisse aus erfahrungswissenschaftlichen Disziplinen, die sich mit mentalen Leistungen des Menschen beschäftigen, häufig Anregungen für die Entwicklung neuer Methoden und Algorithmen. Dabei wird genutzt, was erfolgreich scheint – es besteht kein Anspruch, dass die entwickelten intelligenten Systeme nach ähnlichen Informationsverarbeitungsprinzipien funktionieren wie der Mensch. KI ist hier „*Psychonik*“, analog zur Bionik – der auf biologischen Vorbildern basierenden Entwicklung von Maschinen und Materialien. Als Teil der Kognitionswissenschaft liefert die KI aber auch die Möglichkeit, mit den formalen und algorithmischen Methoden der Informatik kognitive Theorien in lauffähige Modelle umzusetzen. Die kognitive KI hat damit den Anspruch *generative Theorien* menschlicher Informationsverarbeitungsprozesse zu entwickeln – also Theorien, die aufgrund von Berechnungen Verhalten erzeugen [54].

Im Folgenden wird das Forschungsgebiet Künstliche Intelligenz nach verschiedenen Aspekten charakterisiert: Der Begriff „Künstliche Intelligenz“, die historische Entwicklung des Forschungsgebiets, grundsätzliche Herangehensweisen, sowie Teilbereiche und Anwendungsbereiche der KI.

1.1 Zum Begriff Künstliche Intelligenz

Die Bezeichnung „Künstliche Intelligenz“ ist historisch zu verstehen: Zunächst im Englischen als „Artificial Intelligence“ geprägt, ist sie als wörtliche Übersetzung nicht sinngemäß und gibt Anlass zu dem Missverständnis, sie würde eine Definition von „Intelligenz“ liefern oder hätte gar einen operationalisierbaren Intelligenzbegriff insgesamt zu entwickeln. Da die KI eine relativ junge Disziplin ist, zeichnet sich ihre Grundlagendiskussion zudem durch eine metaforphreiche und aufgrund ihres Gegenstands auch stark anthropomorphe Sprache aus.

„Künstliche Intelligenz“ ist ein synthetischer Begriff, der – vermöge seines suggestiven Potentials – viele Missverständnisse und falsche Erwartungen verursacht hat. Sein Ursprung lässt sich auf das Jahr 1956 zurückverfolgen, ein Jahr, das in vielerlei Hinsicht bedeutsam war. Zum Beispiel erschien in diesem Jahr das Buch „Automata Studies“ mit einer Reihe heute berühmter Artikel im Gebiet der Kybernetik [48]. Ebenfalls in diesem Jahr erhielten Bardeen, Shockley und Brattain den Nobelpreis für die Erforschung des Transistors. Noam Chomsky war im Begriff, seinen berühmten Artikel über syntaktische Strukturen zu veröffentlichen, der den Weg für eine theoretische Betrachtung der Sprache eröffnete [14].

Die Bezeichnung „Artificial Intelligence“ wurde von John McCarthy als Thema einer Konferenz geprägt, die im Sommer 1956 am Dartmouth College stattfand und an der eine Reihe renommierter Wissenschaftler teilnahmen (u.a. Marvin Minsky, Nathaniel Rochester, Claude Shannon, Allan Newell, Herbert Simon). Dieses Treffen wird allgemein als Gründungseignis

der Künstlichen Intelligenz gewertet. Im Förderungsantrag an die Rockefeller-Stiftung wurde ausgeführt [33, S. 93]:

„Wir schlagen eine zweimonatige Untersuchung der Künstlichen Intelligenz durch zehn Personen vor, die während des Sommers 1956 am Dartmouth College in Hanover, New Hampshire, durchgeführt werden soll. Die Untersuchung soll auf Grund der Annahme vorgenommen, dass jeder Aspekt des Lernens oder jeder anderen Eigenschaft der Intelligenz im Prinzip so genau beschrieben werden kann, dass er mit einer Maschine simuliert werden kann.“

Es geht, so McCarthy später, um die „Untersuchung der Struktur der Information und der Struktur von Problemlösungsprozessen, unabhängig von Anwendungen und unabhängig von ihrer Realisierung“. Und Newell: „Eine wesentliche Bedingung für intelligentes Handeln hinreichender Allgemeinheit ist die Fähigkeit zur Erzeugung und Manipulation von Symbolstrukturen. Zur Realisierung symbolischer Strukturen sind sowohl die Instanz eines diskreten kombinatorischen Systems (lexikalische und syntaktische Aspekte), als auch die Zugriffsmöglichkeiten zu beliebigen zugeordneten Daten und Prozessen (Aspekte der Bezeichnung, Referenz und Bedeutung) erforderlich.“

Als Instrument der Forschung sollte der Universalrechner dienen, wie Minsky begründete: „... weil Theorien von mentalen Prozessen zu komplex geworden waren und sich zu schnell entwickelt hatten, als dass sie durch gewöhnliche Maschinerie realisiert werden konnten. Einige der Prozesse, die wir untersuchen wollen, nehmen substantielle Änderungen in ihrer eigenen Organisation vor. Die Flexibilität von Computerprogrammen erlaubt Experimente, die nahezu unmöglich in ‚analogen mechanischen Vorrichtungen‘ wären“.

Im September 1956 fand am Massachusetts Institute eine zweite wichtige Konferenz statt, das „Symposium on Information Theory“. So, wie die KI ihren Ursprung auf die Dartmouth Conference zurückführt, kann dieses Symposium als Grundsteinlegung der Kognitionswissenschaft gelten (vgl. 19). Unter den Teilnehmern beider Konferenzen waren Allen Newell und Herbert Simon. Zusammen mit John Shaw hatten sie gerade die Arbeiten an ihrem „Logic Theorist“ abgeschlossen, einem Programm, das mathematische Sätze aus Whiteheads und Russells „Principia Mathematica“ beweisen konnte. Dieses Programm verkörperte schon, was später der Informationsverarbeitungs-Ansatz des Modellierens genannt wurde. Der Grundgedanke dieses Ansatzes ist, dass Theorien des bewussten menschlichen Handelns auf der Basis von Informationsverarbeitungs-Systemen formuliert werden, also Systemen, die aus Speichern, Prozessoren und Steuerstrukturen bestehen und auf Datenstrukturen arbeiten. Seine zentrale Annahme besteht darin, dass im Hinblick auf intelligentes Verhalten der Mensch als ein solches System verstanden werden kann.

Dennoch bleibt uns eine grundsätzlichere Auseinandersetzung mit dem Begriff der *Intelligenz* nicht erspart. „Intelligenz ist die allgemeine Fähigkeit eines Individuums, sein Denken bewusst auf neue Forderungen einzustellen; sie ist allgemeine geistige Anpassungsfähigkeit an neue Aufgaben und Bedingungen des Lebens.“ Diese noch recht unpräzise Bestimmung durch den Psychologen William Stern aus dem Jahre 1912 hat eine Vielzahl von Versuchen nach sich gezogen, eine zusammenhängende Intelligenztheorie zu erstellen, deren keiner dem komplexen Sachverhalt auch nur annähernd gerecht werden konnte [24]. Heute besteht weitgehend Konsens darüber, dass Intelligenz zu verstehen ist als Erkenntnisvermögen, als Urteilsfähigkeit, als das

Erfassen von Möglichkeiten, aber auch als das Vermögen, Zusammenhänge zu begreifen und Einsichten zu gewinnen [15].

Sicherlich wird Intelligenz in besonderer Weise deutlich bei der Fähigkeit, Probleme zu lösen. Die Art, die Effizienz und die Geschwindigkeit, mit der sich der Mensch bei der Problemlösung an die Umwelt anpasst (Adaptation) oder die Umwelt an sich angleicht (Assimilation), ist ein wichtiges Merkmal von Intelligenz. Dabei äußert sich Intelligenz durchaus nicht nur in abstrakten gedanklichen Leistungen wie logischem Denken, Rechnen oder Gedächtnis und insbesondere in der Fähigkeit zur Reflexion, sondern wird ebenso offenkundig beim Umgang mit Wörtern und Sprachregeln oder beim Erkennen von Gegenständen und Situationsverläufen. Neben der konvergenten Fähigkeit, eine Vielzahl von Informationen zu kombinieren, um dadurch Lösungen zu finden, spielt bei der Problemlösung aber auch die *Kreativität* eine wichtige Rolle, insbesondere auch das Vermögen, ausserhalb der aktuellen Informationen liegende Lösungsmöglichkeiten einzubeziehen. Andererseits ist gerade die Fähigkeit zur Begrenzung der Suche nach Lösungen bei hartnäckigen Problemen eine typische Leistung der Intelligenz [6,25].

Und all dies, so müssen wir an dieser Stelle fragen, soll Gegenstand einer künstlichen Intelligenz sein? Kurz gesagt: nein, denn schon wenn wir Intelligenz beurteilen oder gar messen wollen, bedarf es einer Operationalisierung, wodurch wir einen Übergang vom personalen Handeln zum schematischen, nicht-personalen Operieren vollziehen. Das, was operationalisierbar ist, lässt sich grundsätzlich auch mit formalen Systemen darstellen und auf einem Computer berechnen. Vieles aber, was das menschliche Denken kennzeichnet und was wir mit intentionalen Terminen wie Kreativität oder Bewusstsein benennen, entzieht sich weitgehend einer Operationalisierung. Dies wird jedoch angezweifelt von Vertretern der sog. „starken KI-These“, die besagt, dass Bewusstseinsprozesse *nichts anderes* als Berechnungsprozesse sind, die also Intelligenz und Kognition auf bloße Informationsverarbeitung reduziert. Ein solcher Nachweis konnte aber bisher nicht erbracht werden – die Behauptung, es sei *im Prinzip* der Fall, kann den Nachweis nicht ersetzen. Hingegen wird kaum bestritten, dass Intelligenz *auch* Informationsverarbeitung ist – dies entspricht der „schwachen KI-These“.

So, wie wir Intelligenz erst im sozialen Handlungszusammenhang zuschreiben, ja sie sich eigentlich erst darin konstituiert, können wir dann allerdings auch davon sprechen, dass es – in einem eingeschränkten Sinn – Intelligenz in der Mensch-Maschine-Interaktion, in der Wechselwirkung gibt, als „Intelligenz für uns“. Es besteht gar keine Notwendigkeit, einem technischen System, das uns als Medium bei Problemlösungen unterstützt, Intelligenz *per se* zuzuschreiben – die Intelligenz manifestiert sich in der Interaktion.

1.2 Die Entwicklung der KI

In der Folge der oben genannten Tagungen im Jahr 1956 wurden an verschiedenen universitären und ausseruniversitären Einrichtungen einschlägige Forschungsprojekte ins Leben gerufen. Die Prognosen waren zunächst optimistisch, ja geradezu enthusiastisch: Die Künstliche Intelligenz sollte wesentliche Probleme der Psychologie, Linguistik, Mathematik, Ingenieurwissenschaften und des Managements lösen. Fehlschläge blieben nicht aus: So erwies sich das Projekt der automatischen Sprachübersetzung, dessen Lösung man in greifbarer Nähe sah, als enorm unterschätzte Aufgabe. Erst in den 1990er Jahren wurde es – allerdings mit größerer Bescheidenheit – wieder in Angriff genommen.

In der Entwicklung der Künstlichen Intelligenz im letzten Jahrhundert kann man mehrere Phasen unterscheiden: Die Gründungsphase Ende der 1950er Jahre, gekennzeichnet durch erste Ansätze zur symbolischen, nicht-numerischen Informationsverarbeitung, beschäftigte sich mit der Lösung einfacher Puzzles, dem Beweisen von Sätzen der Logik und Geometrie, symbolischen mathematischen Operationen, wie unbestimmter Integration, und Spielen wie Dame und Schach. Das Gewicht lag darauf, die grundsätzliche technische Machbarkeit zu zeigen. Wesentliches Forschungsziel war im Sinne der kognitiven KI – grundsätzliche Prinzipien menschlichen intelligenten Verhaltens maschinell umzusetzen. In dieser ersten Phase – oft durch die Bezeichnung „Power-Based Approach“ charakterisiert, erwartete man sehr viel von allgemeinen Problemlösungsverfahren, deren begrenzte Tragweite allerdings bald erkennbar wurde.

Die zweite Entwicklungsphase der Künstlichen Intelligenz ist gekennzeichnet durch die Einrichtung von Forschungsgruppen an führenden amerikanischen Universitäten, die begannen, zentrale Fragestellungen der KI systematisch zu bearbeiten, z.B. Sprachverarbeitung, automatisches Problemlösen und visuelle Szenenanalyse. In dieser Phase begann die massive Förderung durch die „Advanced Research Projects Agency“ (ARPA) des amerikanischen Verteidigungsministeriums.

In den siebziger Jahren begann eine dritte Phase in der Entwicklung der KI, in der u.a. der Entwurf integrierter Robotersysteme und „expertenhaft problemlösender Systeme“ im Mittelpunkt stand. Letztere machten Gebrauch von umfangreichen codierten Wissensbeständen über bestimmte Gebiete, zunächst in Anwendungen wie symbolische Integration oder Massenspektroskopie. Im Gegensatz zum „Power-Based Approach“ trat die Verwendung formalisierten Problemlösungswissens und spezieller Verarbeitungstechniken in den Vordergrund, was durch die Bezeichnung „Knowledge-Based Approach“ charakterisiert wird. Durch diese Schwerpunktsetzung wurden große Fortschritte bei Techniken der Wissensrepräsentation und in der Systemarchitektur, besonders im Hinblick auf Kontrollmechanismen, erzielt. Im weiteren Verlauf wurde erhebliches Gewicht auf komplexe Anwendungen gelegt: Erkennung kontinuierlich gesprochener Sprache, Analyse und Synthese in der Chemie, medizinische Diagnostik und Therapie, Prospektion in der Mineralogie, Konfiguration und Fehleranalyse technischer Systeme. Zu dieser Zeit waren auch in Europa, vor allem in Großbritannien und Deutschland, KI-Forschungsgruppen an verschiedenen Universitäten entstanden und Förderprogramme installiert. Das Gebiet wurde umfassend mathematisiert und das Konzept der Wissensverarbeitung präzisiert.

Gegen Mitte der 1980er Jahre folgte der „AI Winter“. Die stark auf wissensbasierte Systeme ausgerichteten Entwicklungen stießen an Grenzen und in der Konsequenz wurden auf Symbolverarbeitung basierende Ansätze, die bis dahin die KI-Forschung dominierten, zunehmend kritisch gesehen. Es begann die vierte Phase der KI, in der sub-symbolische Ansätze, insbesondere künstliche neuronale Netze als Alternative zur symbolverarbeitenden KI ins Zentrum rückten.

In den 1990er Jahren wurden neue Themen wie Situiertheit, Verteiltheit und Multiagentensysteme sowie maschinelles Lernen aufgegriffen. Es ist ein deutlicher Trend zu integrierten Ansätzen zu beobachten und eine entsprechende Erweiterung der Begriffe „Wissensverarbeitung“ und „intelligentes System“ auf die neuen Themen. Anwendungen und Anwendungsperspektiven (in vielen Fällen mit dem Boom des Internet verbunden) beeinflussen die aktuellen Forschungsarbeiten in einem sehr hohen Maß. Zudem wurde der Bedarf deutlich, heterogene Wissensquellen in übergreifenden Anwendungen zusammenzuführen und vorhandene Wissensbestände kurz-

fristig auf konkrete Einsatzzwecke zuschneiden zu können. Dies führte zu Arbeiten, die sich auf die Wiederverwendung von Wissensbasen und das sog. Wissensmanagement beziehen.

Aktuell ist die KI-Forschung geprägt durch ein neu erwachtes Interesse an Kognitionsforschung („*Human-level AI*“), durch einen starken Fokus auf probabilistischen Ansätzen und maschinellem Lernen, durch zahlreiche technische Fortschritte im algorithmischen Bereich, beispielsweise in der automatischen Handlungsplanung, und durch die Erschließung neuer Anwendungsgebiete wie „*Ambient Assistant Living*“ und „*Digital Humanities*“.

1.3 Grundsätzliche Herangehensweisen

1.3.1 Symbolische Repräsentation – die Wissensebene

In allen Entwicklungsphasen der KI wurde mit jeweils verschiedenen Ansätzen das Ziel verfolgt, Prinzipien der Informationsverarbeitung zu erforschen und zwar dadurch, dass

1. strikte *Formalisierungen* versucht und
2. exemplarische Realisierungen durch *Implementation* vorgenommen werden.

Dabei galt und gilt auch heute noch zentrale Aufmerksamkeit der Repräsentation und Verarbeitung von Symbolen als wichtige Basis interner Prozesse, von denen man annimmt, dass sie rationales Denken konstituieren. In der Arbeit an ihrem „Logic Theorist“ hatten Simon und Newell erste Eindrücke von den Möglichkeiten des Computers zur Verarbeitung nicht-numerischer Symbole erlangt. Symbole wurden dabei als bezeichnende Objekte verstanden, die den Zugriff auf Bedeutungen – Benennungen und Beschreibungen – ermöglichen. Die symbolische Ebene, repräsentiert in den frühen Arbeiten von Newell, Shaw und Simon [42] wie auch 1956 von Bruner, Goodnow und Austin (vgl. 11), ermöglicht die Betrachtung von Plänen, Prozeduren und Strategien; sie stützt sich ebenfalls auf Vorstellungen regelgeleiteter generativer Systeme [14].

Dabei ist der wichtigste Aspekt, dass sich geistige Fähigkeiten des Menschen auf der symbolischen Ebene unabhängig von der Betrachtung neuronaler Architekturen und Prozesse untersuchen lassen.² Gegenstand der symbolischen KI sind folglich nicht das Gehirn und Prozesse des Abrufs von Gedächtnisinhalten, sondern vielmehr die Bedeutung, die sich einem Prozess vermöge symbolischer Beschreibungen zuordnen lässt. Unbestreitbar hatten die Arbeiten von Newell und Simon in der Präzisierung des Informationsverarbeitungs-Paradigmas einen entscheidenden forschungsorientierenden Einfluss, der zur Ausformung der „symbolischen KI“ führte. Die These, Intelligenzphänomene allein auf der Basis von Symbolverarbeitung untersuchen zu können, ist mittlerweile durch den Einfluss der Kognitions- und der Neurowissenschaften relativiert worden; es zeigte sich, dass zur Erklärung bestimmter Phänomene – insbesondere der Wahrnehmung – der Einbezug der physikalischen Basis, auf der Intelligenz realisiert ist, zu weiteren Erkenntnissen führt. Allerdings ist damit der Ansatz der symbolischen KI nicht obsolet, sondern eher sinnvoll ergänzt und zum Teil integriert worden.

Ein zentrales Paradigma der symbolischen KI wurde mit der Beschreibung des „*intelligenten Agenten*“ („general intelligent agent“) [43] formuliert. Auf einer abstrakten Ebene betrachten

² Diese These ist allerdings nicht unumstritten; vor allem von Forschern auf dem Gebiet der neuronalen Netze wurde versucht, sie zu relativieren.

die Autoren den Gedächtnisbesitz des Individuums und seine Fähigkeit, beim Handeln in der Welt darauf aufzubauen, als funktionale Qualität, die sie mit *Wissen* bezeichnen. Der intelligente Agent verfügt über Sensoren, zur Wahrnehmung von Information aus seiner Umgebung, und über Aktuatoren, mit denen er die äußere Welt beeinflussen kann. Spezifisch für diese Auffassung ist, dass der Agent zu einem internen „Probetun“ fähig ist: Bevor er in der Welt handelt und sie dadurch möglicherweise irreversibel verändert, manipuliert er eine interne Repräsentation der Außenwelt, um den Effekt alternativer, ihm zur Verfügung stehender Methoden abzuwägen. Diese sind ihm in einem internen Methodenspeicher verfügbar, und ihre Exploration wird durch ebenfalls intern verfügbares Weltwissen geleitet.

Die Fragen, mit denen sich vor allem Newell in den frühen achtziger Jahren befasste, waren die folgenden [41]:

- Wie kann Wissen charakterisiert werden?
- Wie steht eine solche Charakterisierung in Beziehung zur Repräsentation?
- Was genau zeichnet ein System aus, wenn es über „Wissen“ verfügt?

Die Hypothese einer *Wissensebene* („Knowledge Level Hypothesis“) wurde von Newell in seinem Hauptvortrag auf der ersten National Conference on Artificial Intelligence in Stanford 1980 (siehe [41]) unterbreitet. In ihr wird eine besondere Systemebene postuliert, über die Ebene der Programmsymbole (und die Ebenen von Registertransfer, logischem und elektronischem Schaltkreis und physikalischem Gerät) hinausgehend, die durch Wissen als das Medium charakterisiert ist. Repräsentationen existieren auf der Symbolebene als Datenstrukturen und Prozesse, die einen Wissensbestand auf der Wissensebene realisieren. Die Verbindung zwischen Wissen und intelligentem Verhalten wird durch das *Rationalitätsprinzip* beschrieben, welches besagt: Wenn ein Agent Wissen darüber hat, dass eine seiner möglichen Aktionen zu einem seiner Ziele beiträgt, dann wird der Agent diese Aktion wählen. In dieser Perspektive spielt Wissen die Rolle der Spezifikation dessen, wozu eine Symbolstruktur in der Lage sein soll. Wichtiger noch wird mit dieser Konzeption Wissen als eine *Kompetenz* betrachtet – als ein Potential, Aktionen zu generieren (zu handeln) – und mithin als eine abstrakte Qualität, die an eine symbolische Repräsentation gebunden sein muss, um einsatzfähig zu sein. Newell und Simon [43][40] postulieren, dass ein dafür geeignetes physikalisches Symbolsystem zur Ausstattung eines jeden intelligenten Agenten gehört.

Eine zentrale Feststellung in Newells oben genanntem Ansatz besagt, dass Logik ein fundamentales Werkzeug für Analysen auf der Wissensebene ist und dass Implementationen von Logikformalismen als Repräsentationsmittel für Wissen dienen können. Der Wissensebenen-Ansatz in der KI ist damit ein Versuch der Mathematisierung bestimmter Aspekte der Intelligenz – unabhängig von Betrachtungen ihrer Realisierung auf Symbolebene; dies betrifft vor allem die Aspekte des rationalen Handelns und des logischen Schlussfolgerns beim Problemlösen. Dementsprechend werden Logikformalismen vielfach in der KI benutzt, um eine explizite Menge von Überzeugungen (für wahr gehaltene Aussagen, engl. „Beliefs“) eines rationalen Agenten zu beschreiben. Eine solche Menge von Überzeugungen, ausgedrückt in einer Repräsentationssprache, wird typischerweise mit dem Terminus *Wissensbasis* bezeichnet.

Diese logikorientierte Auffassung der Wissensebene hat zur Klärung zahlreicher Debatten, die bis Ende der siebziger Jahre um den Begriff der internen Repräsentation geführt wurden, beigetragen [7]. Zum Beispiel wurden unterschiedliche Ansätze der Darstellung von Wis-

sen – wie semantische Netzwerke oder Frames – als notationelle Varianten herausgestellt, so weit es Ausdrucks- und Schlussfähigkeit anbelangt [13]. Die Prominenz, die diese alternativen Notationen nach wie vor in vielen Anwendungsfeldern haben, leitet sich aus ihrer „Objekt-Zentriertheit“ ab, die eine Bequemlichkeit der Beschreibung von Wissensbeständen durch ausgezeichnete Konzepte bietet, und das gesamte Gebiet der objektorientierten Programmierung ist in Verbindung damit groß geworden. Formalismen für die Wissensrepräsentation sind mittlerweile sehr weitgehend und grundsätzlich untersucht worden. Zentrale Gesichtspunkte sind hier u.a. die Ausdrucksfähigkeit und die Komplexität von Repräsentationen, aber auch ihre prädikatenlogische Rekonstruktion bzw. Spezifikation.

Ein standardisiertes Vorgehen bei der Wissens- und Domänenmodellierung hat sich als ausgesprochen schwierig erwiesen. Die Erfahrung hat gezeigt, dass beim Entwurf eines wissensbasierten Systems vor allem hinsichtlich der Wissensbasis eine komplexe kreative Design-Leistung gefordert ist, die mit beinahe jedem neuen System und für jeden weiteren Gegenstandsbereich neu erbracht werden muss. Deshalb stellt sich angesichts des wachsenden Umfangs projektierter wissensbasierter Systeme immer stärker die Frage nach einer Wiederverwendbarkeit schon existierender Wissensbasen bzw. nach einer Aggregation großer Wissensbasen aus bibliotheksmäßig gesammelten oder inkrementell entwickelten Teilen. Vorstöße in dieser Richtung sind verschiedene Ansätze des „Knowledge Sharing“ (vgl. 39) oder der Modularisierung wissensbasierter Systeme (siehe z.B. 34). Die Entwicklung allgemeiner Vorgehensweisen zum Entwurf wissensbasierter Systeme orientiert sich dabei an der von Newell proklamierten Wissensebene mit ihrer Abgrenzung von der Ebene der symbolischen Verarbeitung (vgl. z.B. die KADS-Methode 47). Hiermit verbindet sich der Anspruch, die Wissensinhalte und ihre Funktion für einen Systemzweck ins Zentrum der Modellierungstätigkeit zu stellen und zu abstrahieren von der Form der symbolischen Darstellung des Wissens und den symbolverarbeitenden Prozeduren, die die Funktionalität eines Systems hervorbringen.

Vorangetrieben wurde diese Entwicklung vornehmlich im Kontext des Entwurfs von Expertensystemen. Typischerweise ist der Gegenstandsbereich hier ein eng umrissenes Spezialgebiet, in dem ein hohes Potential an spezifischer Problemlösefähigkeit in einem weitgehend vorab festgelegten Verwendungsrahmen verlangt ist. Bei der Entwicklung von Systemen, die zur semantischen Verarbeitung von natürlicher Sprache fähig sind, geht es dagegen zentral um die Identifikation und Modellierung intersubjektivierbarer Bestände an Welt- oder Hintergrundwissen. Die Modellierung von Alltagswissen, d.h. von allgemeinen Kenntnissen und Fertigkeiten, erhält einen wesentlich höheren Stellenwert und muss umfangreicheren Begriffssystemen Rechnung tragen. Bereits bei Expertensystemen war nun aber eine bittere Erfahrung, dass maschinell verfügbare Expertise genau dort ihre Grenzen hat, wo Alltagswissen und Alltagserfahrung entscheidend zum Tragen kommen. Menschliches Problemlösen zeichnet sich dadurch aus, dass das dabei verwendete Wissen zumeist vage und unvollständig ist. Die Qualität menschlicher Experten zeigt sich gerade darin, dass und wie sie unerwartete Effekte und Ausnahmesituationen aufgrund ihrer Berufserfahrung bewältigen können, dass sie aus Erfahrung *lernen*, ihr Wissen also ständig erweitern, und dass sie aus allgemeinem Wissen nicht nur nach festen Schlussregeln, sondern auch durch Analogie und mit Intuition Folgerungen gewinnen.

Richten sich die systematischen Ansätze im Bereich Expertensysteme vornehmlich auf Strukturen und Typentaxonomien von Problemlösungsaufgaben („Problemlöseontologien“), so stellt die systematische Untersuchung formal repräsentierbarer kognitiver Modelle der menschlichen Weltwahrnehmung eher noch größere Anforderungen. Dieser Ansatz wurde etwa in den Projek-

ten CYC [28] und LILOG (siehe hierzu [26]) angegangen, und er wird derzeit mit dem System WATSON [16] sehr erfolgreich im Bereich des Beantwortens von Quiz-Fragen umgesetzt.

Hier geht es nicht nur darum, generische Problemlöseaufgaben zu betrachten, sondern auch darum, das Fakten- und Relationengefüge diverser Domänen wie auch der Strukturen von Wissensmodellen des Menschen zu erschließen – z.B. durch gestaffelte generische und bereichsbezogene formale „Ontologien“ –, um den Entwurf wissensbasierter Systeme bei der Wissensrepräsentation zu systematisieren. Derartige Ansätze sind in jüngerer Zeit durch den Versuch, heterogene Informationsbestände im Internet semantisch zu erschließen, erheblich beflogt worden. Dies führte zum einen zur Entwicklung von XML-basierten Sprachen zur Repräsentation von Metadaten wie z.B. RDF und RDFS, zum anderen zu signifikanten Fortschritten bei logik-basierten Repräsentationssprachen, insbesondere den sog. Beschreibungslogiken, und effizienten Inferenzmaschinen hierfür. Im Einklang mit der Zielsetzung eines „Semantic Web“ werden seit einigen Jahren beide Strömungen in der „Web Ontology Language“ OWL zusammengeführt und weiterentwickelt [23].

1.3.2 Verteilung und Situiertheit

Unter „Agenten“ werden heute vielfach hardware- oder auch software-basierte Systeme („Software-Agenten“) verstanden, die als mehr oder weniger unabhängige Einheiten innerhalb größerer Systeme agieren. Solche Systeme werden bereits in vielen Disziplinen betrachtet, nicht nur in der KI, sondern als Modellierungsmittel z.B. auch in der Biologie, den Wirtschafts- und den Sozialwissenschaften („Sozionik“). Der Einsatz von Agenten-Techniken interessiert uns in der KI besonders im Hinblick auf Systeme, die in einer dynamischen, sich verändernden Umgebung eingesetzt werden und in größerem Umfang Anteile von Lösungen eigenständig erarbeiten können. Ähnlich wie der allgemeinere Begriff „Objekt“ befindet sich der Begriff des „Agenten“ noch stark in der Diskussion; in der gegenwärtigen Literatur lässt sich deswegen kaum eine allgemein akzeptierte Definition finden. Noch am ehesten ist damit ein Gesamtsystem bezeichnet, das Fähigkeiten der Wahrnehmung, Handlung und Kommunikation miteinander verbindet und, bezogen auf eine zu erfüllende Aufgabe, situationsangemessen ein- und umsetzen kann. Dabei zeichnen sich unterschiedlich stark gefasste Agentenbegriffe ab [58, 59]. In einem schwachen Sinne ist ein Agent ein System mit Eigenschaften der Autonomie (selbstgesteuertes Handeln ohne direkte Außenkontrolle), sozialen Fähigkeiten (Kommunikation und Kooperation mit anderen Agenten), Reaktivität (Verhalten in Erwiderung äußerer Stimuli) und Proaktivität (zielorientiertes Verhalten und Initiative-Übernahme). In der KI werden zumeist stärkere Annahmen gemacht; hier kann ein Agent zusätzlich über „mentalistische“ Eigenschaften verfügen, die mit Begriffen wie Wissen, Überzeugung, Intention, Verpflichtung und zuweilen auch Emotion charakterisiert werden.

Eine vom Modell des „General Intelligent Agent“ abweichende Perspektive wird in Minksys „Society of Mind“ [35] eingenommen. Dieses Paradigma, welches intelligentes Verhalten in der verteilten Tätigkeit vieler kleiner und noch kleinerer Systeme („Agenten“) begründet sieht, beeinflusst eine immer noch wachsende Zahl von Forschern in der KI und führt offensichtlich zu einer andersartigen Vorstellung von Intelligenz. Auf der technischen Seite haben andererseits die Versuche, immer größer und komplexer werdende wissensbasierte Systeme zu entwickeln, Nachteile zentralisierter „Single-Agent“-Architekturen enthüllt und in den 1990er Jahren die Konzeption einer „Verteilten Künstlichen Intelligenz“ (VKI) beflogt [1, 37]. Sog. Multi-Agenten-Systeme stellen den Aspekt der aufgabenbezogenen Kooperation im Wettbe-

werb unabhängiger (autonomer) Teilsysteme (Agenten) heraus, bei welchen kein Agent eine globale Sicht des gesamten Problemlöseprozesses innehat, also keine zentrale Systemsteuerung vorliegt (siehe [58]).

Beim Entwurf von Multi-Agenten-Systemen ist – neben der Realisierung von Fähigkeiten der einzelnen Agenten – die Art der Teilnahme an einem Kooperationsverfahren zur Aufgabenverteilung mit anderen Agenten (ggfs. auch dem beteiligten Benutzer) und der Zugriff auf Kommunikationskanäle zu regeln [52]. Kooperationsprotokolle, die in der VKI betrachtet werden, sind z.B. Master-Slave oder Contract-Net (Vertragsverhandlung). Zur Durchführung von Kooperation erfolgt in der Regel ein Nachrichtenaustausch in einer geeigneten Kommunikationssprache; dazu wird beispielsweise die aus dem oben erwähnten Ansatz des „Knowledge Sharing“ übernommene Knowledge Query and Manipulation Language (KQML) eingesetzt [17]. In Anlehnung an die Sprechakttheorie spezifiziert KQML verschiedene sog. Performative, mit denen Nachrichtentypen (wie Aussage, Frage, Antwort) übermittelt werden können. Für die einzelnen Agenten wird je nach Erfordernis ein offenes Fähigkeitsspektrum betrachtet (vgl. bereits bei [38]), das von sensorgetriebenen, reaktiven bis zu schlussfolgernden Fähigkeiten reicht. Auch wenn in der Regel kein Agent Überblick über das gesamte zu lösende Problem hat, können „höhere“ Agenten Wissen über andere Agenten und ihre Fähigkeiten haben oder erlangen.

Eines der grundsätzlichen Probleme in bisherigen Intelligenzmodellen der KI wie auch in vielen technischen Anwendungen liegt allerdings darin, dass das benötigte Weltwissen kaum jemals vollständig verfügbar bzw. modellierbar ist. Dies beruht auf der kontextuellen Variabilität und der Vielzahl von Situationen, mit denen ein intelligenter Agent konfrontiert sein wird. Deshalb geht die Forschungsrichtung der „situierteren KI“ von der Erkenntnis aus, dass die Handlungsfähigkeit eines intelligenten Agenten entscheidend von seiner Verankerung in der aktuellen Situation abhängt [9]. Situiertheit bezieht sich auf die Fähigkeit eines intelligenten Systems, die aktuelle Situation – durch Wahrnehmung seiner Umgebung oder durch Kommunikation mit kooperierenden Partnern – in weitestgehendem Maße als Informationsquelle auszunutzen, um auch Situationen bewältigen zu können, für die kein komplettes Weltmodell vorliegt (vgl. [30]). Diese Ansätze haben entscheidenden Einfluss auf die Entwicklung einer kognitiven Robotik gehabt, aber auch auf Techniken für Software-Agenten, die zur Erfüllung ihrer Aufgaben durch Situierung in der digitalen Umwelt zusätzliche Informationen beschaffen können.

Rascher Aufschwung nimmt seit einiger Zeit die kognitive Robotik mit der Untersuchung von stationären und mobilen Systemen, die mit Sensoren ihre dynamische Umgebung während der Ausführung von Aufgaben wahrnehmen können. Die Szenarien reichen von forschungsorientierten Ansätzen wie kommunikationsfähigen Montagerobotern und RoboCup-Fußball zu ersten Anwendungen wie teilautonomen Rollstühlen und Robotern für die Kanalisation-Inspektion.

Anwendungsfelder für Software-Agenten sind heute bereits in zahlreichen Bereichen zu finden, u.a. bei verteilten Systemen/Netzwerken und dem Internet (Informations- und Internet-Agenten). Im Bereich der Mensch-Maschine-Interaktion werden sog. Interface-Agenten vielfach diskutiert; sie sind typischerweise als „Personal Assistants“ ausgelegt und können durch Einbettung in die Aufgabenumgebung Wissen über die Tätigkeiten, Gewohnheiten und Präferenzen ihrer Benutzer einsetzen, um an deren Stelle Handlungen auszuführen [27]. Um das Problem der Wissensakquisition durch explizite Programmierung zu umgehen, sind Lernverfahren für Einzel- wie auch Multi-Agenten-Systeme entwickelt worden (z.B. [31] bzw. [29]).

1.3.3 Nicht-Symbolische Ansätze und Maschinelles Lernen

Bereits 1949, als die ersten Digitalrechner ihren Siegeszug angetreten hatten, wurde von D.O. Hebb die Grundlage für ein Verarbeitungsmodell formuliert, das eher in der Tradition des Analogrechnens steht [22]. Er postulierte, dass eine Menge von (formalen) Neuronen dadurch lernen könnte, dass bei gleichzeitiger Aktivierung zweier Neuronen die Stärke ihrer Verbindung vergrößert würde. F. Rosenblatt griff diese Idee auf und arbeitete sie zu einer Alternative zum Konzept der KI in symbolverarbeitenden Maschinen aus:

„Viele der Modelle, die diskutiert wurden, beschäftigen sich mit der Frage, welche logische Struktur ein System besitzen muss, um eine Eigenschaft X darzustellen... Ein alternativer Weg, auf diese Frage zu schauen, ist folgender: Was für ein System kann die Eigenschaft X (im Sinne einer Evolution) hervorbringen? Ich glaube, wir können in einer Zahl von interessanten Fällen zeigen, dass die zweite Frage gelöst werden kann, ohne die Antwort zur ersten zu kennen.“ [45]

1956, im selben Jahr, als Newell und Simons Programm einfache Puzzles lösen und Sätze der Aussagenlogik beweisen konnte, war Rosenblatt bereits in der Lage, ein künstliches neuronales Netzwerk, das Perceptron, lernen zu lassen, gewisse Arten ähnlicher Muster zu klassifizieren und unähnliche auszusondern. Er sah darin eine gewisse Überlegenheit seines subsymbolischen Ansatzes und stellte fest:

„Als Konzept, so scheint es, hat das Perceptron ohne Zweifel Durchführbarkeit und Prinzip nichtmenschlicher Systeme begründet, die menschliche kognitive Funktionen darstellen können... Die Zukunft der Informationsverarbeitungssysteme, die mit statistischen eher als logischen Prinzipien arbeiten, scheint deutlich erkennbar.“ [45]

Zunächst jedoch gewann der symbolische Ansatz in der KI die Oberhand, was nicht zuletzt darin begründet war, dass Rosenblatts Perceptron gewisse einfache logische Aufgaben nicht lösen konnte – eine Beschränkung, die aber ohne weiteres überwunden werden kann. So erfuhr Rosenblatt seit dem Ende der 1980er Jahre eine Rehabilitation, und das Arbeitsgebiet der Neuronalen Netze bzw. des „Konnektionismus“ hat sich rapide zu einem umfangreichen Teilgebiet der KI entwickelt. Der Informatiker Bernd Mahr hat diese Konzeption treffend charakterisiert, so dass wir hier auf seine Darstellung zurückgreifen [32]:

„Für die Erzeugung künstlicher Intelligenz wird ein Maschinenmodell zugrundegelegt, das Arbeitsweise und Struktur des Neuronengeflechts im Gehirn imitiert. Den Neuronenkerne mit ihren Dendriten und deren Verknüpfung über Synapsen entsprechen ‚processor‘-Knoten, die über Verbindungen miteinander gekoppelt sind. ... Die Idee des Lernens durch die Stärkung der Verbindung, die auch schon Rosenblatts Perceptron zugrundelag, findet sich hier in der Fähigkeit wieder, dass die Gewichte der Verbindungen sich ändern können und dass so nicht nur das Pattern der Verbindungen wechselt, sondern auch das Verhalten des gesamten Systems. ... Das ‚Wissen‘, das in einem System steckt, erscheint dann als Pattern der Verbindungsgewichte. ... Künstliche neuronale Netze geben als Computerarchitektur die Manipulation bedeutungstragender Symbole auf ... Sie stellen ‚Wissen‘ ... nicht als aus einzelnen Wissensbestandteilen zusammengesetztes Ganzes dar.“

Als Vorteile künstlicher neuronaler Netze gelten ihre Eigenschaften der verteilten Repräsentation, der Darstellung und Verarbeitung von Unschärfe, der hochgradig parallelen und verteilten Aktion und die daraus resultierende Geschwindigkeit und hohe Fehlertoleranz. Überdies wird mit den Modellierungsansätzen der neuronalen Netze auch ein wichtiges Bindeglied zu den Neurowissenschaften und damit eine Erweiterung des Erkenntnisfortschritts verfügbar.

In den letzten beiden Jahrzehnten wurden in der Forschung im Bereich maschinelles Lernen die neuronalen Netze fast vollständig von statistischen Ansätzen verdrängt. Das Interesse an statistischen Methoden begann durch den großen Erfolg von *Support Vector Machines* im Bereich der automatischen Klassifikation [46]. Zudem werden klassische probabilistische Ansätze wie Bayes'sche Netzwerke weiterentwickelt und erfolgreich sowohl in der kognitiven Modellierung [55] als auch im maschinellen Lernen [5] eingesetzt.

1.3.4 Verkörperung (Embodiment)

Ursprünglich ein Feld der Untersuchung von Intelligenz durch Berechnungsmodelle des Symbolgebrauchs, hat die KI über die letzte Dekade einen Paradigmenwechsel erfahren, der die wissenschaftliche Untersuchung verkörperter künstlicher Agenten in den Gebieten „künstliches Leben“ („Artificial Life“), „humanoide Roboter“ und „virtuelle Menschen“ betrifft. Hiermit verbindet sich eine Abkehr vom strengen Funktionalismus, in dem angenommen wird, dass kognitive Vorgänge prinzipiell unabhängig von ihrem Substrat untersucht werden können. Die Anwendungsforschung hat diesen Wechsel mit Richtungen wie perzeptiven oder anthropomorphen Mensch-Maschine-Schnittstellen und Interface-Agenten aufgegriffen. Diese Vorhaben werden dadurch komplementiert, dass neuartige Interface-Technologien verfügbar werden, wie Kraft- und Positionssensoren, Miniaturkameras, berührungsempfindliche und immersive Displays der virtuellen Realität. Erste Hardware-Plattformen für humanoide Roboter sind kommerziell verfügbar und stellen eine Basis für physische Assistenzsysteme in der häuslichen oder öffentlichen Umgebung bereit. Mit synthetischen Agenten, die komplexe Kommunikationsfähigkeiten des Menschen annähern, steht eine Revolution der heute bekannten Mensch-Technik-Schnittstellen bevor.

Die Paradigmenwechsel in der KI haben zu neuen Forschungsrichtungen geführt, die unter Metaphern wie „verhaltensbasierte KI“, „situierter KI“, „verkörperte KI“ etc. bekannt wurden. In diesen neuen Richtungen wird die Interaktion von Agent und Umgebung anstelle körperloser und rein mentaler Problemlösung als Kern von Kognition und intelligentem Verhalten gesehen (z.B. [2, 9, 44]). Ihr Ziel ist es, künstliche Agenten zu bauen, die mit ihren Umgebungen interagieren und sich daran adaptieren, auch solchen, die ihnen (und ihren menschlichen Entwerfern) vorher großenteils unbekannt sind. Durch ihre Verkörperung sind solche Agenten kontinuierlich an die aktuelle Realweltsituation gekoppelt (d.h. situiert). Forscher in der verkörperten KI und verhaltensbasierten Robotik sind der Überzeugung, dass Verkörperung und Situertheit auch die Hauptmerkmale natürlicher intelligenter Agenten sind und dass beides eine Basis dafür bieten könnte, das Problem zu lösen, wie Symbole in sensorischen Repräsentationen begründet sind (symbol grounding; siehe [21]).

Diese neuen KI-Paradigmen haben auch zu neuen Typen von Modellen geführt. In der Birobotik verwendet man Roboter, um spezifische Verhaltensphänomene zu modellieren, die an Lebewesen beobachtet werden (vgl. 56). Die Modelle zielen hier im allgemeinen auf die neuroethologische (oder in einigen Fällen neurophysiologische) Erklärungsebene. Vor allem aber sind

sie empirisch insofern, als künstliche neuronale Netzwerke in Robotermodellen implementiert werden, die unter analogen Bedingungen wie die Lebewesen getestet und den gleichen Evaluationsmethoden unterzogen werden können wie die Untersuchung von Lebewesen in der Realwelt, z.B. in der Navigation (z.B. 36). Des weiteren verwendet man Roboter zur Modellierung, um zu illustrieren, wie Verhaltensmuster, die auf wichtige Fähigkeiten natürlicher intelligenter Agenten abheben (z.B. „Lernen“, „Imitieren“ oder „Kategorisieren“) implementiert werden können. In solchen Modellen ist es weniger das Ziel, Daten zu reproduzieren, die in einer kontrollierten Umgebung erhoben wurden, sondern stattdessen ein detailliertes Verständnis von kognitiven Fähigkeiten in einem situierten und verkörperten Kontext zu erhalten (z.B. 10). Mit ausdrucksfähigen Gesichtern, Gliedmaßen und Händen von Robotern richten sich die Vorhaben auf die Simulation menschenähnlicher Fähigkeiten wie Aufmerksamkeit und emotionalem Ausdruck (z.B. [8]), Imitationen des Greifens (z.B. [51]) und die Entwicklung von Vorformen sprachlichen Ausdrucks („Proto-Sprache“, siehe [4]).

Ein weiteres Thema in der verkörperten Künstlichen Intelligenz ist die empirische Untersuchung der Evolution sprachlicher Interaktion durch Modellierungsansätze, die sowohl robotische wie simulierte Agenten betreffen (vgl. 49). Hier wird argumentiert (vgl. 50), dass Roboter zumindest mit basalen Kommunikationsfähigkeiten auszustatten sind, um von einfachen Anforderungen wie Hindernisvermeidung und Navigation zu Agenten zu gelangen, von denen sich sagen ließe, sie zeigten „Kognition“. Diese Fähigkeiten müssen bottom-up von den Agenten selbst entwickelt werden, und die kommunizierbaren Bezeichnungen wie auch die Kommunikationsmittel müssen in der sensomotorischen Erfahrung der Agenten ankern. In solcher Weise könnten sich Roboter zu experimentellen Untersuchungen der Entstehung von Sprache und Bedeutung heranziehen lassen.

Ein Vorstoß zur Untersuchung von Kommunikation in vorstrukturierten simulierten Umgebungen wird in der Forschung über „virtuelle Menschen“ („virtual humans“ [20]) und „verkörperte konversationale Agenten“ („embodied conversational agents“ [12]) unternommen. Hier geht es um künstliche Agenten, die wie Menschen aussehen und agieren und in virtueller Realität mit dem Menschen Gespräche führen oder aufgabenbezogen kooperieren. Ausgestattet mit einer synthetischen Stimme, verbalen Konversationsfähigkeiten, visuellen und mitunter auch taktilen Sensoren können sie ihren virtuellen Körper einsetzen, um paralinguistische Qualitäten wie Gestik und emotionale Gesichtsausdrücke mit Sprachausgaben zu synchronisieren. Der Bau solcher Systeme ist ein multidisziplinäres Vorhaben, das traditionelle Künstliche Intelligenz und das volle Spektrum der Forschung über natürliche Sprache mit einer Vielzahl von Anforderungen zusammenführt, die von der Computeranimation über Multimodalität bis hin zu sozialwissenschaftlichen Themen reichen. Der Anspruch dieser Forschung ist es, den Reichtum und die Dynamik menschlichen Kommunikationsverhaltens einzufangen, und die potentiellen Anwendungen sind beträchtlich.

1.4 Teilbereiche und Anwendungsgebiete der KI

Mit den folgenden Kapiteln wird versucht, die etablierten Grundlagen- und Anwendungsbereiche der KI so weit wie möglich abzudecken. Zur Frage einer systematischen Anordnung der verschiedenen Teildisziplinen der KI gibt es durchaus unterschiedliche Auffassungen. Daher wurde lediglich eine grobe Einteilung in drei große Gruppen vorgenommen, wobei der erste

Teil die grundlegenden Theorien und Methoden, der zweite Teil die darauf aufbauenden und weiterführenden Theorien, Methoden und der dritte Teil Anwendungen umfasst.

- Die Betrachtung der *Kognition* als Informationsverarbeitung liefert Grundlagen für eine Fülle von Methoden der KI, die sich schon immer dadurch auszeichnete, nicht nur technische Lösungen zu erarbeiten, sondern diese zur Informationsverarbeitung in Organismen und besonders beim Menschen in Bezug zu setzen.
- *Heuristische Suchverfahren* und *Problemlösemethoden*, die dem Zweck dienen, in hochkomplexen Suchräumen mit möglichst geringem Aufwand kostengünstige Lösungswege zu finden, sind in fast allen Teilgebieten der KI von großer Bedeutung.
- Die *Wissensrepräsentation* befasst sich mit der Darstellung von Objekten, Ereignissen und Verläufen und von Performanz- und Meta-Wissen durch formale, i.a. logikbasierte Systeme.
- Methoden der *Inferenz* sind grundlegend, um Folgerungen aus gegebenen Wissensrepräsentationen automatisch zu extrahieren. Sie realisieren deduktives Schließen in der Maschine. Entsprechende Methoden werden u.a. zur Herstellung und Überprüfung mathematischer Beweise sowie zur Analyse (Verifikation) und Synthese von Programmen angewandt.
- Gerade in alltäglichen Situationen wie auch in der Praxis technischer Anwendungen stoßen die Idealisierungen einer strikten logischen Formalisierung an Grenzen. Hier können neue Methoden zum Umgang mit *unsicherem und vagem Wissen* weiterhelfen.
- Um aus normalerweise unvollständigem Wissen dennoch Fakten ableiten zu können, die für Entscheidungen, Handlungen und Pläne erforderlich sind, werden z.B. Regeln mit Ausnahmen verwendet. *Nichtmonotonen Schließen* behandelt allgemein den Umgang mit Verfahren, die fehlendes Wissen ergänzen.
- Zahlreiche Aufgaben der Künstlichen Intelligenz können durch Systeme von *Constraints* modelliert und gelöst werden, so dass man hier mit Recht von einer Querschnittsmethodik der KI sprechen kann.
- Gerade auf dem Gebiet der *Planung* wurden bahnbrechende Fortschritte erzielt, die zu effizienten Algorithmen für komplexe Planungsaufgaben und die dynamische Planrevision führten.
- *Fallbasiertes Schließen, modellbasierte Systeme und qualitative Modellierung* gehören zum zentralen Methodeninventar wissensbasierter Systeme, deren hauptsächliche Einsatzgebiete in der Lösung komplexer *Planungs-, Konfigurations-, und Diagnoseprobleme* liegen.
- *Künstliche Neuronale Netze* betrachten Verarbeitungsmodelle, die sich durch Lernfähigkeit, Darstellung und Verarbeitung von Unschärfe, hochgradig parallele Aktion und Fehlertoleranz auszeichnen.
- Verfahren des *maschinellen Lernens* sind die Grundlage von Programmsystemen, die aus „Erfahrung“ lernen, also neues Tatsachen- und Regelwissen gewinnen oder Priorisierungen adaptieren können. Sie sind u.a. auch für die Entdeckung zweckbestimmt relevanter Beziehungen in großen Datenmengen („Data Mining“) von großer Bedeutung.
- Von intelligenten Maschinen wird erwartet, dass sie ihre Umgebung wahrnehmen, Ziel orientierte Handlungen planen und die dabei auftretenden Entscheidungen treffen können, wie dies Menschen auch tun würden. *Multiagentensysteme* sollen den Nutzer in seinem Handeln dadurch unterstützen, dass sie die Kontrolle und Steuerung der technischen Einrichtungen übernehmen, ohne dass der Nutzer explizit angeben muss, wie dies im Detail zu geschehen hat, um einen bestimmten Zweck zu erreichen.

- Das nachrichtentechnische Kommunikationsmodell von Shannon und Weaver, wonach ein „Sender“ Informationen verschlüsselt, die ein „Empfänger“ aufnimmt und entschlüsselt, ist bis heute in der Mensch-Maschine-Interaktion stark verbreitet. Mit *verkörperte Kommunikation in intelligenten virtuellen Agenten* wird thematisiert, dass dieses Modell für Verstehen und Modellierung der menschlichen Kommunikation um den Aspekt der körperlichen Einbettung kognitiver Agenten erweitert werden muss, wie zahlreiche empirische Befunde in den Kognitionswissenschaften deutlich gemacht haben.
- In der Anfangsphase der KI nahm die Beschäftigung mit Spielen breiten Raum ein, dienten sie doch als Vehikel zur Erforschung von Heuristiken und der Entwicklung von Strategien. Mit dem neuen Forschungsgebiet der *universellen Spielen* wird eine Abstraktionsebene betreten mit dem Ziel, Computerprogramme zu entwerfen, die selbstständig lernen, Spiele zu spielen und die damit eine weit höhere und allgemeinere Intelligenz als spezielle Spielprogramme zeigen.
- Im World Wide Web, das heute einen breiten Raum in unserem beruflichen und Alltagsleben eingenommen hat, werden unstrukturierte Informationen und informelles Wissen in Form von Hypertext dargestellt. Ziel des *Semantic Web* ist es, mittels Verfahren der Wissensrepräsentation Informationen strukturiert und Wissen formal im Web verteilt bereitzustellen, so dass daraus mithilfe automatisierter Schlussfolgerungen Antworten abgeleitet werden können: Themensuche tritt ergänzend an die Seite der üblichen wortbasierten Suche.

Aus der Grundlagenforschung ging eine Reihe zunächst eher prototypischer Anwendungssysteme hervor, viele ihrer Ergebnisse sind aber heute bereits Teil in der Praxis genutzter Anwendungen geworden. Die Vorteile der *KI-Technologie* sind im wesentlichen von zweierlei Art: Zum einen eröffnet sie neue Anwendungen wie z.B. im maschinellen Sprach- oder Bildverstehen, in der Robotik und mit Expertensystemen. Zum anderen aber ermöglicht sie auch bessere Lösungen für alte Anwendungen; hierzu gehören vor allem die maschinelle Unterstützung von Planen, Entscheiden und Klassifizieren sowie die Verwaltung, Erschließung und Auswertung großer Wissensbestände und schließlich die Simulation und die Steuerung technischer Anlagen.

Ab etwa 1990 schien sich zunächst im Gebiet „Künstliche Intelligenz“ ein Paradigmenwechsel abzuzeichnen – von einer globalen Betrachtung intelligenten Verhaltens hin zu einer Sicht von einfacheren interagierenden Systemen mit unterschiedlichen Repräsentationen oder auch gar keiner Repräsentation –, vertreten durch die Arbeiten zu Multi-Agenten-Systemen, Verteilter KI und Neuronalen Netzwerken. Diese Ansätze bringen eine Erweiterung auf die Untersuchung „situierter“ Systeme ein, welche durch Sensoren und Aktuatoren in ständigem Austausch mit ihrer Umgebung stehen, um etwa auch während einer Problemlösung Situationsdaten aufzunehmen und auszuwerten. Doch wurden auch herkömmliche wissensbasierte Sichtweisen weiterentwickelt, freilich nicht mehr vorwiegend in der Form des autonomen „Expertensystems“, so dass die heutige Situation eher durch das Eindringen wissensbasierter Methoden in vielfältige anspruchsvolle Anwendungssysteme gekennzeichnet ist. Manche Forderungen nach einer feinkörnigen und umfassenden formalen Modellierung komplexer Anwendungsbereiche ließen sich nicht so einfach einlösen, wie man zunächst geglaubt hatte; vor allem aus Komplexitätsgründen müssen immer wieder Vereinfachungen vorgenommen und Kompromisse zwischen formaler Ausdruckskraft und praktischer Beherrschbarkeit geschlossen werden. So bietet sich heute insgesamt ein Bild der KI, das durch eine Koexistenz unterschiedlicher Herangehensweisen, methodischer Ansätze und Lösungswege gekennzeichnet ist und damit aber auch eine beachtenswerte Bereicherung erfahren hat.

Bislang hat kein einzelner Ansatz eine Perspektive geboten, mit der sich alle Aspekte intelligenten Verhaltens reproduzieren oder erklären ließen, wie es auf der Dartmouth-Konferenz als Programm formuliert wurde. Vor gut zehn Jahren hat der „Scientific American“³ Minsky mit dem treffenden Satz zitiert: „The mind is a tractor-trailor, rolling on many wheels, but AI workers keep designing unicycles.“ Erscheinen nach wie vor noch viele Fragen als grundsätzlich ungelöst, so gibt es doch Evidenz dafür, dass mittlerweile mehr als ein „Rad“ untersucht wird und dass gerade die Integration verschiedener Ansätze weitergehende Perspektiven für die Grundlagenforschung und Anwendungsentwicklung eröffnet.

Danksagung. Die Autoren danken Clemens Beckstein für eine Reihe hilfreicher Hinweise.

Literaturverzeichnis

- [1] Adler, M., Durfee, E., Huhns, M., Punch, W., und Simoudis, E. (1992). AAAI Workshop on Cooperation Among Heterogeneous Intelligent Agents. *AI Magazine*, 13(2):39–42.
- [2] Arkin, R. (1998). *Behavior-Based Robotics*. MIT Press, Cambridge, MA.
- [3] Bermúdez, J. L. (2010). *Cognitive Science: An Introduction to the Science of the Mind*. Cambridge University Press.
- [4] Billard, A. (2002). Imitation: a means to enhance learning of a synthetic proto-language in an autonomous robot. In Dautenhahn, K. und Nehaniv, C., editors, *Imitation in Animals and Artifacts*, pages 281–311. MIT Press, Cambridge, MA.
- [5] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- [6] Boden, M. (2004). *The creative mind: myths and mechanisms*. Routledge.
- [7] Brachman, R. (1979). On the epistemological status of semantic networks. In Findler, N., editor, *Associative Networks: Representation and Use of Knowledge by Computers*, pages 3–50. Academic Press, New York.
- [8] Breazeal, C. und Scassellati, B. (1999). A context-dependent attention system for a social robot. In *Proc. Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 1146–1151, Stockholm, Sweden.
- [9] Brooks, R. (1991). Intelligence without representation. *Artificial Intelligence*, 47:139–160.
- [10] Brooks, R., Breazeal, C., Marjanovic, M., Scassellati, B., und Williamson, M. (1998). The cog project: Building a humanoid robot. In Nehaniv, C., editor, *Computation for Metaphors, Analogy, and Agents*, number 1562 in LNAI, pages 52–87. Springer, New York.
- [11] Bruner, J. S., Goodnow, J. J., und Austin, G. A. (1956). *A Study of Thinking*. Wiley, New York.
- [12] Cassell, J., Sullivan, J., Prevost, S., und Churchill, E., editors (2000). *Embodied Conversational Agents*. MIT Press, Cambridge, MA.
- [13] Charniak, E. und McDermott, D. (1985). *Introduction to Artificial Intelligence*. Addison-Wesley, Reading, MA.
- [14] Chomsky, N. (1957). *Syntactic Structures*. Mouton, The Hague.

³ Scientific American, Nov. 1993, Profiles: „Marvin L. Minsky – The Mastermind of Artificial Intelligence“, S. 14–15.

- [15] Cianciolo, A. T. und Sternberg, R. J. (2004). *Intelligence: A brief history*. Blackwell Publishing, Malden, MA.
- [16] Ferrucci, D., Brown, E., Chu-Carroll, J., Gondek, J. F. D., Kalyanpur, A. A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J., Schlaefer, N., und Welty, C. (2010). Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31(3):59–79.
- [17] Finin, T., Labrou, Y., und Mayfield, J. (1997). KQML as an agent communication language. In Bradshaw, J., editor, *Software Agents*. MIT Press, Cambridge, MA.
- [18] Frankish, K. und Ramsey, W., editors (2012). *The Cambridge Handbook of Cognitive Science*. Cambridge University Press.
- [19] Gardner, H. (1985). *The Mind's New Science – A History of the Cognitive Revolution*. Basic Books, New York.
- [20] Gratch, J., Rickel, J., André, E., Badler, N., Cassell, J., und Petajan, E. (2002). Creating interactive virtual humans: Some assembly required. *IEEE Intelligent Systems*, 2:2–11.
- [21] Harnad, S. (1990). The symbol grounding problem. *Physica D*, 42:335–346.
- [22] Hebb, D. (1949). *The Organization of Behavior*. Wiley, New York.
- [23] Hitzler, P., Krötzsch, M., und Rudolph, S. (2009). *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC.
- [24] Irrgang, B. und Klawitter, J. (1990). Künstliche Intelligenz – Technologischer Traum oder gesellschaftliches Trauma. In Irrgang, B. und Klawitter, J., editors, *Künstliche Intelligenz*, pages 7–54. Hirzel, Edition Universitas, Stuttgart.
- [25] Kaplan, C. und Simon, H. (1990). In search of insight. *Cognitive Psychology*, 22:374–419.
- [26] Klose, G., Lange, E., und Pirlein, T. (1992). *Ontologie und Axiomatik von LILOG*. Springer (IFB 307), Berlin.
- [27] Laurel, B. (1990). Interface agents: Metaphors with character. In Laurel, B., editor, *The Art of Human-Computer Interface Design*. Addison-Wesley, Reading, MA.
- [28] Lenat, D. und Guha, R. (1990). *Building Large Knowledge-Based Systems – Representation and Inference in the Cyc Project*. Addison-Wesley, Reading, MA.
- [29] Lenzmann, B. und Wachsmuth, I. (1997). Contract-net-based learning in a user-adaptive interface agency. In Weiss, G., editor, *Distributed Artificial Intelligence Meets Machine Learning – Learning in Multi-Agent Environments*, number 1221 in LNAI, pages 202–222. Springer, Berlin.
- [30] Lobin, H. (1993). Situiertheit. *KI (Rubrik KI-Lexikon)*, 1:63.
- [31] Maes, P. und Kozierok, R. (1993). Learning interface agents. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI-93)*, pages 459–465, Menlo Park. AAAI Press/The MIT Press.
- [32] Mahr, B. (1989). Chaos-Connection. Einwände eines Informatikers. *Kursbuch*, 98:83–99.
- [33] McCorduck, P. (1979). *Machines Who Think*. Freeman, San Francisco.
- [34] Meyer-Fujara, J., Heller, B., Schlegelmilch, S., und Wachsmuth, I. (1994). Knowledge-level modularization of a complex knowledge base. In Nebel, B. und Dreschler-Fischer, L., editors, *KI-94: Advances in Artificial Intelligence*, LNAI, pages 214–225, Berlin. Springer.
- [35] Minsky, M. (1986). *The Society of Mind*. : Simon & Schuster, New York.
- [36] Möller, R., Lambrinos, D., Roggendorf, T., Pfeifer, R., und Wehner, R. (2001). Insect strategies of visual homing in mobile robots. In Consi, T. und Webb, B., editors, *Biorobotics – Methods and Applications*, pages 37–66. AAAI Press, Menlo Park.
- [37] Müller, J., editor (1993). *Verteilte Künstliche Intelligenz – Methoden und Anwendungen*. BI Wissenschaftsverlag, Mannheim.

- [38] Müller, J. und Siekmann, J. (1991). Structured social agents. In Brauer, W. und Hernández, D., editors, *Verteilte Künstliche Intelligenz und kooperatives Arbeiten. 4. Internationaler GI-Kongress Wissensbasierte Systeme*, Heidelberg. Springer.
- [39] Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., und Swartout, W. (1991). Enabling technology for knowledge sharing. *AI Magazine*, 12(3):37–56.
- [40] Newell, A. (1980). Physical symbol systems. *Cognitive Science*, 4:135–183.
- [41] Newell, A. (1981). The knowledge level. *AI Magazine*, 2(2):1–20.
- [42] Newell, A., Shaw, J., und Simon, H. (1958). Chess playing programs and the problem of complexity. *IBM Journal of Research and Development*, 2(4).
- [43] Newell, A. und Simon, H. (1972). *Human Problem Solving*. Prentice-Hall, Englewood Cliffs, N.J.
- [44] Pfeifer, R. und Scheier, C. (1999). *Understanding Intelligence*. MIT Press, Cambridge, MA.
- [45] Rosenblatt, F. (1962). Strategic approaches to the study of brain models. In v.Foerster, H., editor, *Principles of Self-Organization*, page 387. Pergamon Press, Elmsford, N.Y.
- [46] Schölkopf, B. und Smola, A. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA.
- [47] Schreiber, A. (1999). *Knowledge Engineering and Management: The CommonKADS Methodology*. MIT Press, Cambridge, MA.
- [48] Shannon, C. und McCarthy, J. (1956). *Automata Studies*. Princeton University Press, Princeton, NJ.
- [49] Steels, L. (2000). The puzzle of language evolution. *Kognitionswissenschaft*, 8:143–150.
- [50] Steels, L. und Vogt, P. (1997). Grounding adaptive language games in robotic agents. In Husbands, C. und Harvey, I., editors, *Proceedings of the Fourth European Conference on Artificial Life (ECAL-97)*, pages 474–482, Cambridge, MA. MIT Press.
- [51] Steil, J., Röthling, F., Haschke, R., und Ritter, H. (2004). Situated robot learning for multimodal instruction and imitation of grasping. *Robotics and Autonomous Systems (Special Issue on Imitation Learning)*, 47:129–141.
- [52] Steiner, D., Haugeneder, H., Kolb, M., Bomarius, F., und Burt, A. (1992). Mensch-maschine-kooperation. *KI*, 1:59–63.
- [53] Strube, G., editor (1996). *Wörterbuch der Kognitionswissenschaft*. Klett-Cotta, Stuttgart, Germany.
- [54] Strube, G. (2000). Generative theories in cognitive psychology. *Theory & Psychology*, 10:117–125.
- [55] Tenenbaum, J., Griffiths, T., und Kemp, C. (2006). Theory-based Bayesian models of inductive learning and reasoning. *Trends in Cognitive Sciences*, 10(7):309–318.
- [56] Webb, B. (2001). Can robots make good models of biological behaviour? *The Behavioral and Brain Sciences*, 24(6):1033–1050.
- [57] Winston, P. (1992). *Artificial Intelligence (3rd edition)*. Addison-Wesley, Reading, MA.
- [58] Wooldridge, M. (2002). *Introduction to Multiagent Systems*. Wiley, New York.
- [59] Wooldridge, M. und Jennings, N. (1995). Agent theories, architectures, and languages: A survey. In Wooldridge, M. und Jennings, N., editors, *Intelligent Agents: Theories, Architectures, and Languages*, number 890 in LNAI, pages 1–21. Springer, Berlin.

Teil I

Grundlagen

2 Kognition

Gerhard Strube, Evelyn Ferstl, Lars Konieczny, und Marco Ragni

Auch heute noch gilt für die Künstliche Intelligenz (KI): Ihr Maßstab ist die Intelligenz des Menschen als artspezifisches Merkmal, ihr Ziel demnach: technische Systeme zu entwickeln, die in ihrer Leistungsfähigkeit und Flexibilität dem nahekommen, was (mehr oder weniger) alle Menschen können, oder es sogar übertreffen. Dieses Kapitel handelt von den Leistungen natürlicher kognitiver Systeme, also der Kognition von Organismen, speziell der menschlichen Kognition.

Allgemein sind kognitive Systeme dadurch charakterisiert,

- dass sie in eine Umgebung (zu der in der Regel auch andere kognitive Systeme gehören) handelnd eingebunden sind und mit ihr auch informationell im Austausch stehen,
- dass sie ihr Handeln flexibel und umgebungsadaptiv steuern, und zwar dadurch, dass sie relevante Aspekte der Umwelt intern (mental) repräsentieren, und
- dass ihre Informationsverarbeitung durch Lernfähigkeit und Antizipation ausgezeichnet ist.

Typischerweise sind kognitive Systeme entweder Organismen (biologische kognitive Systeme) oder technische Systeme (Roboter, Agenten). Es können aber auch Gruppen solcher Systeme (z.B. gemischte Mensch-Maschine-Verbünde) als ein (komplexes) kognitives System analysiert werden.

Kognitive Prozesse werden als solche der Informationsverarbeitung verstanden, also als Berechnungsvorgänge. Dies ist die Grundlage, von der aus biologische und technische Systeme gleichermaßen hinsichtlich ihrer kognitiven Funktionen betrachtet werden können.

Ein kognitives System handelt in einer Umwelt, auf die es sich bezieht. Solche Bezugnahme, in der Philosophie „Intentionalität“ genannt, ist unverzichtbar für die menschliche Kognition; wir denken und sprechen *über* etwas, glauben oder wollen *etwas*, usw. [180]. Das Kind entwickelt Kategorien und Begriffe in handelnder Interaktion mit der Umwelt; Wörter erfahren ihre Bedeutung unter Bezugnahme auf die Welt „draußen“ – dies ist die heute weithin akzeptierte Lösung des symbol grounding problem [84,192]. Dies gilt sowohl für Symbolverarbeitung – die Symbole stehen für etwas in der Welt –, als auch für (natürliche wie künstliche) neuronale Netze, denn auch „subsymbolische“ Einheiten sind nicht bedeutungslos. Durch den Umweltbezug kognitiver Systeme kommt es zu Rückkopplungen mit der Umwelt. Die vom System aufgenommene Information dient zur Handlungssteuerung des Systems, und die Wahrnehmung dadurch bewirkter Veränderungen wiederum zur Kontrolle der Systemtätigkeit. Am deutlichsten wird dies am Dialog zwischen zwei Systemen, deren jedes „Umwelt“ für das andere ist, wie bei der klassischen Mensch-Computer-Interaktion.

2.1 Kognitionswissenschaft

2.1.1 Charakterisierung und historische Entwicklung

Die Kognitionswissenschaft entstand durch ein Fünfjahresprogramm der *Sloan Foundation* ab 1976; die Grundlage dafür war ein Gutachten, das konvergente Tendenzen in mehreren Disziplinen (darunter Informatik, Psychologie, Linguistik, Neurowissenschaften, Philosophie und Kulturanthropologie) aufzeigte, wonach kognitive Prozesse als Informationsverarbeitung, mithin also als Berechnungsprozesse zu verstehen seien. Innerhalb kurzer Zeit entstand eine Zeitschrift (*Cognitive Science*, 1977) und eine wissenschaftliche Gesellschaft (*Cognitive Science Society*, 1979); in Deutschland wurden später ebenfalls eine Zeitschrift (*Kognitionswissenschaft*, 1990–2002) und die *Gesellschaft für Kognitionswissenschaft* (1994) gegründet.

Die Entwicklung der Kognitionswissenschaft ist eng mit dem jeweiligen Stand der Informatik und vor allem der KI verknüpft. Die „klassische“ Kognitionswissenschaft setzte sich zum Ziel, Computermodelle solcher geistigen Leistungen zu erstellen, für die Menschen gemeinhin Intelligenz zugesprochen wird, z.B. algebraische Ableitungen zu machen, logische Schlüsse zu ziehen, oder Schach zu spielen. Implementiert wurden sie vorwiegend in LISP oder Produktionsregelsprachen; Logik und Deduktion standen im Vordergrund, sowie Problemlösen durch Suche. Die Verbindung zur Kognitionspychologie markiert das Buch *Human Problem Solving* von Newell und Simon [145]. Die Bindung an Logik und den damit einhergehenden (naiven) Realismus wird auch an den einflussreichen Schriften von Jerry Fodor [54] deutlich. Wesentlich war die Bindung an Symbolverarbeitung, die von Newell & Simon [146] programmatisch als notwendige und hinreichende Grundlage intelligenten Verhaltens aufgefasst wurde.

Eine erste deutliche Veränderung erfolgte um 1970, als in der KI wie der (Entwicklungs-)Psychologie gleichermaßen die Bedeutung bereichsspezifischen Wissens erkannt wurde. Daraus entstand die Zielsetzung der in den achtziger Jahren massiv beworbenen „Expertensysteme“ (d.h. Systeme, die Experten ersetzen können), woraus inzwischen vernünftigerweise Assistenz- und Entscheidungs-Unterstützungs-Systeme geworden sind.

Mitte der achtziger Jahre trat ein schon einmal totgesagtes, alternatives Berechnungsmodell auf den Plan, der sog. Konnektionismus, also die künstlichen neuronalen Netze (KNN, vgl. Kapitel 2). In KI wie in der Kognitionswissenschaft wurden KNN aggressiv als bessere Alternative zur Symbolverarbeitung vermarktet: „The computer metaphor is detrimental to cognitive science, what we need is brain-style modeling“ (so David Rumelhart 1988 auf einem Vortrag in Sydney). Dass KNN nicht natürlichen Nervennetzen entsprechen, sondern vielmehr ein Berechnungsparadigma darstellen, hat Smolensky [190] klargestellt. Inzwischen sind KNN wie Symbolverarbeitung akzeptierte und – wie in hybriden Systemen, z.B. ACT-R [7] – sogar kombinierbare Berechnungsmodelle.

Etwa um die gleiche Zeit wurde die Erkenntnis, dass (natürliche) Kognition in aller Regel in – und mit Nutzung – der situativen Umgebung, ja sogar kollaborativ sich entfaltet, unter dem Stichwort *situated cognition* propagiert [199]. Erweitert wurde dies in neuerer Zeit um *embodied cognition*: Berücksichtigung dessen, dass unsere Kognition nicht unwe sentlich auf unsere leibliche Existenz bezogen ist. Allerdings sind die diversen Ansätze unter dem Namen „embodied cognition“ nicht einheitlich ([221] s. Abschnitt 2.2.2). Aber die Entwicklung insgesamt entspricht verblüffend dem Fortschritt der KI von reiner Symbolverarbeitung, wie sie John Searle [182] publikumswirksam kritisiert hatte, hin zu Anforderungen

von Echtzeitfähigkeit, Nebenläufigkeit, und vor allem zur „Leiblichkeit“ der Robotik – also der Erkenntnis, dass unsere körperliche Ausstattung, vor allem die Sensorik und Motorik, auch die „höheren“ kognitiven Prozesse entscheidend mitbestimmen. Parallel dazu hat die KI sich mehr und mehr von abgeschlossenen Mini-Welten abgewandt und auch solche Aufgaben in den Blick genommen, die typischerweise von (fast) allen Menschen erfolgreich bewältigt werden.

Von Anfang an war die Kognitionswissenschaft eine „Interdisziplin“ [206], welche die methodologischen Traditionen ihrer Mutterdisziplinen integrierte, nämlich formale Analysemethoden (aus Philosophie und Logik, theoretischer Informatik und formaler Linguistik), empirische Analyse (Methoden der Psychologie und Neurowissenschaften, insbesondere das naturwissenschaftliche Experiment) und die vor allem der KI entlehnten synthetischen Methoden zur Konstruktion von Computermodellen. Die sog. *kognitive Modellierung* (s. Abschnitt 2.1.2) geht aus von vorliegenden Erkenntnissen, erstellt Computermodelle mentaler Repräsentationen und der auf diesen operierenden kognitiven Prozesse und überprüft die Qualität der Modelle wiederum durch empirische Daten. Dabei gelten für derartige Computermodelle weit schärfere Kriterien als für Computersimulationen generell. Kognitive Modelle sollen z.B. ähnliche Fehlermuster aufweisen wie Menschen, ihr Lösungsprozess sollte vergleichbare Stadien durchlaufen, sie sollten mehr Ressourcen verbrauchen für Aufgaben, die auch für Menschen schwieriger sind, usw. [184]. Indem solche Modelle die kognitiven Leistungen, die sie erklären sollen, überprüfbar Schritt für Schritt hervorbringen, stellen sie als *generative Theorien* [198] wissenschaftliche Theorien höchster Qualität dar.

Eine solch anspruchsvolle Integration vielfältiger und je für sich komplizierter Methoden und Techniken zu leisten kann kaum eine Anforderung an einzelne Individuen sein; vielmehr ist hier interdisziplinäre Kooperation erforderlich. Dennoch hat sich in der Kognitionswissenschaft spätestens seit der Gründung der *Cognitive Neuroscience Society* 1994 gezeigt, dass die Entwicklung eher zu einem multidisziplinären (statt strikt interdisziplinären) Forschungsfeld geführt hat. Man mag dies bedauern, aber die rasante Entwicklung der Kognitionswissenschaft hat eine solche Verbreiterung der Forschungsperspektive und entsprechende Detailfülle hervorgebracht, dass niemand mehr das Fach zur Gänze und zugleich in allen wesentlichen Einzelheiten zu überblicken vermag. Insbesondere hat das verständliche Interesse an „unserer“ menschlichen Implementierung von Kognition zu einer florierenden Hirnforschung geführt, die technische Systeme weitgehend ignoriert. Über das Interesse an Modellierung bleibt die Kognitionswissenschaft weiterhin eng mit der KI verbunden, und umgekehrt haben die neueren Entwicklungen in der Robotik zu einem verstärkten Interesse der KI an der Kognitionswissenschaft geführt. Zudem steigt erfreulicherweise die Zahl der Forschenden, die sich nicht nur in einer kognitionswissenschaftlichen Disziplin zu Hause fühlen.

Insgesamt lässt sich festhalten, dass die kognitionswissenschaftliche Grundannahme, wonach Kognition als Informationsverarbeitung zu begreifen sei, zu einer Fülle von Einsichten in die menschliche Kognition und ebenso zu zahlreichen Verbesserungen in speziellen Arbeitswelten, vor allem in der Mensch-Computer-Interaktion, geführt hat. Über ihr Interesse an einer präzisen Modellierung kognitiver Prozesse ist die Kognitionswissenschaft eng mit der Informatik verbunden. Für KI und Robotik ist die menschliche Kognition Maßstab, und oft Vorbild. Dies zeigt sich am deutlichsten in der Sprachverarbeitung bzw. Computerlinguistik, denn anders als z.B. Fortbewegung, wofür Natur wie Technik eine Fülle verschiedenster Lösungen gefunden ha-

ben, ist die natürliche Sprache in ihrer Vielfalt von Einzelsprachen dem Menschen artspezifisch eigen.

2.1.2 Methoden kognitionswissenschaftlicher Forschung

Die Methoden innerhalb der Kognitionswissenschaft entstammen den Disziplinen, aus denen sie entstanden ist. Dies erklärt die Breite und Heterogenität der angewandten Verfahren. Wenigstens drei Gruppen von Methoden lassen sich unterscheiden:

- Theoretische Analyse kognitiver Funktionsbereiche und Aufgabenstellungen mit dem Ziel der Formalisierung, sowie der Bestimmung von Regularitäten und grundlegenden Beschränkungen (Methoden der Geistes- und Formalwissenschaften, z.B. Philosophie, Logik und Linguistik),
- Empirische Untersuchung der Organisation der Informationsverarbeitung bei Mensch und Tier, sowie der biologischen Grundlagen natürlicher kognitiver Systeme (Methoden der Naturwissenschaften, z.B. Psychologie und Neurowissenschaften, also die strenge Methode des naturwissenschaftlichen Experiments, nicht etwa irgendwelches „Herumexperimentieren“),
- Modellierung kognitiver Prozesse durch Konstruktion virtueller kognitiver Maschinen, also Computerprogrammen, mit den Mitteln der Informatik (Methoden der Ingenieurwissenschaften, z.B. im Bereich Künstliche Intelligenz).

Die Ergebnisse, die über einen Inhaltsbereich (etwa „Sprache“) mit diesen Methoden gewonnen werden können, müssen für eine kognitionswissenschaftliche Betrachtung aufeinander bezogen werden. Dies ist ein anspruchsvolles Ziel, das mehr die Ausrichtung der Arbeit anzeigt, als dass es für gewöhnlich erreicht würde. Zumindest in den empirisch ausgerichteten Fachgebieten ist jedoch insofern eine gewisse Einigkeit festzustellen, als die Computersimulation eine vorrangige Rolle bei der Theoriebildung spielt.

In der Tat kann die kognitive Modellierung [198] insofern als die kognitionswissenschaftliche Methodologie angesehen werden, als sie nicht einfach Computersimulation ist, sondern auf theoretischen Analysen und bekannten empirischen Resultaten aufbauend zur Modellierung kognitiver Prozesse forschreitet und das Modell durch weitere empirische Untersuchungen überprüft.

2.1.2.1. Theoretische Analyse kognitiver Funktionsbereiche

Hier kann die Kognitionswissenschaft auf die reiche Tradition der Philosophie, insbesondere der Erkenntnistheorie, zurückgreifen. Hinzu kommen Denkrichtungen, die vom Hauptstrom abendländischen Denkens abweichen. Hinzu kommen ferner die spezielleren Disziplinen, die sich im Laufe der letzten Jahrhunderte aus der Philosophie ausgegliedert haben: (theoretische) Psychologie, (philosophische und mathematische) Logik sowie die Formalwissenschaften generell, von denen als eine kognitionswissenschaftlich besonders bedeutsame die theoretische Linguistik hier erwähnt sei. Diese Wissenschaften haben wesentliche Beiträge zur inhaltlichen und formalen Analyse von Produkten der Kognition geleistet: von sprachlichen Äußerungen und sozialer Kommunikation, von Kunstwerken und anderen kulturellen Leistungen, von Erkenntnisprozessen selbst und von Argumentationsstrukturen. Infolgedessen dürfen Untersuchungen menschlicher Sprachverarbeitung die von der Sprachwissenschaft beschriebenen Phänomene

und Theorien, insbesondere linguistische Methoden zur Überprüfung sprachbezogener Modellvorstellungen nicht ignorieren, ohne kognitionswissenschaftlich zu kurz zu greifen. Entsprechend muss die Konstruktion einer Konzepthierarchie für ein wissensbasiertes System, wenn sie kognitionswissenschaftliche fundiert sein soll, auf philosophische Untersuchungen zur Ontologie und auf psychologische Untersuchungen (s.u.) gestützt sein. Es fehlt hier der Platz, entsprechende Methoden auch nur in Auswahl vorzustellen. Auch existiert kein Kanon dessen, was ein Kognitionswissenschaftler unbedingt an Methoden beherrschen sollte (wenn auch wohl Konsens besteht, dass eine Mindestbeherrschung formaler Logik und Grundkenntnisse in der formalen Linguistik unverzichtbar sind). Ein möglichst breiter Hintergrund und die Integration solcher inhaltlichen und methodischen Kenntnisse mit informatischen und experimentell-naturwissenschaftlichen Methoden (s. die folgenden Abschnitte) ist für die Kognitionswissenschaft charakteristisch.

2.1.2.2. Experimentelle Methoden

Grundlage von Modellen der kognitiven Verarbeitung ist oft eine Aufgabenanalyse. Dabei wird eine umfassende kognitive Leistung (z.B. „ein Gedicht auswendig lernen“) in Teilprozesse zerlegt, und es werden Annahmen gemacht über den zeitlichen Verlauf, die gegenseitige Beeinflussung der Prozesse, usw. So wird z.B. das auswendig Lernen eines Gedichtes die Teilprozesse: Lesen, Einprägen, Wiederholen, Aufsagen beinhalten, die in diesem Fall evtl. genau in dieser Reihenfolge abgearbeitet werden.

Experimentelle Forschung hat zum Ziel, die dabei getroffenen Annahmen zu überprüfen. Über die Introspektion hinaus gehend sollen eigene Intuitionen objektiviert und quantifiziert werden. Dies kann auch mittels empirischer Daten erreicht werden, die auf der Beobachtung von Personen in ihrem Alltag basieren. Z.B. kann man in einer Feldstudie erfassen, ob die Wegweisung in einem Flughafen gelungen ist, indem man zählt, wie häufig Passagiere stehenbleiben oder eine falsche Abzweigung einschlagen. Auch die Fragebogenmethode ist dafür geeignet, bei der die subjektive Einschätzung der Teilnehmer erhoben wird.

Im Gegensatz zu qualitativen Befragungen und Feldstudien versucht man in experimenteller Forschung, das Verhalten von Versuchspersonen als direkte Folge einer *Manipulation* oder Variation von geeigneten Faktoren zu beschreiben. Wichtig dabei ist, dass die Versuchsleiter Kontrolle über diese Faktoren haben, während andere Einflussgrößen, die sogenannten Störvariablen oder konfundierenden Variablen, so weit wie möglich ausgeschlossen werden. Manchmal ergeben sich Konfundierungen zufällig. Ein Beispiel ist ein unbeabsichtigter Altersunterschied in zwei Gruppen von Versuchspersonen. Es können jedoch auch konfundierende Faktoren ein Ergebnis beeinflussen, an die bei der Ausarbeitung des Experiments einfach nicht gedacht wurde. Die wichtigsten Störvariablen, die in kognitionspsychologischen Experimenten kontrolliert, ausbalanciert oder randomisiert werden, sind Bearbeitungszeit, Reihenfolge der Aufgaben und demographische Eigenschaften der Versuchspersonen.

Die manipulierten Faktoren nennt man *unabhängige* Variablen, die Verhaltensmessung *abhängige* Variablen. Zusammengenommen bilden diese Variablen das *Design* des Experiments. Die abhängigen Variablen können in behaviorale und neurowissenschaftliche Methoden gruppiert werden.

Behaviorale Messungen

Das Ziel bei der Auswahl von abhängigen Variablen ist, eine möglichst objektive Messung einer Verhaltenweise zu ermöglichen. Dies erfordert vor allem die Quantifizierung des Verhaltens. Die Abbildung eines theoretischen Konzepts auf eine messbare Größe nennt man *Operationalisierung*. Die Operationalisierung ist ein wichtiger Schritt im experimentellen Design, der immer kritisches Hinterfragen erfordert. Im obigen Beispiel könnte man z.B. die Versuchspersonen auf einer Skala einschätzen lassen, wie schwer es ihnen gefallen ist, das Gedicht zu lernen. Eine objektivere Messung wäre zu zählen, wie viele Fehler beim Rezitieren gemacht wurden.

Die Fehlerrate ist eines der wichtigsten behavioralen Maße. Um die Kodierung zu erleichtern, werden häufig forced-choice-Verfahren angewendet, bei denen zwei (ja/nein) oder wenige Antwortalternativen ausgewählt werden müssen. Die Fehlerrate gibt zwar Aufschluss über das Ergebnis nach Bearbeiten der Aufgabe („off-line“), nicht aber Information darüber, wie es während der Aufgaben-Bearbeitung („on-line“) zustande kam. Daher werden meist zusätzlich subtilere Masse aufgezeichnet, allen voran die Messung von Reaktionszeiten.

Unter chronometrischen Verfahren versteht man alle Messungen der Verarbeitungsgeschwindigkeit, wie z.B. Lesezeiten, Entscheidungszeiten, die Reaktionszeit bis zur Aussprache eines Wortes, uvm. Die genaue Messung dieser Zeiten kann Aufschluss über die relative Taktung der an einer Aufgabe beteiligten Prozesse geben. Wichtig ist dabei, dass der Interpretation immer eine theoretische Annahme zugrunde liegen muss. Darüber hinaus werden RTs erst durch eine geeignete Manipulation der unabhängigen Variablen informativ.

Es gibt eine Reihe von Standard-Verfahren in der Kognitionspsychologie, die man auch *Paradigmen* nennt. Aufmerksamkeitssteuerung wird oft mit Doppelaufgaben (dual-task paradigm) oder Aufgabenwechsel-Experimenten getestet. Beim Priming-Paradigma wird die Reaktionszeit auf einen Zielreiz durch die vorherige Präsentation eines relationalen Reizes verkürzt, was auf eine Verknüpfung der beiden Reize in der mentalen Repräsentation schließen lässt. Interferenz-Paradigmen werden genutzt, um zu prüfen, ob ein kognitiver Prozess automatisch ist oder um zu testen, wie zwei Prozesse miteinander interagieren. Ein klassisches Beispiel ist der sogenannte Stroop-Effekt [196], bei dem das Benennen der Schriftfarbe eines Wortes (z.B. „blau“) länger dauert, wenn das Wort ein Farbadjektiv ist (z.B. „rot“). Dieser Effekt zeigt, dass das Lesen und der damit verbundene Zugriff auf die Wortbedeutung nicht unterdrückt werden können, sondern automatisiert sind.

Reaktionszeitmessungen in solchen und ähnlichen Paradigmen erfordern meist explizite Instruktionen, auf dargebotene Reize z. B. durch einen Knopfdruck auf einer Computertastatur zu reagieren. Solche Experimente werden oft kritisiert, weil die Aufgabenanforderungen künstlich erzeugt werden. Die Versuchsleiterin kann nicht sicher sein, dass ein Prozess erfasst wird, der auch ohne die – manchmal komplexe – Experimentsituation ablaufen würde. Ein Beispiel für eine behaviorale Methode, die ohne solche speziellen Instruktionen auskommt, ist die Messung von Augenbewegungen beim Lesen [167]. Basierend auf der Annahme, dass die Blickrichtung direkt den Aufmerksamkeitsfokus anzeigt, geben Augenbewegungen unmittelbare Information über die in diesem Moment ablaufenden Verarbeitungsprozesse. *Eye-tracking* liefertreichere Daten als reine Reaktionszeitmessungen. Beim Lesen zum Beispiel, kann zusätzlich zur Lese-dauer erfasst werden, wie oft der Leser zurückspringt, wie häufig ein Satz oder Satzteil gelesen wird, oder ob Wörter übersprungen werden.

Da experimentelle Forschung das Ziel hat, verallgemeinerbare und replizierbare Aussagen zu erzeugen, werden die Ergebnisse mittels statistischer Verfahren geprüft. Meist werden inferenzstatistische Tests genutzt, z.B. Varianzanalysen oder t-Tests. Aufgrund von Annahmen über die Wahrscheinlichkeiten von zufälligen Schwankungen prüfen diese Tests, ob ein Ergebnis zufällig zustande gekommen sein könnte, oder ob tatsächlich ein Einfluss der unabhängigen Variablen nachweisbar ist. Nur falls die statistische Prüfung ein Ergebnis als signifikant ausweist, ist eine Interpretation sinnvoll. Da dieses Vorgehen manchmal dazu führt, dass sogenannte Null-Ergebnisse nicht berichtet werden, entwickelt man auch alternative Verfahren (z.B. Bayes-Statistik), die jedoch noch nicht die gleiche Bedeutung erlangt haben.

Neurowissenschaftliche Methoden

Zusätzlich zu den behavioralen Methoden haben sich in den letzten 20–25 Jahren neurowissenschaftliche Methoden etabliert. Der Teilbereich der Kognitionswissenschaft, der sich dieser Methoden bedient, wird auch *kognitive Neurowissenschaft* oder *Neurokognition* genannt. Inzwischen ist eine Beschreibung kognitiver Funktionen ohne Verweis auf neurowissenschaftliche Befunde fast nicht mehr denkbar. Gute Einführungen sind [216] oder [176], ausführlichere methodisch-technische Details findet man in [107], und eine Einführung in die klinischen Aspekte gibt [71].

Lange bevor die technischen Möglichkeiten für direkte Erfassung von Aktivität im lebenden Gehirn entwickelt waren, wurde versucht, mittels Patientenbeobachtung eine Verbindung zwischen Neuroanatomie und Funktion abzuleiten, also den Hirnarealen kognitive Funktionen zuzuordnen. Hirnschädigungen, – etwa nach Schlaganfällen oder traumatischen Kopfverletzungen –, können spezifische Defizite verursachen, die durch den Ausfall der Funktion der lädierten Areale bedingt sind. Klassische – und bekannte – Beispiele sind Sprachstörungen nach linkshemisphärischen Läsionen, wie sie im 19. Jahrhundert von Paul Broca beschrieben wurden, oder der berühmte Fall von Phineas Gage, einem Bahnharbeiter, der Persönlichkeitsveränderungen nach einer Frontalhirnschädigung erlitt. Die Zuordnung dieser Defizite zu speziellen Hirnarealen konnte zu dieser Zeit nur in einer post-mortem Untersuchung des Gehirns vorgenommen werden. Neuere Verfahren (z.B. Computer-Tomographie, SPECT oder Magnetresonanztomographie) ermöglichen die Darstellung der strukturellen Anatomie des Gehirns, und somit eine Eingrenzung der lädierten Areale, schon kurz nach dem Ereignis.

Der stärkste Hinweis auf eine qualitative Differenzierung zweier postulierter Prozesse A und B ist die *doppelte Dissoziation*, die vorliegt, falls ein Patient ein Defizit in Prozess A aufweist, aber B intakt ist, und bei einer zweiten Patientin das umgekehrte Muster nachgewiesen werden kann. Ein – wiederum klassisches – Beispiel liefert die Neurolinguistik. Für Aphasien, d.h. erworbene Sprachstörungen nach Hirnschädigung, ist inzwischen ausreichend dokumentiert, dass sowohl relativ isolierte Grammatik-Defizite vorkommen, als auch Defizite, die speziell die Wortbedeutung betreffen. Eine Dissoziation legt eine qualitative Unterscheidung der beiden Prozesse nahe, auch wenn diese noch nicht unbedingt einem lokalisierbaren Hirnareal zugeordnet sein müssen.

Trifft eine Diagnose von spezifischen Defiziten mit einer genauen Lokalisation der betroffenen Hirnregionen zusammen, wird dies als Evidenz für die *Notwendigkeit* des betroffenen Areals für die jeweilige Funktion interpretiert. Wichtig dabei ist, dass diese sogenannte Differentialdiagnose eine genaue Modellvorstellung über die beteiligten Prozesse voraussetzt.

Zusätzliche strukturelle Untersuchungen der Hirnanatomie bzw. der Hirnschädigungen werden auch im Gruppenvergleich betrachtet. Eine Methode ist die Morphometrie, bei der die (relative)

Größe von Hirnarealen bestimmt wird, um daraus funktionelle Schlussfolgerungen zu ziehen. Eine weitere strukturelle Methode ist die der Läsionsüberlappung. Da bei individuellen Patienten die Läsionen selten ausschließlich ein eng begrenztes Gebiet betreffen, sondern sich oft auch in subkortikale Areale und in die weiße Substanz (d.h., in Faserverbindungen) erstrecken, kann ein Funktionsdefizit selten eindeutig einem umgrenzten kortikalen Areal zugeordnet werden. Daher werden Patienten gruppiert, die ein ähnliches Defizit aufweisen und untersucht, ob bzw. wo ihre Läsionen überlappen (z.B. [103]).

In der Tierforschung ist es möglich – aber keineswegs unumstritten –, die Hypothesen durch Erzeugen von Läsionen direkt zu testen, also durch die Resektion einer umschriebenen Hirnstruktur. Mit ähnlicher Zielsetzung wird seit wenigen Jahren beim Menschen die Transkranielle Magnetstimulation (TMS) verwendet. Ein Magnet-Puls, der auf die Kopfoberfläche aufgebracht wird, erzeugt eine „transiente Läsion“ in dem darunter liegenden Hirnareal. Beeinträchtigt diese Stimulation die Performanz in einer experimentellen Aufgabe, kann ein kausaler Zusammenhang zwischen der Aktivität des Hirnareals und dem kognitiven Prozess abgeleitet werden (s. [217]).

Physiologische Methoden messen physikalische Parameter auf der Kopfoberfläche, die durch neuronale Aktivität beeinflusst werden. Bei der Elektroenzephalographie (EEG) wird der durch das Feuern von Neuronenverbänden erzeugte Strom gemessen, bei der Magnetenzephalographie (MEG) das dadurch entstehende Magnetfeld. Um kleinste Signale vom Hintergrundrauschen trennen zu können, wird eine große Zahl ähnlicher Stimuli präsentiert. Mittelung der EEGs ergibt ereignis-korrelierte Potentiale (EKPs). Diese Komponenten lassen sich anhand ihrer Verteilung über den Kopf, die Polarität, die Amplitude und die Latenz beschreiben. So ist z.B. die P300 eine positive Komponente, deren höchste Amplitude bei ca. 300 ms erreicht wird, und die meist an postero-zentralen Elektroden am deutlichsten wird. Wichtig ist, dass systematische Zuordnungen der Komponenten zu Stimuluseigenschaften bzw. kognitiven Prozessen möglich sind. Die P300 wird etwa von seltenen, unerwarteten Tönen ausgelöst (oddball-Effekt), während ein unerwartetes Wort eine andere Komponente, die N400, zur Folge hat. Die zeitliche Auflösung dieser Methode ist ausgezeichnet, aber die räumliche Verteilung über die Kopfoberfläche lässt (zumindest ohne zusätzliche aufwändige Modellierungen) keinerlei Aussage über die Lokalisierung der Quelle zu. Ein Vorteil gegenüber behavioralen Methoden ist, dass das EKP kognitive Prozesse häufig auch ohne explizite experimentelle Aufgabe messbar macht.

Zusätzlich zu Patientenstudien und physiologischen Verfahren werden sogenannte *bildgebende Verfahren* als experimentelle Methoden zur Untersuchung von Hirnfunktionen verwendet. Dieser Ausdruck reflektiert die Gepflogenheit, Ergebnisse solcher Untersuchungen als Abbild des Gehirnes darzustellen. Die bekannteste – und inzwischen am meisten verbreitete – Methode ist die funktionelle Magnetresonanztomographie (fMRT). Im Gegensatz zur Positron-Emissions-Tomographie (PET), die das Verabreichen einer radioaktiven Markierungs-Substanz erfordert, ist diese Methode nicht-invasiv, birgt also keine bekannten Risiken. Weiterhin erlaubt die fMRT eine genauere zeitliche Auflösung, wird also auch strikteren experimentellen Anforderungen gerecht.

Das Prinzip beider bildgebender Verfahren ist, aus dem Blutfluss im Gehirn indirekte Rückschlüsse auf die aktuelle Funktion der Nervenzellen zu ziehen. Neuronale Aktivierung erfordert erhöhten Blutfluss in der aktivierten Region. Bildgebende Verfahren machen sich diese „hämodynamische Kopplung“ zunutze. Bei der PET wird der direkte Weg des radioaktiven Kontrastmittels im Gehirn verfolgt. Bei der fMRT wird der sogenannte BOLD-Kontrast gemessen

(„blood oxygen level dependent“ contrast). Der relevante physiologische Parameter ist dabei die Sauerstoffsättigung des Blutes, die wiederum indirekten Zusammenhang mit dem Aktivierungsniveau eines Areals hat. Da die hämodynamische Antwort erst mit einer Verzögerung von ca. 4–6 Sekunden beobachtbar wird, ist die zeitliche Auflösung dieser Verfahren nicht besonders fein (s. [107], für methodische Details).

Anders als Zeiten oder Fehlerwerte haben die Messwerte im MRT keine Bedeutung per se. Wichtig für diese Verfahren ist also, dass die Ergebnisse erst im Vergleich mit geeigneten Kontrollbedingungen interpretierbar werden. Die Frage, welche Bedingungen dabei „geeignet“ sind, führt uns wieder zurück zur eingangs beschriebenen Aufgabenanalyse und Theoriebildung, die als Grundlage für das experimentelle Design dienen. Vergleicht man z.B. das Hören von Geschichten mit einer Ruhebedingung, in der die Versuchsperson still im Tomographen liegt, findet man Aktivierung vor allem im auditorischen Kortex, jedoch nicht in den Spracharealen oder gedächtnisrelevanten Strukturen. Diese werden erst „sichtbar“, wenn als Vergleichsbedingung einfache verbale Reize, wie z.B. Wortlisten präsentiert werden [49]. Warum ist dies der Fall? Die Ruhebedingung ist keinesfalls mit Nichts-Tun gleichzusetzen. Die Versuchspersonen denken an etwas, vielleicht in sprachlicher Form, so dass in beiden Bedingungen ähnliche Areale aktiv sind.

Neben dieser Subtraktionsmethode werden häufig auch parametrische Verfahren angewendet, in denen die Hirnaktivierung als Funktion einer kontinuierlichen Variable beschrieben wird. Diese Methode identifiziert Areale, in denen die Variable (z.B. Wortlänge, Aufgabenkomplexität) einen Anstieg der Aktivierung bedingt.

Neurowissenschaftliche Methoden bieten die Chance, Kognition direkt mit Hirnfunktionen in Zusammenhang zu bringen, auch wenn noch diskutiert wird, welche Schlussfolgerungen über kognitive Prozesse zulässig sind (z.B. [119, 160]). Neurowissenschaftliche Daten sind multidimensional: Neben quantitativen Ergebnissen („mehr Aktivierung“, „größere Amplitude“, „längere Latenz“) erhält man Lokalisationsinformation, sowie – in neueren Studien – Aufschluss über die funktionelle Konnektivität in einem aktivierten Netzwerk. Zusätzlich zu quantitativen Unterschieden (z.B. der Reaktionszeiten) ermöglichen physiologische und bildgebende Verfahren damit die Dissoziation von qualitativ unterscheidbaren Prozessen. Für manche Fragestellungen lassen sich also kognitive Modellannahmen leichter überprüfen, als dies mit behavioralen Methoden möglich ist. Bei anderen Fragestellungen sind jedoch erprobte behaviorale Paradigmen vorzuziehen.

2.1.2.3 Modellierungstechniken

Das grundlegende Paradigma der Kognitionswissenschaft ist die Gleichsetzung geistiger Prozesse mit Prozessen der Informationsverarbeitung. Aber wie kann man sich die „Architektur“ dieser Informationsverarbeitung, also im Grunde die Architektur menschlicher Kognition, vorstellen? Eine Vielzahl konkreter Vorschläge macht deutlich, dass es dabei nicht nur um eine Computermetapher geht, sondern um die ernst gemeinte Annahme, dass kognitive Prozesse Berechnungsvorgänge sind. Diese Berechnungsvorgänge lassen sich dabei in mindestens drei Ebenen unterscheiden [127, 131]: Die erste Ebene, die sogenannte *Berechnungsebene*, repräsentiert abstrakt, was berechnet werden soll; es werden also die Merkmale und Ziele im Problem spezifiziert. Die zweite Ebene, die *algorithmische Ebene*, gibt an, wie diese Berechnung algorithmisch realisiert wird. Die dritte Ebene, die *Implementationsebene*, spiegelt die biologische Realisierung wider, d.h. die neuronale Implementation. Diese drei Ebenen werden auch als *semantische*

Ebene, syntaktische Ebene und physische Ebene bezeichnet [131, 164]. Alle drei Ebenen spielen dabei zunehmend eine Rolle. Im Grunde gesehen ist die Methodik der kognitiven Modellierung ein Dreischritt: Zunächst werden bestimmte psychologische Phänomene bzw. Effekte identifiziert. Diese werden dann in einem zweiten Schritt durch ein kognitives Modell reproduziert und erklärt. In einem dritten Schritt kann dann eine empirische Überprüfung des kognitiven Modells stattfinden, indem neue, noch nicht empirisch belegte Vorhersagen des Modells experimentell überprüft werden. Allerdings sind kognitive Modelle nie nur einfach *Simulationsmodelle*. Simulationsmodelle reproduzieren nur die bestehende experimentelle Datenlage. Ein gutes kognitives Modell ist weitgehend unabhängig von experimentellen Daten und besitzt allgemeine Strategien, aus denen heraus die experimentellen Daten (Reaktionszeiten und gegebene Antworten) generiert werden können. Zusammengefasst lässt sich die Güte eines kognitiven Modells durch mehrere Kriterien bestimmen [165]: Der erste Punkt ist die weitgehende Unabhängigkeit der Modellierungsprinzipien des kognitiven Modells von experimentellen Daten; ein Modell ist nie nur eine post-hoc Erklärung experimenteller Daten. Der zweite Punkt ist die Anzahl erklärbarer Phänomene bzw. relevanter Effekte. Der Abdeckungsgrad kann weiter spezifiziert werden in Produkt-, Zeit-, und Zwischenschrittkorrespondenz, d.h. wird das gleiche Ergebnis generiert, geschieht dies in ähnlicher Zeit und entsprechen sich die Zwischenschritte [185]. Der dritte Punkt ist die Berücksichtigung inter-individueller Differenzen. Ein Beispiel dafür ist die unterschiedliche Arbeitsgedächtnisgröße. Der vierte Punkt ist die Verallgemeinerungsfähigkeit des Berechnungsmodells. Der letztere Punkt bedeutet insbesondere, ob sich neue und bisher ungetestete Vorhersagen bezüglich bestimmter Phänomene machen lassen. Im Folgenden werden wir verschiedene Modellierungsansätze betrachten.

Kognitionspsychologische Ansätze

Das traditionsreichste Modell in der KI wie in der kognitiven Psychologie ist das der aufeinanderfolgenden Verarbeitungsstufen. Neisser [143] charakterisierte sie bissig als „processing, more processing, still more processing...“. Eine solche Kaskaden-Architektur ist nicht mit serieller (vs. paralleler) Verarbeitung zu verwechseln. Die frühen Stadien der Verarbeitung visueller Reize zeigen, dass eine kaskadierte Architektur sehr wohl mit hochparalleler Verarbeitung gekoppelt sein kann. Dieses Modell, das in seiner einflussreichsten Fassung [8] in strikter Analogie zu herkömmlichen Computerarchitekturen gestaltet wurde, dominierte die kognitionspsychologische Forschung bis in die achtziger Jahre. Sein Hauptkennzeichen ist eine Art Central Process Unit (*CPU*), das Kurzzeitgedächtnis, dessen angenommene Kapazität von ungefähr 7 Einheiten (wobei jede Einheit ein bekanntes Konzept oder Faktum ist) recht gut empirischen Ergebnissen [136] entspricht. Eine Weiterentwicklung stellt das Strukturmodell des Arbeitsgedächtnisses von Baddeley [9, 11] dar.

Klassische Symbolverarbeitungsmodelle

Tatsächlich nahm Turing bei der Entwicklung der Turingmaschine auch den Menschen zum Vorbild: „For the present I shall only say that the justification lies in the fact that the human memory is necessarily limited“ [212]. Eine Position, die anfänglich auch von Putnam vertreten, dann aber zurückgezogen wurde [162]. Grundsätzlich stellen alle Berechnungsmodelle – also alle KI-Architekturen und konnektionistischen Netze – mögliche Architekturen für kognitive Modelle dar. Eine Begründung dafür findet sich in der sogenannten *Physical Symbol System Hypothesis* [147]: „A physical symbol system has the necessary and sufficient means for general intelligent action“. Überblickshalber seien kurz erwähnt:

1. *Kaskadenarchitektur* [27]. Hier wird die serielle Verarbeitung ergänzt um die Möglichkeit einer Rückkopplung jeweils zwischen benachbarten Verarbeitungsschritten.
2. *Hierarchische Architektur*. Dies ist wohl das in der KI verbreitetste Verarbeitungsmodell, bei dem die Kontrolle der gesamten Verarbeitung durch ein besonderes Modul („Supervisor“) gewährleistet wird.
3. *Produktionsregelsysteme* [5, 114, 200] ergänzen die direkte Kontrolle, die durch einen Interpreter realisiert wird, durch eine zentrale Datenstruktur, die verschiedene Module für kognitionspezifische Aufgaben umfassen („Arbeitsgedächtnis“).
4. *Blackboard-Architektur* [44]. In solchen Architekturen kommunizieren mehrere autonome Subsysteme durch eine gemeinsam zugängliche Datenstruktur, die sogenannte Blackboard.
5. *Multi-Agenten-Modelle* [90] dezentralisieren die Kommunikation, indem autonome Subsysteme (Agenten, *actors*) direkt Nachrichten austauschen; diesem Paradigma entspricht die Softwaretechnik des objektorientierten Programmierens. Hierher gehören auch quasi-konnektionistische Modelle wie das von [123].
6. *Subsumptionsarchitektur* [21]. Diese basiert auf selbständigen Funktionseinheiten, über die weitere, diese aktivierende oder hemmende Einheiten, geschichtet werden. Insbesondere die Modularität spielt eine entscheidende Rolle. Dabei ist eine Besonderheit die Abwesenheit expliziter Repräsentation und abstrakter Inferenz. Die wesentlichen Eigenschaften sind emergent.
7. *Agenten-Architekturen* [201], insbesondere solche für soziale Agenten (z.B. INTERRAP [142]).

Kognitive Architekturen

Eine Möglichkeit, den Begriff der *Kognitiven Modellierung* zu fassen, ist ihn als Algorithmisierung psychologischer Theorien in einer kognitiven Architektur zu verstehen. Das Ziel ist es, neben der Identifikation von Widersprüchen in psychologischen Theorien, ein algorithmisches und zugleich psychologisch fundiertes Erklärungsmodell zu entwickeln, welches die präzise Reproduktion empirischer Ergebnisse (Reaktionszeiten und Fehlerraten) gestattet. In neuster Zeit ermöglichen kognitive Architekturen auch die Vorhersage aufgabenspezifischer Gehirnaktivierungen von fMRI-Untersuchungen wie beispielsweise ACT-R 6.0 oder 4-CAPS (s.u.).

Alle sogenannten „kognitiven Architekturen“ gehen letztlich auf das an der Carnegie Mellon University bereits in den sechziger Jahren in der Gruppe um Newell und Simon entwickelte Programm GPS (General Problem Solver, ein Programm, das Mittel-Ziel-Analyse als Suchheuristik verwendet) und dessen Reimplementierung als Produktionsregelsystem zurück [145]. Produktionsregelsysteme bestehen aus (i) einem Arbeitsgedächtnis, das als Input wie als Output für die in (ii) einem Langzeitgedächtnis gespeicherten Produktionsregeln dient. Produktionsregeln sind Wenn-dann-Regeln, deren Bedingungsteil mit dem aktuellen Inhalt des Arbeitsgedächtnisses verglichen wird. Bei Übereinstimmung kann die Regel ausgeführt werden („feuern“) und so den Inhalt des Arbeitsgedächtnisses verändern. (iii) Ein Regelinterpreter sorgt für die Ausführung der Produktionsregeln und z. B. für den Fall, dass mehrere Regeln feuern können, für Konfliktlösung. Zur Realisierung der Mittel-Ziel-Analyse wurde früher ein *goal stack* als besonderer Bestandteil des Arbeitsgedächtnisses benutzt.

Aus dieser Grundarchitektur haben Newell und Mitarbeiter das System SOAR¹ (States, Operators, And Reasoning) entwickelt [115, 144]. Dieses System zeichnet sich gegenüber seiner

¹ <http://sitemaker.umich.edu/soar/home>.

Urform vor allem durch Lernfähigkeit aus. Dabei wird *Produktionsregelkompilation* benutzt, d.h. zur Zielerreichung nacheinander ausgeführte Produktionsregeln werden bei Erfolg zu einer neuen Produktion kompiliert. Der Regelspeicher ist den Zielen entsprechend partitioniert. Zahlreiche anspruchsvolle Anwendungen sind inzwischen in SOAR implementiert worden, z.B. eine Luftkampf-Simulation, die zur Pilotenausbildung bei der USAF verwendet wird [207]. Die aktuelle Version SOAR 9 integriert nichtsymbolische Repräsentationen und weitere Lernmechanismen [113]. Ein aktueller Überblick über die Architektur findet sich in [114].

Die heute am weitesten verbreitete kognitive Architektur ist ACT-R 6.0² [5, 6], welches die aktuelle Version des ebenfalls an der Carnegie Mellon University entwickelten Produktionsregelsystems ACT [2] samt seinen Nachfolgern ACT* (Adaptive Control of Thought [4]) und ACT-R (Atomic Components of Thought [7]) ist. Dabei ist es zugleich eine hybride Theorie (mit symbolischen und subsymbolischen Elementen) menschlicher Kognition. Die Architektur umfasst eine Reihe von Grundannahmen, die es erlauben, kognitive Phänomene verschiedener Bereiche zu modellieren. Die einzelnen Elemente (sogenannte Module) sind dabei psychologisch fundiert und schränken zugleich mögliche kognitive Modelle sinnvoll ein. In seiner gegenwärtigen Form zeichnet sich diese Architektur dadurch aus, dass sie außer dem „prozeduralen Gedächtnis“ (d.h. dem Regelspeicher) ein deklaratives Gedächtnis besitzt, nämlich ein konnektionistisches Netzwerk, dessen Knoten Konzepte sind, auf die per Aktivationsausbreitung zugegriffen werden kann. Die Architektur besteht aus Modulen für Wahrnehmung (z.B. *visual*, *aural*), Ziel- und Subzielrepräsentationen (*goal*, *imaginal*) und Schnittstellen (sogenannten Buffern), auf die die Produktionsregeln zugreifen können. Die Anwendungen dieses Modells reichen dabei von kognitionspsychologischen Modellen des Lernens und Gedächtnisses, des Problemlösens, der Wahrnehmung, der Aufmerksamkeitssteuerung bis hin zur Mensch-Computer-Interaktion (HCI).

Weitere ähnliche Architekturen sind EPIC³ [135] und die CAPS⁴-Architekturen [100, 210], die explizit die knappe Kapazität des menschlichen Arbeitsgedächtnisses berücksichtigen. Ein grundsätzlicher Kritikpunkt an den gängigen „kognitiven Architekturen“ ist, dass sie zu wenige Constraints aufweisen und teilweise Turing-äquivalent sind [4]. Das führte nach einer anfänglichen Euphorie zu der Einsicht, dass drastische Einschränkungen notwendig sind, um plausiblere und vielleicht kognitiv-adäquate Modelle zu beschreiben. Eine der neusten hybriden Architekturen ist Clarion⁵ [200]. Dieses System hat analog zu ACT-R eine modulare Struktur und alle Subsysteme basieren auf neuronalen Netzen – im Gegensatz zu ACT-R, wo dies nur für Aktivierungen deklarativer Gedächtnisinformation gilt. Da neuronale Netze nicht einfach zu programmieren sind, sind die Beschränkungen in Clarion emergent. Eine Besonderheit stellt dabei dar, dass die kognitive Architektur Clarion sowohl zwischen implizitem und explizitem Wissen als auch zwischen impliziten und expliziten Motiven unterscheidet (Triebes versus bewusste Ziele) [86]. Vertiefende Diskussion über weitere kognitive Architekturen und Herausforderungen finden sich in [116, 202].

Es gibt auch rein konnektionistische Ansätze der Modellierung menschlicher Kognition, die sich die neuronale Struktur als Vorbild nehmen, sogenannte Künstliche Neuronale Netze.

Künstliche Neuronale Netze

² <http://act-r.psy.cmu.edu/>.

³ <http://web.eecs.umich.edu/~kieras/epic.html>.

⁴ <http://www.ccbi.cmu.edu/4CAPS/index.html>.

⁵ <http://www.cogsci.rpi.edu/~rsun/clarion.html>.

Eine andere Perspektive der Modellierung umfasst dabei Künstliche Neuronale Netze (KNN). Sie haben die menschliche neuronale Struktur und Lernfähigkeit als Vorbild. McCulloch und Pitts [133] konnten dabei beispielhaft zeigen, dass sich logische und arithmetische Funktionen durch solche KNNs berechnen lassen. Die Hebb'sche Lernregel [85] bildet dabei die Grundlage nahezu aller Lernverfahren und besagt vereinfacht, dass eine Verbindung zwischen zwei Neuronen verstärkt wird, wenn beide Neuronen aktiv sind. Anfänglich wurden KNNs aus zwei Schichten von Knoten (*Perzeptron*) betrachtet. Allerdings hat ein Perzeptron aus informatischer Perspektive drastische Beschränkungen wie u.a. Minski und Papert [137] systematisch gezeigt haben. So benötigen eine Vielzahl von Problemen Berechnungsmodelle, die nicht nur linear trennbare Muster unterscheiden, sondern auch die Möglichkeit, den Booleschen Operator XOR zu berechnen. Diese Beschränkungen führten zur Entwicklung von Mehrschichtmodellen mit dem *Backpropagation*-Lernalgorithmus [132]. Trotzdem ist das formale Neuron eine drastische Vereinfachung der Hirnneuronen. Es wird nur die elektrische Erregungsleitung modelliert, es spielen also weder Neurotransmitter in Synapsen, noch hormonelle Wechselwirkungen eine Rolle. Die KNN-Architekturen sind von ihrer Struktur regelmäßig im Gegensatz zu realem Nervengewebe, welches scheinbar chaotisch ist. Die meisten Vertreter des Konnektionismus (zuerst [190]) sehen KNNs überwiegend als Berechnungsmodelle an und nicht als Modelle biologischer Realität. Eine gänzlich andere Art der Modellierung menschlicher kognitiver Leistungen sind Bayessche Modelle. Diese haben keine spezifischen Annahmen über die Struktur des menschlichen Gedächtnisses oder seine neuronale Implementation. Es sind rein mathematische Modelle.

Bayessche Modellierungen

Eine Vielzahl induktiver Probleme wie z.B. Modellierungen induktiven Lernens [208], kausale Inferenzen [76, 77, 195], Spracherwerb und Sprachverarbeitung [25, 223] und semantisches Gedächtnis [194] lassen sich durch Bayessche Modelle erklären. Der Ausgangspunkt und die Grundidee Bayesscher kognitiver Modelle ist die Frage, wie ein kognitiver Agent seine momentanen Annahmen im Lichte beobachteter Daten gegebenenfalls revidiert [75]. Die Wahrscheinlichkeit, dass eine gegebene Datenmenge d durch eine zunächst vorläufige Hypothese h erklärt wird (also $p(h | d)$) lässt sich durch die Bayessche Formel durch die Wahrscheinlichkeit für $p(h)$ und der Wahrscheinlichkeit, die Daten d zu beobachten, falls die Hypothese h wahr ist (also $p(d | h)$), berechnen. Vertreter Bayesscher Ansätze weisen darauf hin, dass aus der „Behauptung, dass der menschliche Geist nach Bayesschen Prinzipien lernt und schlussfolgert, nicht die Behauptung folgt, dass der menschliche Geist Bayessche Inferenzen implementiert“ [209]. Bayessche Modelle setzen dabei oftmals Heuristiken und Hintergrundwissen voraus, z. B. für menschliches syllogistisches Schlussfolgern [149].

Bayessche Modelle stellen im Gegensatz zu den oben präsentierten symbolischen und hybriden Modellierungen keine eigentlichen kognitiven Prozessmodelle dar [5, 97]. Beispielsweise spielen kognitive Beschränkungen des Arbeitsgedächtnisses (*cognitive bottlenecks*) keine Rolle wie Anderson, einer der Begründer der „rationalen Analyse“, bemerkt [5]. [75] weisen darauf hin, dass „eine Vielzahl bekannter konnektionistischer Algorithmen eine Bayessche Interpretationen“ haben und so könnten diese als ein erster Ansatz für eine neuronale Modellierung Bayesscher Ansätze dienen. Dennoch bleiben diese Fragen nach einem Prozessmodell nach wie vor offen.

Kausale Bayes-Netze erlauben die Repräsentation struktureller Aspekte zwischen Zufallsvariablen [155, 156]. Die kausale Struktur, also die Abhängigkeiten der Zufallsvariablen, wird dabei

durch einen azyklischen gerichteten Graphen repräsentiert. Dabei repräsentieren die gerichteten Graphen den Zusammenhang zwischen Ursache und Effekten. Solche kausalen Bayes-Netze haben nicht nur eine rein probabilistische Beziehung zwischen den Variablen, sondern durch die kausale Struktur auch eine klare Vorhersagekraft, die dann durch empirische Daten überprüft werden können [78]. Die Stärke dieses Ansatzes liegt insbesondere darin, eine Modellierung auf der Berechnungsebene darzustellen und doch robust genug gegenüber verrauschten Daten zu sein. Damit lassen sich Bayessche Modelle in die Kategorie der hybriden Ansätze, wie beispielsweise die kognitive Architektur ACT-R, einreihen.

Kognitive Modellierung ist also sehr vielschichtig und kann dabei rein *symbolisch, konnektivistisch* oder *hybrid* sein und dabei eine oder mehrere Marrsche Ebenen umfassen. Sie kann spezifische Annahmen über die Struktur des Arbeitsgedächtnisses integrieren, individuelles Verhalten oder wahrscheinlichkeitsbasiert das Verhalten einer Gruppe vorhersagen. Die verschiedenen Ansätze spiegeln dabei verschiedene Bereiche menschlicher Kognition wider und bieten gegenüber reinen Lehrbuchtheorien auch die Möglichkeit der empirischen Falsifizierbarkeit kognitiver Theorien.

2.2 Menschliche Kognition

Unser unmittelbares Erleben zeigt uns die Welt „da draußen“, samt uns selbst als dem darin handelnden Akteur. Doch ist das von uns bewusst Erlebte nur ein Teil dessen, was an kognitiven Prozessen in unserem Gehirn abläuft, ja sogar, wenn man vielen Autoren aus der Philosophie und den Neurowissenschaften folgt, eine Täuschung, ein „cartesianisches Theater“ ([34], unter Bezug auf die von Descartes 1637 vorgenommene Annahme einer physischen und einer geistigen Substanz, seither als Dualismus bezeichnet). Die Kognitionswissenschaft hat jedoch ein anderes Modell vor Augen, in unverkennbarer Analogie zum Computer: Kognitive Prozesse sind demnach (vereinfacht) als die Software anzusehen, die auf einem neuronalen Computer, dem Gehirn, ausgeführt wird. Das heißt erstens, Abkehr vom Dualismus: Es gibt nur die physische Materie und keine geistige „Substanz“. Aber es gibt, implementiert im Gehirn, eine geistige Funktionalität. Diese, in der neueren Philosophie als Funktionalismus bekannte Position gibt uns die Möglichkeit, kognitive Prozesse in (relativer) Unabhängigkeit von ihrer Implementierung zu untersuchen. Newell und Simon [146] haben dies als *Physical Symbol Systems Hypothesis* zur Grundlage der Kognitionswissenschaft und der KI erklärt.

Die Inhalte unseres Bewusstseins können als Ergebnis einer Modellbildung begriffen werden, wobei ein Weltmodell und ein Selbstmodell unterschieden werden können [134]. Der Nutzen solcher (für uns selbst nicht als Modell erkennbarer) Modelle liegt beispielsweise in der Handlungsplanung, die von einem „Probehandeln“ (Freud) mit Abschätzung möglicher Konsequenzen unseres Tuns profitiert. Indem die Kognitionswissenschaft Modelle unserer kognitiven Tätigkeit konstruiert, betreibt sie also eine Modellierung zweiter Ordnung. – Obwohl Bewusstsein eine grundlegende Komponente unseres Erlebens darstellt und seit Husserl in der sog. Phänomenologie (einer wichtigen, neuerdings auch mit der Kognitionswissenschaft verbundenen philosophischen Richtung, vgl. [63]) systematisch behandelt wird, mangelt es noch immer an einer zureichenden neurowissenschaftlichen Theorie des Bewusstseins, wenn auch nicht an interessanten Ansätzen dafür.

Viele psychologische und neurowissenschaftliche Untersuchungen haben gezeigt, dass die zu grunde liegende funktionale Architektur des Gehirns sehr komplex ist. Während frühe Modelle (z.B. [8]) strikt an der seriellen Architektur damaliger Computer orientiert waren, sind heute Parallelen zu aktuellen Computerarchitekturen mit mehreren Prozessoren und teilweise autonomen Subsystemen auch das Vorbild moderner Theorien des Gehirns. Felleman und van Essen [47] beispielsweise fraktionieren das Sehsystem von Primaten in etwa dreißig funktionelle Einheiten. Auch wird deutlich, dass unser Gehirn im Laufe der Evolution „historisch gewachsen“ und keinesfalls ingeniermäßig entworfen ist: Motorische Kontrolle wird sowohl von einem schnellen, weil einfachen, lokalen und nicht zentral beeinflussbaren System von Reflexen ausgeübt, sondern von zwei weiteren zentralen Systemen: erstens der bewusst kontrollierten, langsamen kortikalen Kontrolle (sog. Willkürbewegungen), zweitens der bewusst nur initiierten, dann aber unbeeinflussbar ablaufenden Kontrolle durch hochgeübte motorische Programme, die im Kleinhirn gespeichert sind; die Unbeeinflussbarkeit beim Ablauf zeigt sich etwa beim Tippen von Text (wo Hochgeübte merken, dass sie dabei sind, sich zu vertippen, ohne noch korrigierend eingreifen zu können).

2.2.1 Wahrnehmung, Aufmerksamkeit, Bewusstsein, Handlungskontrolle

In der Natur wie in der Technik gibt es eine Fülle von Sensoren, die kognitive Systeme über den aktuellen Zustand ihrer Umwelt, oder auch über den ihres eigenen Körpers informieren; beim Menschen kommen zu den bekannten fünf Sinnen (Sehen, Hören, Riechen und Schmecken, Lagewahrnehmung und Gleichgewichtssinn, sowie der Tastsinn mit Druck- und Temperaturrempfindung) noch Rezeptoren für zahlreiche propriozeptive Reize hinzu, z.B. Muskelspannung, Schmerz, oder Hungergefühl. Einige biologische Sensoren verfügen über einen extrem breiten Funktionsbereich, weil sie hoch adaptiv sind: So kann das menschliche Auge nach ca. 40 Minuten in absoluter Dunkelheit bereits ein einziges Lichtquant erkennen, aber (nach einer Adaptationszeit von nur wenigen Sekunden) auch im grellen Sonnenlicht Gegenstände erkennen.

Wesentlich ist die den Sensoren nachgeschaltete neuronale Reizverarbeitung. Ein simples Beispiel ist folgendes: Im Auge muss das Licht – außer durch Medien eher zweifelhafter optischer Qualität – durch mehrere Zellschichten der Retina, bevor es auf die Rezeptoren trifft. Die resultierenden Kontrastverluste aufgrund von Beugung in diesen Zellschichten werden durch lokale neuronale Prozesse, die eine Kontrastverschärfung bewirken, noch in der Retina abgemildert. Im kortikalen Sehsystem (das im hintersten Teil des Gehirns liegt) werden dann Form-, Farb- und Bewegungsinformation zunächst getrennt analysiert und später integriert, wobei sich das Problem zeigt, wie die Repräsentation von wahrgenommenen Objekten (im unteren Temporalappen) mit der Repräsentation der örtlichen Lage im Sehfeld (im Parietallappen, also deutlich entfernt) integriert wird [138], so dass wir Objekte an bestimmten Stellen im Raum wahrnehmen; die heute bevorzugte Annahme setzt auf phasensynchrones Feuern von Neuronenverbänden [187]. Die Objekterkennung selbst wird durch Vergleich der aktuellen Ansicht mit gespeicherten Ansichten von Objekten bewirkt [172]. Auch hier ist die Wahrnehmung vom Kontext beeinflusst: In passendem Kontext wird ein Objekt schneller erkannt als alleine (z.B. ein Küchengerät in der Küche, oder ein Buchstabe im Kontext eines Wortes, der sog. Wortüberlegenheitseffekt); hingegen kann ein unpassender Kontext für die Objekterkennung hinderlich sein.

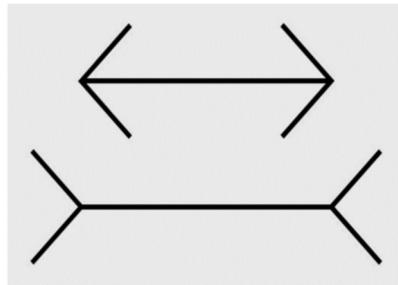


Abbildung 2.1: Müller-Lyer-Täuschung. Beide Linien sind gleich lang, sehen aber unterschiedlich lang aus.

(z.B. ein Sofa in einer Straßenszene), weil Kontexte allgemein schneller erkannt werden als einzelne Objekte [18].

Charakteristika unserer Wahrnehmung sind Adaptivität, Kontextsensitivität, Gruppierung und Konstanz. Adaptivität, also die Anpassung an die mittlere Intensität (z.B. Helligkeit oder Lautstärke) ist eine allgemeine und wünschenswerte Eigenschaft aller Sensoren. Ein Spezialfall ist die Habituation: Eine unwichtige und regelmäßig wiederkehrende Reizung (z.B. das Ticken einer Uhr) wird nach einiger Zeit nicht mehr wahrgenommen. – Zur Kontextsensitivität ein einfacher Versuch: Man stecke eine Hand einige Zeit in sehr warmes, die andere in kaltes Wasser; wenn man daraufhin beide Hände zugleich in lauwarmes Wasser steckt, empfindet die „warme“ Hand das Wasser als kalt, die andere Hand als warm. Kontext kann auch das Vorausgegangene sein; so werden die ersten Reize einer Reihe als sog. Ankerreiz [87] zum Maßstab der Beurteilung nachfolgender Reize genommen. – Die Gruppierung von Reizen ist ebenfalls eine sinnvolle Leistung unseres Wahrnehmungssystems; sie wurde bereits von der Gestaltpsychologie (ca. 1910–1935) untersucht und erfolgt nach Form, Farbe, relativem Abstand oder gemeinsamer Bewegung. Auch die Ergänzung (z.B. teilweise verdeckter Linien) gehört dazu. – Die bekannteste Konstanzeleistung ist die Größenkonstanz: Ein Mensch, der auf uns zu kommt, scheint deshalb nicht zu wachsen, weil die Größe seiner Abbildung auf der Netzhaut mit seiner wahrgenommenen Entfernung verrechnet wird.

Wichtige Darstellungen der Berechnungsprozesse im visuellen System finden sich bei [127], [153] und [124]. Es finden aufwendige und dank einer hoch parallelen neuronalen Architektur auch genügend schnelle Berechnungen statt. Dies hatte schon Helmholtz Mitte des 19. Jahrhunderts mit seiner umstrittenen Äußerung gemeint, dass Wahrnehmung auf „unbewussten Schlüssen“ beruhe modern gesprochen: auf Berechnungsprozessen. Zu diesen Vorgängen kommt noch eine erste, schnelle emotionale Bewertung des Wahrgenommenen hinzu (etwa: gut – schlecht, oder: kann ich schaffen – muss die Flucht ergreifen) [117].

Unsere Wahrnehmung wird sowohl von äußeren Reizen, als auch von unserem eigenen Wahrnehmungssystem beeinflusst. Dies wird durch die sog. optischen Täuschungen eindrucksvoll belegt. Ein Beispiel ist die Müller-Lyer-Täuschung (Abb. 2.1). Obwohl beide Linien gleich lang sind, erscheint die obere Linie deutlich länger.

Wahrnehmung steht im Dienst der Handlungskontrolle [161]. Das impliziert, dass wir nicht alles wahrnehmen, sondern vornehmlich dasjenige, was für unsere aktuellen Handlungsziele relevant ist. Drastische Beispiele dafür haben Simons und Chabris ([186]; Videos auffindbar unter „nat-

tentional blindness gorilla“) unter der Bezeichnung *inattentional blindness* gegeben: Wenn die Aufmerksamkeit auf etwas anderes gerichtet ist, übersehen wir sogar recht auffällige Dinge (im angeführten Beispiel eine Gestalt im Gorillakostüm, die sich unter einige Personen mischt, die einander Bälle zuwerfen). Aufmerksamkeit kann aktiv gelenkt werden, entsprechend unseren Absichten und Erwartungen. Ein Beispiel wäre der Fußballschiedsrichter, der nicht den langen Flug des Balles verfolgt, sondern gezielt darauf achtet, ob der wahrscheinliche Empfänger des Passes vorsichtshalber vom gegnerischen Verteidiger gefoul wird. Durch auffällige Reize (z.B. Stellen hohen Kontrasts in Bildern) kann die Aufmerksamkeit aber auch passiv angezogen werden; in bestimmten Fällen kann es zu einer sog. Orientierungsreaktion kommen (ein lauter Knall: wir schauen dorthin; hinter unserem Rücken sagt jemand unseren Namen: wir drehen uns um). Aufmerksamkeit ist selektiv; sie bewahrt das kognitive System vor Überlastung.

Mit Aufmerksamkeit können wir eigentlich immer nur eine Tätigkeit steuern. Das ist einerseits gut, macht aber Schwierigkeiten, wenn mehrere Dinge gleichzeitig zu tun sind. Hier kann man sich abwechselnd auf eine von mehreren Tätigkeiten konzentrieren, was aber Leistungseinbußen in beiden Tätigkeiten (ca. 20–30%) mit sich bringt. Alternativ können gewisse Tätigkeiten durch viel Übung (s. Abschnitt 2.2.3) automatisiert werden; sie benötigen dann als reizgesteuerte Handlungsschemata keine Aufmerksamkeit mehr. So konnte experimentell demonstriert werden, dass eine geübte Stenotypistin gleichzeitig einen schriftlich vorgelegten Text abtippen und einen anderen, ihr über Kopfhörer zugespielten Text nachsprechen kann (Shaffer, 1975).

Die Organisation aufmerksamer, bewusster Handlungskontrolle wird unter dem Stichwort „exe-kutive Prozesse“ erforscht. Die viel zitierte Studie von Miyake et al. [140] unterscheidet drei wesentliche Teilprozesse: 1. *task shifting* (Aufgabenwechsel), z.B. bei Zahlenreihen abwechselnd 3 hinzuaddieren bzw. 5 abziehen; 2. *updating* (sich Veränderungen merken), z.B. bei einer Reihe von Tönen (tief, mittel, hoch) Taste drücken, wenn ein Ton (beliebiger Höhe) zum vierten Mal erklingt; 3. *response inhibition* (Unterdrückung einer gelernten Reiz-Reaktions-Verbindung), z.B. im Stroop-Test: Man soll farbig gedruckte Farbwörter, bei denen die Druckfarbe nicht der Wortbedeutung entspricht (etwa das Wort BLAU in roter Farbe) nicht die Wörter lesen, sondern die Druckfarben benennen. Neurowissenschaftlich wird für die Realisierung solcher exekutiven Funktionen das Frontalhirn, speziell der präfrontale Cortex verantwortlich gemacht.

Aufmerksamkeit und Bewusstheit von Wahrnehmungen und Tätigkeiten gehören zusammen. Oder anders gesagt: Nur das, was wir mit Aufmerksamkeit beachten, nehmen wir auch bewusst wahr. Andererseits kann uns mehr bewusst sein als nur die aktuelle Wahrnehmung; insbesondere gilt dies für Inhalte des Gedächtnisses. Beides kann zur bewussten Handlungskontrolle dienen, nämlich beim Abwägen von Handlungsalternativen, bei der mentalen Simulation möglicher Abläufe, bei der begründbaren Entscheidung für ein bestimmtes Handeln. Daneben steht die oft unbewusste, reizgesteuerte und von automatisierten Handlungsschemata bestimmte Steuerung unseres Tuns, die allerdings nicht selten zu Fehlern führt [169]. – Aufmerksamkeit, und zwar geteilte Aufmerksamkeit (in dem Sinne, dass einer die Aufmerksamkeit des Artgenossen auf etwas lenkt, z.B. durch Zeigen) ist eine wesentliche Grundlage der sozialen und besonders der Sprachentwicklung und wird bei anderen Primaten als dem Menschen nicht beobachtet [211]; mehr darüber in den Abschnitten 2.2.5 und 2.2.6.

2.2.2 Mentale Repräsentation, Begriffe und mentale Modelle

Ohne irgendeine Art mentaler Repräsentation anzunehmen ist es nicht möglich, über Kognition zu sprechen [104], doch über den Charakter solcher Repräsentation (oder Repräsentationen) lässt sich trefflich streiten. Die klassische Formulierung stammt aus der spätmittelalterlichen Philosophie: „vox significat (rem) mediantibus conceptis“ (etwa: „ein Wort bezeichnet ein Objekt in der Welt mittels der Begriffe“). Daran knüpft der frühe Wittgenstein an, wenn er schreibt: „3. Das logische Bild der Tatsachen ist der Gedanke. – 4. Der Gedanke ist der sinnvolle Satz.“ [222] Daran hält sich auch Jerry Fodor [54], wenn er in Anlehnung an die logische Semantik und auch an Franz Brentano einen Gedanken als eine „intentionale Einstellung“ auffasst: Ein Gedanke besteht demnach erstens aus einem Inhalt (d.h. einer logischen Proposition) und zweitens aus einer Einstellung des Subjekts zu diesem Gedanken, etwa des Glaubens oder Wunsches, wobei er postuliert, dass diese Einstellung eine „algorithmische“ sei. Hiermit hat sich die klassische Kognitionswissenschaft allerdings eine Bindung an eine geradezu naive Erkenntnistheorie eingehandelt, die bald gelockert wurde. Als Alternative wird heute davon ausgegangen, dass mentale Repräsentation nichts anderes ist als Modelle der Welt, die sich – wie auch wissenschaftliche Modelle – dadurch auszeichnen, dass sie unvollständig sind und durchaus auch falsch sein können. Die Dynamik solcher Modellierung hatte schon Piaget [158] in den Kategorien der Assimilation (von Neuem gemäß der bereits vorhandenen begrifflichen Kategorien) und Akkommodation (d.h. der Veränderung kognitiver Strukturen angesichts des Scheitern von Assimilation) beschrieben.

Bezüglich der begrifflichen Strukturen war bis in die siebziger Jahre des letzten Jahrhunderts das auf Aristoteles zurückgehende Modell von Begriffen bzw. Kategorien maßstäblich, die durch Bezug auf den jeweils übergeordneten Begriff (*genus proximum*) und die hinzu kommenden Spezifizierungen (*differentia specifica*) eindeutig definiert sind, so wie die 1853 vorgelegte und viel bewunderte Einteilung der Pflanzen von Linné. Der von Eleanor Rosch [175] gelieferte Nachweis, dass Menschen gemeinhin anders denken, samt den von ihr beobachteten Prototypen-Effekten, wonach die Mitgliedschaft in einer Kategorie kein Entweder-Oder, sondern graduell ist, und zudem die Kategoriengrenzen unscharf, darf als wissenschaftliche Revolution gewertet werden. Es konnte gezeigt werden, dass Prototypen, also (ideal-)typische Beispiele für eine Kategorie, wie etwa „Apfel“ für die Kategorie „Obst“, schneller korrekt klassifiziert werden können als weniger typische Vertreter wie z.B. „Melone“, aber auch, dass Prototypen-Effekte selbst bei künstlichen Kategorien auftreten (so ist z.B. 64 eine „geradere“ Zahl als 86) und auch bei nicht fest organisierten Ad-hoc-Kategorien wie „Dinge, die man auf einem Flohmarkt verkaufen kann“. Solche Befunde weisen darauf hin, dass graduelle Unterschiede statt fester Kategoriengrenzen ein universelles Merkmal menschlichen Denkens darstellen.

Strittig blieb in den siebziger Jahren, ob die in Experimenten zur „mental rotation“ [183] nachgewiesenen anschaulichen, bildhaften oder „analogen“ Repräsentationen auch Bestandteil unseres begrifflichen Wissens sind. Anfangs war die von Pylyshyn vertretene Ansicht populärer, wonach es im Langzeitgedächtnis nur propositionale Repräsentation gäbe. Nach zehnjähriger Debatte neigt die große Mehrheit der vor allem von Kosslyn vertretenen Meinung zu, dass auch analoge Repräsentation möglich ist. – Andere fassten Begriffe als Mini-Theorien über Weltausschnitte auf, oder als Kondensat von Erfahrungen. 1983 wurde der Begriff „mentales Modell“ gleich zweifach eingeführt, einmal als Theoriegebilde [64], zum anderen spezifischer, als anschauliches Modell, das beim Lesen von Texten oder logischen Sätzen aufgebaut wird [93]. Weitere Modelle zur Repräsentation komplexer, strukturierter Sachverhalte wurden

vorgeschlagen, z.B. zur Repräsentation mehr oder weniger stereotyper Ereignisse die scripts („Drehbücher“; [179]).

Barsalou [15] hat die Begriffstheorien um anschauliche Komponenten erweitert. Prototypen sind ja häufig visuell vorstellbar (etwa ein prototypischer Stuhl) und müssen auch analog repräsentiert sein, z.B. für die Objekterkennung (s. vorigen Abschnitt). Barsalou nimmt an, dass durch Erfahrung eine integrierte Repräsentation entsteht, die propositionales Wissen und bildhafte Repräsentation enthält, ein sog. „Simulator“. Diese Theorie hat in den letzten Jahren viel Unterstützung erfahren durch Experimente, die die enge Verzahnung von Wort und Bild zeigten (z.B. [224]). Daraus ist zu schließen, dass nicht nur mehr oder weniger abstrakte Begriffe, sondern auch unterschiedliche konkrete Wahrnehmungsbilder beim Textverstehen evoziert werden, was Zwaan und viele andere zu einer Theorie der *embodied cognition* generalisiert haben, die auch die Motorik und die Emotionen umfasst. Insgesamt folgt daraus, dass die klassische Position der Kognitionswissenschaft, nämlich Symbolverarbeitung, ergänzt werden muss um körpernahe Repräsentationen. Für die kognitive Modellierung wirft das Probleme auf (s. aber Lösungsperspektiven bei Pezzulo et al., [157]), doch sei daran erinnert, dass auch in der Robotik nur unter Berücksichtigung der „Leiblichkeit“ der Roboter (d.h. ihrer sensorischen und motorischen Ausstattung) erfolgreich gearbeitet werden kann.

2.2.3 Gedächtnis und Lernen

Kognition ohne Gedächtnis kann es nicht geben. Um etwas als etwas Bestimmtes erkennen zu können, muss es verglichen werden mit den Inhalten des Gedächtnisses; das gilt für die visuelle Erkennung von Objekten ebenso wie für das Erkennen von Wörtern bestimmter Bedeutung im Fluss der Rede. Die Möglichkeit, etwas über einen längeren Zeitraum speichern und wieder erinnern zu können, ist fundamental. Sie ist auch die Grundlage allen Lernens. Zwar ist es schwierig (und kaum konsensfähig), eine Grenze zwischen nicht kognitiven und kognitiven Organismen zu ziehen, aber Lernfähigkeit und die Möglichkeit, das Gelernte unter Artgenossen zu kommunizieren, sind jedenfalls kognitiven Systemen vorbehalten.

Neuronale Grundlagen

Lernen und langfristige Speicherung werden hauptsächlich durch andauernde Veränderungen der Effizienz von Synapsen zwischen Nervenzellen realisiert, die beispielsweise – gemäß einer bereits 1949 von Donald O. Hebb aufgestellten Hypothese [85] – gesteigert wird, wenn prä- und postsynaptische Zellen gleichzeitig aktiviert sind. Nicht nur das Einspeichern neuer Information, sondern auch der Gebrauch gespeicherter Information wirkt sich verstärkend aus, wohingegen bei längerem Nichtgebrauch die „Gedächtnisspur“ (*trace*) verblasst. Von daher erklärt sich auch die retrograde Amnesie nach Schock bzw. Unfall: Was unmittelbar vorher passiert ist, konnte nicht stabilisiert werden und ist daher unwiederbringlich vergessen. Im Gegensatz zum langfristigen Behalten ist das kurzzeitige durch elektrophysiologische Prozesse realisiert, die schnell abklingen und deshalb (s.u. bei „Arbeitsgedächtnis“) nur durch andauernde Wiederholung erhalten bleiben.

Informationen werden generell verteilt gespeichert, also nicht nur in einer ganz bestimmten Synapse, sondern innerhalb von Zellverbänden aus einigen tausend Neuronen, sog. *cell assemblies*. Dies hat den Vorteil, dass beim Ausfall einzelner Zellen nicht schlagartig Informationen vergessen werden, sondern lediglich die Erinnerung etwas unschärfer wird. Ein derartiger Speicher lässt sich sehr gut durch künstliche neuronale Netze modellieren; solche Netze haben auch

die Eigenschaft des menschlichen Gedächtnisses, inhaltsadressierbar zu sein: Je präziser man weiß, was man sucht, desto effizienter ist die Erinnerung (mit den bekannten Ausnahmen gelegentlicher Blockaden). Neuroanatomisch ist das Gedächtnis weit verteilt (vgl. [69, 126, 189, ch. 4–6]).

Taxonomie des Gedächtnisses

Mindestens drei Arten des Gedächtnisses werden unterschieden: erstens die sensorischen Pufferspeicher, zweitens das Arbeitsgedächtnis, und drittens das Langzeitgedächtnis. Die Inhalte des Langzeitgedächtnisses, das Wissen, werden unterschieden in episodische (Erinnerungen an Begebenheiten) und „semantische“ (Faktenwissen ohne Erinnerung an den Erwerbskontext; die Benennung ist mehr als fragwürdig, da selbstverständlich auch episodisches Material eine Bedeutung hat); die im vorigen Abschnitt 2.2.2 dargelegten Inhalte (Begriffe, bereichs- und sprachspezifisches Wissen, allgemeines Weltwissen, etc.) werden dem semantischen Gedächtnis zugerechnet.

Kurzzeitiges Behalten

Der von Sperling [191] entdeckte visuelle Puffer (*sensory register*; Dauer bis ca. 0,5s) ist nicht mit einem Nachbild zu verwechseln: Ein Nachbild entsteht, wenn durch längeres starres Fixieren des Blicks die Sensoren auf der Netzhaut des Auges erschöpfen; der Eindruck ist nach Helligkeit und Farbe komplementär (starren Sie 20s auf ein rotes Quadrat, und danach auf eine weiße Wand: Sie sehen ein grünes Quadrat). Sensorische Pufferspeicher stellen also kein peripheres Phänomen dar, sondern eher ein „Nachklingen“ im Wahrnehmungssystem, das durch neue Reize sofort gelöscht bzw. überschrieben wird. Das dort Gespeicherte ist nicht in irgendeiner Weise weiter verarbeitet; im Wesentlichen dienen diese Pufferspeicher der selektiven Übernahme von Information ins Arbeitsgedächtnis.

Das Arbeitsgedächtnis (*working memory*, manchmal auch: *short-term memory*) wurde früher als Kurzzeitgedächtnis bezeichnet (*short-term memory*) und durch das Nachsprechen einer Reihe vorgegebener Zahlen (aus dem Bereich 1 bis 10) operationalisiert. Die durchschnittliche Spanne bei Erwachsenen liegt bei der „magischen Zahl 7 ± 2 “ [136], ist also recht beschränkt. Seit Baddeley [9] unterteilt man das Arbeitsgedächtnis meist in drei unterschiedliche Subsysteme, nämlich ein übergeordnetes System (*central executive*, CE), welches die Koordination der kognitiven Prozesse und insbesondere der Aufmerksamkeit gewährleistet und auch die Kommunikation mit dem Langzeitgedächtnis vornimmt, sowie zwei untergeordnete, modalitätsspezifische Subsysteme, nämlich die phonologische Schleife (*phonological loop*, PL) und den visuell-räumlichen Speicher (*visuo-spatial sketchpad*, VSSP).

Die PL entspricht weitgehend dem durch Zahlennachsprechen gemessenen Kurzzeitgedächtnis. Der Erhalt des darin gespeicherten akustisch-verbalen Materials geschieht durch einen Rotationsprozess, das sog. *rehearsal*: stummes (inneres) Aufsagen der Inhalte der PL. Die messbare Kapazität der PL hängt von der Zeit ab, die das *rehearsal* benötigt: Sprecher des Deutschen oder Englischen, wo die Zahlwörter für die ersten zehn natürlichen Zahlen meist einsilbig sind, haben eine größere Spanne als Sprecher z.B. des Walisischen mit durchschnittlich drei Silben pro Zahlwort (sog. Wortlängeneffekt). Die Kapazität der PL kann allerdings durch *chunking* gesteigert werden. Darunter versteht man einen Prozess, der – etwa beim Zahlen-Nachsprechen – mehrere aufeinanderfolgende Zahlen zu einer Einheit integriert. So besteht die Zahlenreihe „1-1-8-0-3-2-0-1-2“ aus 8 Einheiten, rekodiert als „18 03 20 12“ nur noch aus 4 Einheiten, rekodiert als Datum nur aus einer, sofern man es als „Tag, an dem Joachim Gauck zum Bundespräsidenten gewählt wurde“ interpretiert.

denten gewählt wurde“ aus dem Langzeitgedächtnis abrufen kann. Mit aufwändigem Training in Techniken des Rekodierens ist es in Einzelfällen gelungen, die scheinbare Kapazität des PL auf mehr als 80 Ziffern zu steigern [43].

Der „Skizzenblock“ (VSSP) ist ein sowohl bildhaftes, als auch räumliches Arbeitsgedächtnis und kann experimentell (z.B. durch Kombination mit einer abstrakt-verbalen vs. räumlich-verbalen Zweitaufgabe; [13]) von der PL unterschieden werden. Die Leistungsfähigkeit des VSSP wird z.B. durch Tracking-Aufgaben gemessen, bei denen es gilt, einen über einen Bildschirm wandernden Punkt mit der Maus zu verfolgen.

Der CE realisiert weitgehend die bereits in Abschnitt 2.2.1 beschriebenen exekutiven Funktionen und koordiniert neben der nach außen gerichteten auch die „innere“ Aufmerksamkeit, d.h. die Selektion von Inhalten sowohl der beiden Subsysteme des Arbeitsgedächtnisses, als auch des Langzeitgedächtnisses. Zweifellos schlagen sich interindividuelle Unterschiede in der Effizienz des CE auch in Intelligenztests nieder, aber es hat sich als schwierig erwiesen, die individuelle Qualität des CE direkt zu messen; am besten scheint es durch die Generierung von Zufallsfolgen (z.B. eine Taste in zufällig verteilten Zeitintervallen drücken) zu gelingen; je geringer die verfügbare Kapazität des CE ist, desto regelmäßiger werden die generierten Intervalle [12]. Den CE hat Baddeley [10] noch durch eine Speicherkomponente, den *episodic buffer*, ergänzt, der auch den Zugriff auf das Langzeitgedächtnis regelt; eine ähnliche Komponente hatten bereits Ericsson und Kintsch (1995) als *long-term working memory* vorgestellt. Anlass dazu gab der Befund, dass für zusammenhängendes Wortmaterial, z.B. beim Diktieren eines Briefes, eine durchschnittliche Kapazität von etwa 16 Wörtern messbar ist, was die Kapazität der PL mit ca. 7 Einheiten klar übersteigt. Wenn es allerdings darum geht (z.B. beim Kopfrechnen), Teilziele und Zwischenergebnisse zu speichern, sinkt die messbare Kapazität auf etwa 4 Einheiten ab. Diese extrem knappe kognitive Ressource ist auch dafür verantwortlich, dass wir nicht zu systematischer Suche mit Backtracking in der Lage sind (vgl. Abschnitt 2.2.4).

Langfristiges Behalten und Wissenserwerb

Jeder kennt den mühsamen Weg des Lernens von Vokabeln, Texten und vor allem Fakten. Die ersten präzisen Untersuchungen stammen von Hermann Ebbinghaus [39], der in langwierigen Versuchen, sich Listen künstlicher Vokabeln („sinnlose Silben“) einzuprägen, das Grundgesetz des Lernens fand: schneller Lernfortschritt am Anfang, dann allmählich geringerer Lernzuwachs bei gleichem Aufwand – und umgekehrt: Schon nach einer halben Stunde wird ungefähr die Hälfte des Gelernten vergessen, was hingegen nach einer Woche noch erinnert werden kann, wird danach kaum noch vergessen. Bahrick [14] konnte zeigen, dass Vokabeln, die in der Schule gelernt, aber danach nie mehr gebraucht wurden, fünfzig Jahre später noch zu fast einem Drittel erinnert werden konnten. Zudem wirkt jeder Abruf aus dem Gedächtnis wie ein neuer Lerndurchgang; die entsprechenden Gedächtnisinhalte werden sozusagen aufgefrischt (und daher nie gebrauchte Inhalte mit höherer Wahrscheinlichkeit vergessen). Und noch eine wichtige Eigenschaft des Langzeitgedächtnisses gilt es zu nennen: Noch niemand hat jemals die Kapazitätsgrenzen messen können – unendlich viel scheint in unserem Langzeitgedächtnis Platz zu haben. Die Arbeitsweise neuronaler, verteilter Speicherung (s.o.) bringt es mit sich, dass in jeder „Antwort“ unseres Gedächtnisses auf eine Suchanfrage neben dem Gesuchten auch Elemente ähnlicher Gedächtnisinhalte beigemischt sind. Dadurch kann es passieren, dass der gesuchte Inhalt durch einen ähnlichen Inhalt verfälscht oder überdeckt wird. Dieses Phänomen der Interferenz ähnlicher Gedächtnisinhalte ist bereits von Müller und Pilzecker [141] beschrieben worden.

Der Weg ins Langzeitgedächtnis führt über das Arbeitsgedächtnis. Dreijährige Kinder erwerben als erste gedächtnisbezogene Strategie: Du musst hinschauen (also Aufmerksamkeit investieren), wenn du dir etwas merken willst. Je ausgiebiger und „tiefer“ Inhalte bearbeitet werden, desto höher ist die Wahrscheinlichkeit, sie später erinnern zu können. Beim absichtsvollen Einprägen (sog. intentionales Lernen) kommtförderlich hinzu, dass wir das Material organisieren (z.B. kategorisieren) und Bezüge herstellen, die als sog. *retrieval plan* beim Wiedererinnern helfen, indem sie Hinweisreize (*cues*) generieren. – Neurobiologisch ist vor allem der Hippocampus für die dauerhafte Speicherung verantwortlich. Der vielzitierte Patient H.M., dem zur Verhinderung weiterer epileptischer Anfälle der Hippocampus beidseitig entfernt wurde, konnte fast gar nichts Neues mehr lernen. Für die *retrieval plans* hingegen sind Bereiche des Frontalhirns verantwortlich; geschädigte Patienten können nichts bewusst erinnern, haben aber noch ein sog. implizites Gedächtnis: Das Material ist gespeichert, kann aber nicht absichtlich abgerufen werden. Implizites Gedächtnis entsteht auch dann, wenn Lernen nicht absichtsvoll (intentional), sondern nebenbei geschieht (inzidentell). So gibt es Belege dafür, dass unabsichtlich und unbewusst auch die Lernumgebung gespeichert wird und Gelerntes in gleicher Umgebung, die offenbar einen hilfreichen Hinweisreiz darstellt, besser reproduziert wird als in anderer Umgebung (z.B. [70]).

Generell ist das Wiedererkennen (*recognition*) leichter als das aktive Wiedererinnern (Reproduktion, *recall*). Das übliche Experiment besteht darin, dass nach Darbietung einer Reihe von Wörtern ein Teil dieser Wörter, zusammen mit neuen, vorher nicht dargebotenen (sog. Distraktoren) vorgegeben wird. Die Aufgabe besteht darin, die Wörter als „alt“ bzw. „neu“ zu erkennen. Hier müssen keine Hinweisreize generiert werden, denn die (alten) Wörter sind selbst ihr bester Hinweisreiz.

Gegenüber dem Lernen unverbundenen Materials (z.B. einer Liste beliebiger Wörter) müssen beim Lernen aus Texten Wissensbestände organisiert werden, die später beim Erinnern hilfreich sind. Der Text muss verstanden werden (s. Abschnitt 2.2.6), was die Bildung von kausalen Inferenzen voraussetzt, am besten von sog. Selbsterklärungen [26]. Zu speichern ist nicht der Text, sondern die aufgrund des Textes konstruierte Wissensstruktur – eine anspruchsvolle Aufgabe, die durch einschlägige Vorkenntnisse erleichtert wird.

Assoziatives Lernen und Antizipation

In der Zeit des psychologischen Behaviorismus (ca. 1920–1960) dominierte die Ansicht, dass alles Lernen auf den Erwerb von Reiz-Reaktionsverbindungen zurückzuführen sei. Erst die Theorie von Rescorla und Wagner [171] fasste Konditionierungsprozesse als Lernen durch Antizipation auf. Das sog. klassische Konditionieren wurde 1901 von Pawlow im Rahmen seiner Untersuchungen zur Verdauung entdeckt. Es basiert auf der Antizipation eines „biologisch unbedingten Reizes“ (hier positiv: Futter für die Versuchshunde) aufgrund der Wahrnehmung eines neutralen oder „bedingten“ Reizes (*conditioned stimulus*, CS) mit dem „unbedingten“ (*unconditioned stimulus*, UCS) entsteht eine „bedingte Reaktion“ (*conditioned response*, CR), die sich im Fall von Pawlows Hunden nur quantitativ von der Reaktion auf den unbedingten Reiz (UCR) unterscheidet: Der Hund speichelt, aber nicht so stark wie auf das lecker duftende Futter selbst. In anderen Fällen (bekannt der Fall des kleinen Albert: [219]) ist die CR auch qualitativ von der UCR unterschieden (im Beispiel: Vermeidung statt Erschrecken).

Das sog. operante Konditionieren benutzt eine klar positive (z.B. Futter, Lob) oder aversive (z.B. elektrischer Schlag) Verhaltenskonsequenz als biologisch unmittelbaren Reiz, dem hier

aber nicht ein anderer, neutraler Reiz vorangeht, sondern ein bestimmtes Verhalten, etwa: eine Taube pickt auf eine Pickscheibe und erhält als Konsequenz ein Futterkorn nach jedem fünften Picken. Skinner [188] hat diese Technik in vielen Varianten entwickelt und u.a. auf die Tierdresur angewendet. Dabei verstärken positive Reize (also z.B. Futter für ein hungriges Tier) das Verhalten, während aversive Reize (Strafreize) nicht etwa das Verhalten abschwächen, sondern vielmehr ein Vermeidungsverhalten hervorrufen und verstärken. Mit Hilfe des operanten Konditionierens kann man ein Verhalten unter Reizkontrolle bringen, beispielsweise eine Taube in der Skinner-Box dazu bringen, nur bei grünem Licht auf die Pickscheibe zu picken, nicht aber bei gelbem. Durch dieses sog. Diskriminationslernen kann man auch die Wahrnehmungsleistungen von Tieren messen (hier: die Farbdiskrimination von Tauben).

Die Theorie des Konditionierens [171] stellt – wie schon das ältere Modell von Bush und Mosteller [23] – klar, dass auch hier gilt: Der Lernzuwachs pro Lerndurchgang ist anfangs hoch und erreicht nach hinreichend häufiger Übung eine Asymptote. Der Lernzuwachs ist aber nach dieser Theorie nur abhängig von zwei Faktoren, nämlich erstens der (aufmerksamkeits-abhängigen) Salienz (*saliency*), also Auffälligkeit der Reize, und zweitens von der „Überraschung“ (*surprise*) durch den UCS bzw. die Verhaltenskonsequenz, wobei die Überraschung durch das Lernen allmählich gegen Null geht, weil der biologisch wirksame Reiz mental antizipiert wird. Beides, Salienz wie Überraschung (als Gegenstück zur Erwartung), sind kognitive Konzepte, die der ursprüngliche Behaviorismus als unwissenschaftlich gebrandmarkt hätte; indes fügt sich die Antizipation von erwartbaren Folgen gut in die Ausführungen zur Handlungskontrolle im Abschnitt 2.2.1 ein. Erfolgreiches Konditionieren ist nicht an Bewusstheit der Erwartung gebunden; falls man sich aber der Konditionierung bewusst wird, dann kann ihr kognitiv entgegengewirkt werden. Durch die klar positive oder negative Qualität der UCS bzw. der Verhaltensfolgen ergibt sich ferner ein direkter Bezug zur Motiviertheit des Verhaltens, vereinfacht gesagt: Wir tun oft und gerne, was uns Lob und Befriedigung verschafft.

Erwerb sensomotorischer Fertigkeiten

Dinge, die wir *tun* können – vom Fahrradfahren bis zum Kopfrechnen – sind Fertigkeiten (*skills*); man spricht hier auch von prozedurellem Wissen (im Gegensatz zum Faktenwissen, das als deklaratives Wissen bezeichnet wird). Das Lernen beginnt hier oft mit Demonstrationen und verbalen Erklärungen, aber letztlich ist intensives, langes Üben entscheidend. Auf Fitts [53] geht die Unterscheidung dreier Stufen des Fertigkeitserwerbs zurück: Am Anfang steht die mühsame, schrittweise Umsetzung verbaler Anweisungen in einen (meist noch stockenden) Bewegungsablauf, danach vollzieht sich nach einiger Übung eine Integration in eine einzige, fließende Bewegung (ein motorisches Schema); nach Anderson [3] findet hier eine *knowledge compilation* statt. Hierbei kann es vorkommen, dass die deklarative Komponente (d.h. die Erinnerung an die verbalen Anweisungen) vergessen wird, weil nicht mehr gebraucht wird. Weiteres Üben verhilft zum „Feinschliff“, wobei wie bei der Lernkurve für das Vokabellernen immer mehr Übungsaufwand nötig ist, um weitere Lerngewinne zu erzielen. In dieser letzten Phase kann auch Üben in der Vorstellung hilfreich sein. Stets aber braucht es Rückmeldung über den Erfolg (die Genauigkeit, die Geschwindigkeit) der Ausführung: ohne Rückmeldung kein Lernen.

Das eben Ausgeführte erklärt Beispiele wie „im Fahrunterricht lernen, wie man im Auto die Gänge schaltet“ oder „einen neuen Tanz erlernen“. Hingegen greift die in jüngster Zeit wieder populäre Ansicht, dies sei im Wesentlichen Imitation (unterstützt dadurch, dass bei Greifbewegungen von Affen sog. Spiegelneuronen gefunden wurden; [174]) zu kurz. Denn erfolgreich imitieren können wir nur etwas, das bereits in unserem Verhaltensrepertoire vorhanden

ist. Hingegen ist davon auszugehen, dass Spiegelneuronen, die auch beim Menschen vor allem im Hinblick auf Handbewegungen gefunden wurden, sich erst durch frühkindliche Übung entwickeln (Finger bewegen und gleichzeitig betrachten). Vor allem ist die Imitation von Bewegungsmustern, die nicht bereits beherrscht werden, in aller Regel nur oberflächlich ähnlich und nicht funktional wirksam. Daher treten zur Demonstration typischerweise verbale Erklärungen und Anweisungen hinzu, sowie aufmerksamkeitslenkende Hinweise (verbal oder als Zeigehandlung).

2.2.4 Denken und Problemlösen

Schlussfolgerndes Denken

Trotz großer Fortschritte im Bereich der Künstlichen Intelligenz ist es bisher noch nicht gelungen selbstständig denkende Maschinen zu konstruieren. Obwohl menschliches Denken vielfältig ist wurde in der Forschung vor allem das „Schlussfolgern“ [17] untersucht. Die Fähigkeit, aus vorhandenem Wissen durch Schlussfolgerungen neue Einsichten zu gewinnen, gehört zu den grundlegendsten kognitiven Fähigkeiten des Menschen und ist ein zentraler Gegenstand der aktuellen psychologischen Forschung. Menschliches Denken ist nicht einfach die Realisierung der Gesetze klassischer Logik, sondern ist facettenreicher und unterscheidet sich teilweise deutlich. Beispielsweise spielt der Kontext einer Aufgabe eine wesentliche Rolle. Da oftmals der Problemraum zu groß ist, um vollständig im Arbeitgedächtnis repräsentiert zu werden, benutzen wir oftmals Heuristiken. Solche *Heuristiken* sind Strategien, die eine vollständige Problemrepräsentation und -manipulation nicht erfordern und damit das Auffinden einer Lösung erleichtern oder beschleunigen können. Diesen möglichen und aufgabenabhängigen Vorteilen steht ein Nachteil gegenüber: manchmal wird, trotz Existenz einer Lösung, diese nicht oder nur eine suboptimale gefunden.

Schlussfolgerndes Denken lässt sich grob in die Kategorien deduktives, induktives und analoges Schlussfolgern einteilen. Im Folgenden werden wir kurz diese Bereiche betrachten.

Deduktives Denken.

Deduktives Schlussfolgern lässt sich definieren als die Methode, aus einer gegebenen Menge an Informationen (diese werden klassischerweise als Prämissen bezeichnet) eine Schlussfolgerung zu ziehen. Die deduktive Methode geht also vom allgemeinen zum spezifischen Fall und ist unabhängig vom Inhalt.

Gegeben: Eine Menge von Aussagen A_1, \dots, A_n .

Frage: Welche Konklusion K lässt sich daraus ziehen,
d.h. $A_1 \wedge \dots \wedge A_n \rightarrow K$?

Solche Problemstellungen können auch mit einer möglichen Konklusion K definiert sein. Klassischerweise unterscheidet man die Bereiche syllogistisches, konditionales und relationales Schlussfolgern. Dazu später mehr.

Induktives Denken.

Im Gegensatz dazu führt induktives Denken zu einer Verallgemeinerung. Ein typisches Beispiel [80] in Variation ist das folgende:

- (1) Tauben, Adler, Falken, ... sind Vögel und können fliegen.
Also können wir schlussfolgern, dass Vögel fliegen können.

Diese Schlussfolgerung basiert auf einzelnen Beobachtungen, kann also als eine Form von *Beispiellernen* (vgl. Kapitel 12) gelten. Eine etwas formalere Schreibweise ist die folgende:

Gegeben: Zwei Mengen M' , M mit $M' \subset M$ und
die Relation R gilt für alle m aus M' .

Frage: Gilt die Relation R für alle m aus M ?

Dabei steht M' für die beobachteten Vögel und M für die Menge aller Vögel. Das bedeutet für jeden beobachteten Vogel aus M' gilt die Relation „kann fliegen“. Solche Schlussfolgerungen akzeptieren wir allerdings nur solange kein Gegenbeispiel auftritt, uns also beispielsweise einfällt, dass ein Pinguin ein Vogel ist, der nicht fliegen kann. Induktives Schlussfolgern außerhalb der Mathematik ist insgesamt nur empirisch begründet und damit unsicher. Oder anders ausgedrückt: Induktive Schlüsse gelten also nicht mit Sicherheit.

Abduktives Denken.

Abduktives Schlussfolgern [37] lässt sich beschreiben als das Auffinden einer besten Erklärung E_i für Erklärungen E_1, \dots, E_n für einen Fakt F . Eine der bekanntesten Beispiele in der Literatur ist die Arbeitsweise von Sherlock Holmes, der ausgehend von einem Fakt, sukzessive Erklärungsmuster ausschließt.

Gegeben: Erklärungen E_1, \dots, E_n für einen Fakt F , d.h. $E_1, \dots, E_n \rightarrow F$

Frage: Gibt es eine beste Erklärung E_i , d.h. $E_i \rightarrow F$?

Analoges Denken.

Analoges Denken wird oftmals zur Kategorie des induktiven Denkens gerechnet [17]. Es überträgt strukturelles oder inhaltliches Wissen einer Quelldomäne (*source*) auf eine Zieldomäne (*target*). Damit verbindet analoges Denken Wissen verschiedener Domänen. Zwei Beispiele für analoge Aufgaben [170]:

- | | | | | | | | | |
|------|----------|--------|-------|-----|-----------------|----|----|------|
| (i) | Hammer | ist zu | Nagel | wie | Schraubenzieher | zu | ? | |
| (ii) | Portrait | ist zu | Bild | wie | | ? | zu | Buch |

Die Aufgabe ist es einen Begriff für ? zu finden, so dass die analoge Relation zu den vorangehenden Begriffen erhalten bleibt. Ein Hammer kann benutzt werden, um einen Nagel in die Wand zu schlagen. Wofür kann dann ein Schraubenzieher analog benutzt werden? Um eine Schraube fest zu drehen. Die Quell- als auch Zieldomäne ist in beiden Fällen die Domäne der Werkzeuge. Das zweite Beispiel erfordert das Finden einer Relation, in der ein Buch stehen kann, welches analog zum Portrait und Bild ist. Ein Buch also, welches über eine Person geschrieben ist – eine Biographie. Eine mögliche formale Definition eines analogen Problems lautet wie folgt:

Gegeben: Zwei Domänen D_1 und D_2 ; in D_1 gilt die Relation R zwischen zwei Elementen E_1 und E_2 aus D_1 .

Frage: Gibt es eine Funktion f , so dass für die Elemente $f(E_1), f(E_2)$ aus D_2 : $R(E_1, E_2) \in D_1 \Leftrightarrow R(f(E_1), f(E_2)) \in D_2$ gilt?

Eine generalisierte Fassung dieser Problemdefinition wäre, dass nur die Zieldomäne D_2 gegeben ist, aber zusätzlich zur Funktion f auch eine Quelldomäne D_1 gesucht wird. Eine zentrale Unterscheidung bei Analogien ist zwischen Objekt und Relation gegeben. Nach [66, 125] gibt es zwei wesentliche Eigenschaften von Analogien: Die *strukturelle Konsistenz* besagt, dass der relationale Zusammenhang zwischen Elementen der Domäne erhalten bleiben muss. Jedes Element der einen Repräsentation ist damit im Sinne einer eindeutigen (*injektiven*) Abbildung auf höchstens ein Element der anderen Domäne bezogen. Die zweite Eigenschaft bezieht sich auf den *relationalen Fokus*: Analogien erhalten die relationalen Eigenschaften.

Eine der bekanntesten Theorien ist die von Gentner eingeführte Structure-Mapping Engine (SME). Sie schlagen einen dreistufigen Algorithmus [46, 64, 65] zur Beschreibung menschlicher Analogiebildung vor:

- *Zugang*: Identifizierte für eine Zieldomäne aus dem Langzeitgedächtnis eine Quelldomäne, die ähnlich (oder sogar analog) zur Zieldomäne ist.
- *Abbildung und Inferenz*: Identifizierte einzelne Elemente in beiden Domänen, die in einer Relation stehen, und generalisiere diese zu einer möglichst allgemeinen Abbildung zwischen den Domänen. Bilde erste potentielle Inferenzen.
- *Evaluation und Anwendung*: Evaluiere die generalisierten Abbildungen. Die Abbildung, die am besten passt, ist die gesuchte Analogie.

Im Folgenden konzentrieren wir uns auf das deduktive Schlussfolgern, welches neben einer großen Alltagsrelevanz auch den am besten untersuchten Bereich darstellt.

Beispiele für deduktives Schlussfolgern.

Deduktives Schließen wird klassischerweise in drei Gebiete eingeteilt: In syllogistisches Schließen (Schlussfolgern über quantifizierte Aussagen), konditionales Schließen (konditionale Aussagen haben die Form wenn ... dann) und relationales Schließen. Zunächst ein Beispiel für syllogistisches Schließen:

- (2) Alle Astronauten sind Biertrinker.
Manche Biertrinker sind Choleriker.

Welche Schlussfolgerung kann aus dieser syllogistischen Aufgabe gezogen werden? Die meisten Versuchspersonen antworten, dass „manche Astronauten Choleriker“ sind [22, 149]. Das ist zwar durchaus eine Möglichkeit, aber keinesfalls zwingend. Denn es ist logisch korrekt zu folgern, dass alle möglichen Quantoren in einer Konklusion über Astronauten und Choleriker gelten können. Wie bereits erwähnt, werden bestimmte Konklusionen (bewusst oder unbewusst) präferiert. Diesen Aspekt werden wir später wieder aufgreifen. Betrachten wir jetzt ein Beispiel für konditionales Schließen:

- (3) Wenn ich lerne, dann bestehe ich die Prüfung.

Ein Konditional besteht aus einem Bedingungsteil (*Antezedenz*) „ich lerne“ und einer Konsequenz „ich besteh die Prüfung“. Ist nun eine Bedingung gegeben, dann kann ich die Schlussfolgerung (die Konsequenz) ziehen. Diese Schlussfigur heißt Modus Ponens. Die dritte Hauptkategorie ist das sogenannte relationale Schließen:

- | | |
|-------------------------------------|-------------------------------------|
| (D) Der Ferrari parkt links vom VW. | (I) Der Ferrari parkt links vom VW. |
| Der VW parkt links vom Audi. | Der VW parkt links von Audi. |
| Der Audi parkt links von BMW. | Der VW parkt links von BMW. |
| Der BMW parkt links vom Opel. | Der BMW parkt links von Opel. |

Dabei ist das linke Problem (D) ein sogenanntes determiniertes Problem, d.h. es gibt genau eine mögliche Anordnungen der Autos. Das rechte Problem (I) ist ein sogenanntes indeterminiertes Problem ist, d.h. es sind mehrere Anordnungen möglich. Im Gegensatz zu den klassischen informatischen Verfahren sind menschliche Schlussfolgerungen nicht immer korrekt. Auch ist die Frage offen, ob die Mechanismen vollständig sind, d.h. ob alles, was ableitbar ist, sich auch mittels der entsprechenden Methoden ableiten lässt. Kognitive Theorien menschlichen Schlussfolgerns lassen sich in die folgenden vier Hauptbereiche einteilen:

1. *Regelbasierte Theorien*. Diese Theorien werden auch Theorien mentaler Logik genannt. Sie stellen syntaktische Verfahren in den Vordergrund, d.h. es werden Inferenzregeln auf die Menge von Aussagen angewandt [20, 173]. Beispielsweise wird auf das obige Beispiel 3 die Regel Modus Ponens angewandt ($A \rightarrow B$ und A gegeben, dann gilt B). Der Modus Tollens ($A \rightarrow B$ und $\neg B$ gegeben, dann gilt $\neg A$) ist schwieriger für uns, weil dafür eigentlich ein mentaler Beweis notwendig ist. Für die relationale Aufgabe (D) werden Transitivitätsregeln angewandt.
2. *Modellbasierte Theorien*. Diese Theorien werden auch als ein semantisches Verfahren klassifiziert. Die bekannteste Theorie dazu ist die Theorie mentaler Modelle [94, 95]. Gemäß dieser Theorie konstruieren wir sogenannte mentale Modelle für eine gegebene Menge von Aussagen. Betrachten wir das Beispiel für relationales Schließen oben: Nach dieser Theorie konstruieren wir ein mentales Modell, welches einer räumlichen Anordnung entspricht:

Ferrari VW Audi BMW Opel

Falls mehrere Modelle konsistent mit den Prämissen sind, dann sind diese Aufgaben schwerer. Dieser Effekt wird als *Indeterminiertheiteffekt* bezeichnet (vgl. Aufgabe (I) im Gegensatz zu (D)) und durch die zusätzliche Zahl an konstruierbaren Modellen erklärt. Menschen präferieren dabei bestimmte Modelle [166]. In diese Kategorie fallen auch Ansätze von Stenning und Lambalgen [193], welche neben den klassischen Wahrheitswerten *wahr* und *falsch* auch noch ein *unbekannt* hinzunehmen. Sie benutzten Logiken mit drei Wahrheitswerten, um beispielsweise die Suppressionsaufgabe zu beschreiben (siehe unten).

3. *Probabilistische Theorien*. Diese Theorie geht davon aus, dass wir zum Schlussfolgern die Bayessche Regel benutzen. Dabei wird für syllogistisches Schließen die Formel durch zusätzliche Heuristiken wie der Informativitätsheuristik ergänzt [149]. Diese Heuristiken wurden mittels Metastudien identifiziert [148]. Offen ist die Frage nach der Kenntnis von Basiswahrscheinlichkeiten für Ereignisse (insbesondere für singuläre Ereignisse). Außerdem existiert noch keine wahrscheinlichkeitsbasierte Theorie für relationales Schließen (Aufgaben (D) und (I)).

Ein Kritikpunkt an der mentalen Modelltheorie ist, dass vorwiegend absolute Aussagen („A ist B“) repräsentiert werden und keine Wahrscheinlichkeiten („wahrscheinlich ist A ein B“). Allerdings gibt es auch Ansätze, die Modelltheorie, um wahrscheinlichkeitsbasierte Ansätze zu erweitern [96]. Zusätzlich existieren auch einige *domänenspezifische Theorien* [45]. Ein weiterer Ansatz sollte noch erwähnt werden: Gemäß Vertretern sogenannter *Dualer Theorien* [101] ist unser Denken nicht monolithisch. Vielmehr lässt sich menschliches Schlussfolgern in mindestens zwei Phasen einteilen. In der ersten Phase, die auch manchmal als System I bezeichnet wird, schlussfolgern wir schnell und heuristisch (*fast and frugal* [68]). In einer anschließenden Phase, die nicht immer eintreten muss, schlussfolgern wir dann analytisch und korrekt. Die mentale Modelltheorie kann auch zu einer solchen dualen Theorie gerechnet werden (initiales Modell versus Modellvariationsphase).

Schließen mit Heuristiken.

Neben klassischen Inferenzen haben wir Menschen auch Probleme mit Wahrscheinlichkeit umzugehen. Ein bekanntes Denkproblem ist das Linda-Beispiel [51, 213]:

- (4) Linda ist 31 Jahre alt, alleinstehend, nimmt kein Blatt vor den Mund und ist sehr intelligent. Sie hat Philosophie studiert. Als Studentin hat sie sich stark gegen Diskriminierung und für soziale Gerechtigkeit engagiert. Außerdem hat sie an Anti-Atomkraft-Demonstrationen teilgenommen.

Was ist wahrscheinlicher?

A Linda arbeitet als Bankberaterin.

B Linda arbeitet als Bankberaterin und ist in einer feministischen Bewegung aktiv.

Die Aussage B ist zusammengesetzt aus der Aussage A „Linda arbeitet als Bankberaterin“ und der Aussage C „ist in einer feministischen Bewegung aktiv“. Die Wahrscheinlichkeit des gemeinsamen Ereignisses $P(B)$ ist gleich die Wahrscheinlichkeit $P(A \wedge C)$. Diese ist natürlich höchstens so groß wie die Einzelwahrscheinlichkeiten $P(A)$, kurz: $P(A \wedge C) \leq P(A)$. Dennoch neigt die Überzahl der Befragten (85%) dazu Antwort B zu wählen und damit $P(A \wedge C)$ als wahrscheinlicher einzuschätzen [213]. Das bedeutet zunächst, dass Menschen offensichtlich nicht das Äquivalent zu dem mathematischen Wahrscheinlichkeitskalkül nutzen. Es kann aber auch bedeuten, dass wir einem Informationsprinzip folgen und die beiden Informationen integrieren wollen oder dass wir auf das Framing achten [89]. Auf jeden Fall zeigt es, dass wir den „Kontext“ auch hier miteinbeziehen.

Dieses Beispiel zeigt, dass wir Schwierigkeiten haben, die Konjunktion zweier Wahrscheinlichkeiten richtig zu interpretieren (nämlich, dass die Konjunktion zweier Ereignisse maximal der Wahrscheinlichkeit einer der Ereignisse entsprechen kann).

Selbstverständlich wären auch Kombinationen im Sinne hybrider Theorien denkbar. Dabei gilt hier, wie auch an anderer Stelle im Sinne von Occams Rasermesser, dass eine einfachere Theorie einer hybriden Theorie bevorzugt wird. Diese Verfahren demonstrieren bisher im Sinne eines *Proof of Concept*, dass die verschiedenen Theorien in der Lage sind, zumindest manche

menschlichen Schlüsse vorherzusagen. Sie treffen keine Aussagen über die kognitive Adäquatheit [197]. Insgesamt lässt sich der kognitive Adäquatheitsbegriff unterteilen in (i) *repräsentationale Adäquatheit*, d.h. benutzen Menschen ähnliche Repräsentationen wie die kognitionspsychologische Theorie es vorhersagt, und (ii) *inferentielle Adäquatheit*, d.h. ziehen Menschen die gleichen Schlussfolgerungen wie die kognitionspsychologische Theorie es vorhersagt. Dabei ist zu beachten, dass die Theorien wenig Annahmen über das menschliche Arbeitsgedächtnis integrieren. So spielen Beschränkungen des menschlichen Arbeitsgedächtnisses und spezifische Verarbeitungen verschiedener modalitätsspezifischer Informationen bisher keine Rolle. Eine Möglichkeit zur Verbindung und damit der Überprüfung der kognitiven Adäquatheit wäre die Implementation dieser Deduktionstheorien in einer kognitiven Architektur (s.o.). Das wurde bisher allerdings nur vereinzelt unternommen.

Ein Beispiel für eine Verbindung von schlussfolgerndem Denken und Gedächtnisrepräsentation stellt dabei die sogenannte Verfügbarkeitsheuristik dar [102]:

- (5) Gibt es mehr Worte im Englischen die ein „r“ an erster oder an dritter Stelle haben?

Die meisten antworten: „Es gibt mehr Worte mit einem ‚r‘ an erster Stelle“. Dennoch gibt es tatsächlich mehr als dreimal soviele englische Worte mit einem „r“ an dritter Stelle. Warum begehen Menschen konsequent diesen Denkfehler? Der Grund ist, dass der Anfangsbuchstabe eine deutlich relevantere Rolle in unserer Gedächtnisrepräsentation spielt. So scheint eine lexikalische Anordnung eine große Rolle zu spielen. Damit ist also letztlich ein „Speicherverwaltungsprinzip“ die Ursache für diesen Fehlschluss.

Kontextabhängigkeit menschlichen Denkens.

Menschliches Denken und Schlussfolgern hängt vom jeweiligen Kontext bzw. dem Hintergrundwissen über eine entsprechende Domäne ab. Die Domäne entscheidet, ob und wie leicht oder schwer bestimmte Schlussfolgerungen gezogen bzw. nicht gezogen werden. Betrachten wir dazu die sogenannte Wason Selection Task [218] näher:

Es liegen vier Karten auf einem Tisch. Jede Karte hat eine Zahl auf einer Seite und einen Buchstaben auf der anderen Seite. Die sichtbaren Seiten der Karten zeigen 2, 7, A und D. Welche Karte(n) müssen umgedreht werden, um die Gültigkeit der folgenden Aussage zu überprüfen: „Wenn auf einer Kartenseite eine gerade Zahl ist, dann ist auf der Rückseite ein Vokal.“

Korrekt erweise müssen die Karten „2“ und „D“ umgedreht werden. Die „2“ testet die Regel, denn wenn auf der Rückseite kein Vokal ist, dann ist die Regel falsifiziert, das „D“ kann aus dem gleichen Grund die Regel falsifizieren, nämlich wenn auf der Vorderseite eine gerade Zahl ist. Die meisten Versuchspersonen drehen allerdings richtigerweise die „2“ und fälschlicherweise das „A“ um. Eine Vermutung war, dass die Domäne abstrakt ist und es am mangelnden Hintergrundwissen lag, warum Versuchspersonen diesen Fehler begehen. Aus diesem Grund haben Johnson-Laird und Kollegen [92] den Einfluss von Hintergrundwissen untersucht und konnten zeigen, dass zum Beispiel eine analog formulierte Aufgabe mit Briefen und Briefmarken deutlich korrekter gelöst wurde. Schließlich konnten Cosmides und Tooby [31] nachweisen, dass diese Fehler aber nicht im Kontext von sozialen Regeln (*cheater detection*) auftraten. Dazu testeten sie die Regel „Wer Alkohol trinkt, muss über 18 Jahre alt sein“.

Es liegen vier Karten auf einem Tisch. Jede Karte hat ein Alter auf einer Seite und ein Getränk auf der anderen Seite. Die sichtbaren Seiten der Karten zeigen „16“, „trinkt Bier“, „25“, „trinkt Cola“.

Wenige Versuchspersonen haben bei dieser Aufgabe Schwierigkeiten die richtigen Karten auszuwählen, nämlich „16“ und „Bier“. Korrekte Schlussfolgerungen konnten in einer Vielzahl von Experimenten für solche „soziale Regeln“ nachgewiesen werden. Da also der Kontext entscheidend ist, zeigt dieses Resultat eindeutig, dass wir nicht „einfach“ Regeln unabhängig von der Domäne anwenden. Menschliches Denken ist also kontextabhängig. Dieses Resultat stellt dabei Vertreter eines allgemeinen regelbasierten Ansatzes vor Probleme, denn die zugrunde liegende Regel (der sogenannte Modus Tollens) ist bei beiden Aufgaben identisch. Menschliches Denken ist nicht nur kontextabhängig, sondern kann darüber hinaus auch leicht nichtmonotone Charakteristiken aufweisen. Ein Beispiel dafür ist die sogenannte *Suppression task* [24].

- (6) Wenn sie einen Aufsatz fertigstellen muss,
dann wird sie bis spät in der Bibliothek arbeiten.
Sie hat einen Aufsatz zu schreiben.
- (7) Wenn sie einen Aufsatz fertigstellen muss,
dann wird sie bis spät in der Bibliothek arbeiten.
Wenn sie ein Lehrbuch lesen muss,
dann wird sie bis spät in der Bibliothek arbeiten.
Sie hat einen Aufsatz zu schreiben.
- (8) Wenn sie einen Aufsatz fertigstellen muss,
dann wird sie bis spät in der Bibliothek arbeiten.
Wenn die Bibliothek lange offen ist,
dann wird sie bis spät in der Bibliothek arbeiten.
Sie hat einen Aufsatz zu schreiben.

Im Fall von Aufgabe 6 und 7 ziehen jeweils 96% der Versuchspersonen die korrekt Schlussfolgerung, dass „sie bis spät in der Bibliothek arbeiten wird“. Im Fall von Aufgabe 8 allerdings nur 38%. Obwohl der Schluß (Modus Ponens, s.o.) korrekt ist, wird er in Beispiel 8 nicht mehr gezogen. Die zusätzliche Information hat die Anwendung der Inferenz *unterdrückt*. Tatsächlich ist dies ein Indiz darauf, dass menschliches Denken auch nichtmonoton sein kann [193].

Die angeführten Beispiele zeigen, dass sich die „menschliche Logik“ doch in wesentlichen Punkten von formaler Logik unterscheidet. Ein Bereich, in dem das noch stärker gilt, ist menschliches Problemlösen.

Problemlösendes Denken

Ein Problem im Sinne der Kognitionswissenschaft liegt vor, falls ein Anfangszustand gegeben ist, ein Zielzustand erreicht werden soll und der Weg vom Ausgangszustand zum Ziel nicht offensichtlich ist (Duncker [38] spricht von einer „Barriere“). Die anwendbaren Operatoren zur Lösung können gegeben sein, dann spricht man von wohldefinierten Problemen (*well-defined problems*) oder die Operatoren sind nicht gegeben (*ill-posed problems*). Das übliche Verfahren der *Künstlichen Intelligenz*, nämlich extensive Suche, scheitert beim Menschen oftmals

aufgrund von Arbeitsgedächtnisbeschränkungen. Stattdessen wird auf Inhalte des Langzeitgedächtnisses zugegriffen und die Methode des fallbasierten Schließens (*Case-based reasoning*) benutzt. Die Klassen unterschiedlicher Probleme lassen sich noch weiter unterteilen:

Permutationsprobleme.

Erste Ansätze der Kognitionswissenschaft beschrieben analog zur Künstlichen Intelligenz Planungsprobleme als Suchprobleme [145]. Diese Charakterisierung passt für sogenannte *Permutationsprobleme*. Das bekannteste Beispiel stellt hierbei *Kannibalen und Missionare* dar.

Es stehen drei Kannibalen und drei Missionare am Ufer eines Flusses, den sie überqueren müssen. An ihrem Ufer steht ein einziges Boot, in das maximal zwei Personen passen. Alle sechs Reisenden müssen übersetzen. Falls zu einem Zeitpunkt mehr Kannibalen als Missionare an einem Ufer sind, dann werden die Kannibalen die Missionare verspeisen. Gibt es einen Plan, so dass alle Reisenden heil den Fluss überqueren können?

Probleme solcher Art lassen sich mittels Suchverfahren lösen. Dabei kann der gesamte Suchraum vollständig repräsentiert werden. Eine der bekanntesten Heuristiken ist die Mittel-Ziel-Analyse (*Means-Ends Analysis*) [145], die besagt, dass wir ein gegebenes Problem in Teilprobleme zerlegen (die *ends*), mögliche Operatoren identifizieren (die *means*) und dann die Operatoren auswählen, die uns dem Ziel am nächsten bringen. Notwendig ist dafür eine Bewertungsfunktion, die für einen gegebenen Zustand den Abstand zum Ziel evaluiert. Wenn wir von der Zielreduktionsheuristik abweichen müssen (zum Beispiel im obigen Kannibalen und Missionare Szenario), dann fällt uns das schwerer. Diese letztere Heuristik wird als Zielreduktion (*difference reduction*) bezeichnet. Dabei werden in der Künstlichen Intelligenz teilweise verschiedene Namen für ähnliche Heuristiken benutzt. Zum Beispiel wird die Greedy-Strategie mit der Zielreduktion assoziiert. Die einzige Schwierigkeit, die bei Menschen auftreten kann, ist, dass wir eine Greedy-Strategie anwenden und Operationen, die uns wieder vom Ziel entfernen, vermeiden. Allerdings zeigen Ergebnisse, dass wir nicht den gesamten Suchraum durchsuchen, sondern systematisch bestimmte Operationen bevorzugen bzw. heuristisch suchen.

Einsichtsprobleme.

Eine weitere bedeutsame Art von Planungsproblemen sind Einsichtsprobleme (*insight problems*). Diese Probleme sind ursprünglich inspiriert aus der Gestaltpsychologie und lassen sich nicht durch eine erschöpfende Suche lösen. Ein berühmtes Beispiel wurde von Duncker [38] vorgeschlagen, das „Kerzenproblem“:

„An der Tür, in Augenhöhe, sollen nebeneinander drei kleine Kerzen angebracht werden [...] Auf dem Tisch liegen unter vielen anderen Gegenständen einige Reißnägel und die kritischen Gegenstände: drei kleine Pappschachteln (ungefähr von der Größe einer gewöhnlichen Streichholzschachtel, etwas verschieden voneinander in Form und Farbe und an verschiedenen Stellen gelegen). Lösung: Mit je einem Reißnagel werden die drei Schachteln an der Tür befestigt, um je einer Kerze als Standfläche zu dienen.“ [38] (zitiert nach [62])

Eine Erklärung, warum diese Lösung häufig übersehen wird, ist die sogenannte *funktionale Fixierung*. Das bedeutet wir betrachten die Streichholzschachtel als etwas, was benutzt wird, um Streichhölzer aufzubewahren, aber nicht um ein Tropfschutz oder eine Befestigung für die

Kerze zu sein. Einsichtsprobleme lassen sich nicht durch Suchalgorithmen lösen [29]. Sie setzen nahezu immer eine spontane Einsicht (einen sogenannten Aha- oder Heureka-Effekt) voraus. Aus informatischer Perspektive sind solche Problemlösungen nur schwer zu konzeptualisieren und damit zu algorithmisieren.

Explorationsprobleme.

Nach [60] unterscheiden sich komplexe Probleme gegenüber klassischen Problemen (s.o.) dadurch, dass sich das Problem während der Problemlösung dynamisch verändert und intransparent ist. Die Intransparenz ergibt sich daraus, dass die genauen Eigenschaften des gegebenen Zustands, des Zielzustands und der Barriere unbekannt sind. Eine Lösung komplexer Probleme setzt die effiziente Interaktion zwischen dem Lösenden und der situationsbedingten Anforderungen der Aufgabe voraus. Sie verlangt dabei gegebenenfalls kognitive, emotionale, persönliche und soziale Fähigkeiten und Kenntnisse.

Eines der bekanntesten untersuchten komplexen Szenarien ist „Lohhausen“, eingeführt von Dietrich Dörner [36] und zugleich einer der ersten systematischen Ansätze zur Untersuchung der Kognition komplexer Problemlösung. In diesem Szenario mussten Teilnehmer eine kleine Stadt als quasi allmächtiger Bürgermeister regieren. Um komplexe Szenarien wie Lohhausen erfolgreich zu bewältigen, so war die Hoffnung, ist mehr als die eher monolithischen Anforderungen klassischer Intelligenztests notwendig: Unterschiedlichste finanzielle Entscheidungen müssen getroffen werden, das Abstraktum „hohe Zufriedenheit“ der Bevölkerung verstanden werden, Güter müssen produziert werden und zugleich eine niedrige Arbeitslosenquote erreicht werden. Mehrere Merkmale zur Lösung komplexer Probleme spielen eine große Rolle: z. B. die Vernetzung und Abhängigkeit zwischen allen Faktoren (z. B. Zufriedenheit der Bevölkerung ist natürlich abhängig vom Ausmaß der Arbeitslosigkeit usw.). Ein wichtiges Ergebnis dieser Studie ist, dass klassische Intelligenztests eine vernachlässigbare Vorhersage für den Erfolg brachten – im Gegensatz etwa zum Selbstvertrauen [62]. Das Lohhausen-Paradigma, mit mehr als tausend Systemvariablen, ist sicherlich eines der komplexesten Szenarien, welches je in psychologischen Studien untersucht wurde. Typische menschliche Fehler, die dabei auftreten, sind [36] u.a.: *Falsche Annahmen, Mangelndes Verständnis gegenüber zugrunde liegenden zeitlichen Abläufen, Schwierigkeiten Systemveränderungen mental zu repräsentieren* und schließlich das „*Denken in Kausalketten statt in kausalen Netzen*“. Letzteres bedeutet eine Tendenz zur Linearisierung von Ursache-Wirkung und eine Reduktion auf oftmals nur eine Ursache.

Kritik an Lohhausen [62] äußerte eine geringe Zuschreibbarkeit von Aktionen zu abhängigen Variablen und unklare Anweisungen (z. B. wie lässt sich der „Zufriedenheitsgrad der Bevölkerung“ konzeptualisieren?).

Systemmerkmale komplexer Probleme informatischer Perspektive ähneln sich dabei [177]. Eine detaillierte Darstellung findet sich in [62]:

- *Elementkomplexität:* Die Elementkomplexität ist spezifiziert durch die Anzahl der Elemente, die zu einer Problemlösung wichtig sind. Je größer die Anzahl, desto schwieriger ist das Problem im Allgemeinen.
- *Vernetztheit:* Die Vernetztheit (gering vs. hoch vernetzte Probleme) definiert als die Anzahl und Dichte der Verknüpfung zwischen den Elementen ist dabei ein über die klassische Elementkomplexität hinausgehendes Charakteristikum. Dabei bedeutet die Vernetztheit den Einfluss, den die Änderung einer Variable auf die Änderung einer weiteren Variable hat (vgl. Lohhausen Paradigma).

- *Polytelie*: Ein weiterer Unterschied macht die Anzahl der Ziele aus. Beispielsweise sind in den Simulationen (Lohhausen) vielfältige Ziele für den Gesamterfolg relevant. Dabei lassen sich Probleme mit einem und Probleme mit mehreren eventuell konfigurierenden Zielen unterscheiden.
- *Intransparenz*: Die Intransparenz charakterisiert die Bekanntheit der Verknüpfungen und der Wirkungsbeziehungen. In klassischen Permutationsproblemen sind die Operatoren und die Wirkungen bekannt, sie stellen also transparente Probleme dar. Bei Lohhausen sind die Wirkungen einer Operation intransparent.
- *Dynamik*: Ausmaß der Eingriffsabhängigkeit des Systems (statische vs. dynamische Probleme). Ein Charakteristikum ist ein sich veränderndes Problem während der Deliberation des Agenten.

Solche komplexen Probleme sind auch aus informatischer Perspektive komplex und werden dort als Explorationsprobleme klassifiziert. Eine Verbindung von KI-Methoden und Optimierungsproblemen konnte aber zur Berechnung optimaler Lösungen [178] in einem verwandten komplexen Problem [163] führen. Komplexes Problemlösen ist also informatisch und kognitiv schwer und in beiden Bereichen wird oftmals nur eine heuristische Lösung probiert.

Insgesamt lässt sich also resümieren, dass sich menschliches Denken und Problemlösen von formalen Ansätzen der Informatik und Logik teilweise erheblich unterscheidet: Es ist *kontext-abhängig, heuristisch, repräsentationsabhängig* und gehorcht nicht rein *normativen Ansätzen*. Besonderheiten ergeben sich insbesondere aus Struktur und Beschränkung des Arbeitsgedächtnisses, der mentalen Repräsentation und es hat *nichtmonotone Charakteristiken*. Stärken liegen dabei im Auffinden von Querbezügen zwischen verschiedenen ähnlichen Bereichen, also im analogen Denken.

2.2.5 Soziale Kognition

Traditionell befasste sich die Kognitionswissenschaft mit rationalen Wahrnehmungs-, Denk- und Lernvorgängen, während Themenbereiche wie Emotion, Motivation oder soziale Fragen dort nicht betrachtet wurden. Die ersten Befunde, die diese strikte Trennung aufweichten, waren Ergebnisse zu Motivation und Emotion beim Lernen. Es gab auch schon frühe Versuche, den Gegenstand der Kognitionswissenschaft auszuweiten, um damit nicht-kognitive Phänomene zu erklären. Ortony und Turner [151] versuchten z.B. in einem informationstheoretischen Modell Emotionen als Produkt kognitiver Evaluation und emotionaler Valenz (positiv vs. negativ) zu formalisieren.

So richtig etabliert hat sich das Feld jedoch erst mit Entwicklung der kognitiven Neurowissenschaft und insbesondere den bildgebenden Verfahren (vgl. [150]). Statt einer qualitativen Trennung, die ja im Gehirn noch einfacher gelingen sollte, als mittels behavioraler Methoden, wurden immer wieder Ergebnisse berichtet, die eine ganzheitliche Verarbeitung von Reizen nahelegten, und damit auch eine Reihe von Prozess-Annahmen in Frage stellen. Man findet z.B. so frühe Effekte des emotionalen Gehalts von Stimuli, dass eine serielle Verarbeitung unmöglich scheint. Wie kann ein emotional aufregendes Wort, noch bevor es gelesen ist, schon Aktivierung in der Amygdala oder eine galvanische Hautreaktion auslösen? Obwohl solche Effekte oft berichtet werden, und meist evolutionär begründet werden (natürlich ist es wichtig, schnell fliehen zu können, wenn Gefahr droht!), ist bisher noch keine befriedigende Erklärung gefunden worden, wie dieses kognitiv-emotionale Interface aussehen könnte.

Für eine verstärkte Annäherung von Sozialpsychologie bzw. sozialen Fragestellungen und Kognitionswissenschaft lassen sich folgende verbindende Einflüsse feststellen:

- In der Sozialpsychologie werden vermehrt auch die kognitiven Grundlagen von Sozialverhalten untersucht. Welche Wissensstrukturen befördern Vorurteile? Wie beeinflusst das eigene Selbstbild die Selbsteinschätzung? Warum sind Menschen unvernünftig in ihren Entscheidungen, obwohl sie über die Risiken eigentlich aufgeklärt sein sollten (z.B. Rauchen).
- Um diese und ähnliche Fragen zu untersuchen, werden häufig Paradigmen aus der experimentellen Kognitionspsychologie entlehnt
- Die Neurobildung zeigt Überlappungen zwischen sozialen und kognitiven Aufgaben, die eine strikte theoretische Trennung aufweichen. Andere Befunde dissoziieren Prozesse (z.B. Inhibition), die in kognitionswissenschaftlichen Theorien nicht unterscheidbar sind
- Kognitive Prozesse, wie z.B. Problemlösen oder Sprachverständigen, werden stärker durch den Inhalt bzw. die Wissensdomäne beeinflusst (z.B. Evaluation von Selbst vs. Anderen), als dies theoretisch postuliert wird.
- Es wurde vermehrt festgestellt, dass kognitive Prozesse nicht immer abstrakt-rational sind, sondern oft von deutlich von Emotionen, Stimmungen, oder Motivationen beeinflusst werden.

Im Folgenden führen wir kurz in ausgewählte Forschungsthemen dieses Gebietes ein. Ausführlicher kann man das Thema bei Fiske und Taylor [52] nachlesen. Ein ausgezeichnetes, neues Lehrbuch über neurowissenschaftliche Befunde ist Ward [217].

Wahrnehmung sozial-emotionaler Reize: Gesichter

Nach dem klassischen informationstheoretischen Ansatz sollte die Wahrnehmung von Gegenständen oder Personen, oder auch das Verstehen von Sachinformation oder sozialen Gegebenheiten auf qualitativ ähnlichen Prozessen basieren. Die verschiedenen Domänen (physische Welt, Objekte vs. soziales Miteinander, Menschen) werden in entsprechenden Wissensspeichern repräsentiert. Neurowissenschaftliche und behaviorale Befunde scheinen jedoch Hinweise auf qualitative Unterschiede zu geben.

So wurde im unteren Temporallappen eine Hirnregion identifiziert, die spezifisch auf Gesichter reagiert. Reaktionen in diesem Areal, dem Gyrus Fusiformis, werden schon von einfachen Strichzeichnungen ausgelöst („Smileys“). Innerhalb des Areals scheint es aber Neuronen zu geben, die primär auf eine bestimmte Person ansprechen („Großmutterzellen“), und es ermöglichen, Gesichter voneinander zu differenzieren und zu identifizieren. Dies ist eine erstaunlich komplexe Aufgabe: Gesichter ändern sich ständig. Personen bewegen sich, sie sprechen, drehen den Kopf, runzeln die Stirn oder schließen die Augen. Dennoch gelingt es fast allen Menschen ohne Schwierigkeiten, eine Vielzahl von bekannten – oder unbekannten – Gesichtern wiederzuerkennen.

Von der Identifikation möglicherweise trennbar ist die Fähigkeit (vgl. [217]), emotionale Gesichtsausdrücke zu erkennen. Für eine kleine Zahl von Basisemotionen ist diese Fähigkeit universell. Menschen aller Kulturen stimmen in der Bewertung überein, ob ein Gesicht fröhlich, ängstlich oder traurig aussieht. Ein Ansatz, die subtilen Unterscheidungen zu formalisieren – statt von einer intuitiven, erlernten Zuordnung auszugehen –, ist das Facial Action Coding Schema (s. [40]), in dem emotionale Gesichtsausdrücke durch das Zusammenwirken einer Vielzahl von Gesichtsmuskeln beschrieben und objektiviert werden. Damit kann versucht werden, Emotionserkennung auch in künstlichen Systemen zu implementieren.

Der Einfluss von Emotion auf kognitive Prozesse

Eine wichtige Frage ist, ob bzw. wie kognitive Prozesse durch Emotionen oder Motivationen beeinflusst werden (s. [33,35]). Intrinsische Motivation aus eigenem Interesse führt zu besseren Lernerfolgen als extrinsische, die durch Vorgaben Anderer entsteht. Ängste oder Depressionen können Verlangsamung oder Aufmerksamkeitsschwierigkeiten und damit Leistungseinbußen bedingen. Sogar schlechte Laune kann ähnlich wirken. Wird z.B. die Stimmung einer Versuchsperson durch fröhliche Musik oder sogar durch eine wohlig warme Tasse Tee vor dem Experiment verbessert, dann schneidet sie z.B. in Tests zu Kreativität besser ab. Sie ist auch weniger streng in ihren Urteilen, als jemand, der in eine schlechte Stimmung versetzt wurde (z.B. durch hektische, atonale Musik oder durch langes Warten).

Spezifischere Emotionseffekte werden durch eine Übereinstimmung zwischen der emotionalen Situation und den zu verarbeitenden Reizen erzeugt. Positiv gestimmte Personen sind schneller beim Lesen von Sätzen über positive Ereignisse, als über negative („Er hat das Rennen gewonnen/verloren“), und sie erkennen fröhliche Gesichtsausdrücke leichter als traurige oder ängstliche.

Aber auch in neutraler Stimmung hat die emotionale Valenz der zu verarbeitenden Reize einen messbaren Einfluss auf die Kognition. Ein Wort laut vorzulesen geht schneller für Wörter mit positiver Konnotation, als für Wörter mit negativer oder neutraler Konnotation – obwohl die emotionale Valenz für die Aufgabe völlig irrelevant ist. Positiven Aussagen stimmen Versuchspersonen lieber zu („Polyanna-Effekt“). Bei Diskussionen werden die Argumente leichter erinnert und stärker gewichtet, welche die eigenen Überzeugungen stützen, die also schon positiv konnotiert sind („confirmation bias“).

Nüchterne, rationale Verarbeitung gibt es kaum. Kognition ist immer von emotionalen Faktoren beeinflusst oder überlagert. Sowohl der emotionale Gehalt als auch die Stimmung des Menschen sind wesentlich. In der Neurowissenschaft gibt es ansatzweise Erklärungsversuche; in der traditionellen Kognitionswissenschaft sind diese Faktoren bisher noch wenig in die Theoriebildung eingeflossen.

Kognition über andere: Theory-of-Mind

Theory-of-Mind (ToM) bezeichnet die Fähigkeit zu erkennen, dass andere Menschen intentionale Wesen sind, deren Verhalten durch ihre Motivationen, Gefühle und Überzeugungen beeinflusst wird [56]. Der Ausdruck – der nicht ins Deutsche übersetzt wurde –, bezeichnet also keine wissenschaftliche Theorie, sondern die Theorie, welche eine Person über ihre Mitmenschen ableitet. Eine ausgebildete ToM wird als Voraussetzung für sinnvolle soziale Interaktion gesehen. Während es keinen Zweifel gibt, dass schon Kleinkinder einfache intentionale Handlungen interpretieren (z.B. Zeigegesten), ist nicht unumstritten, ob dies auch ToM im engeren Sinne erfordert (auch ein Roboter kann zeigen). Eine klassische Aufgabe, mit der die voll ausgebildete ToM getestet wird, ist die sogenannte „Second-order false belief“ Aufgabe, oder auch Sally-Anne-Aufgabe. Dabei wird eine Geschichte erzählt, oder eine Bildergeschichte gezeigt, bei der Sally erst ihren Ball in eine Kiste legt und dann aus dem Zimmer geht. Nun kommt Anne und versteckt den Ball unter dem Bett. Wo wird Sally den Ball suchen, wenn sie wieder herein kommt? Um die richtige Antwort zu geben (in der Kiste), muss die tatsächliche Situation in der Welt von dem unterschieden werden, was Sally wissen oder glauben kann. Kinder können diese Aufgabe erst im Vorschulalter sicher bewältigen.

Die aus der Sicht der Kognitionspsychologie interessante Fragestellung ist nun, ob ToM als eigenständiger, von anderen kognitiven Prozessen qualitativ unterscheidbarer Prozess gesehen werden kann. Offenbar erfordert die Sally-Anne-Aufgabe vielfältige kognitive Leistungen: Arbeitsgedächtnis, um sich die Geschichte zu merken; Sprachverstehen, um die Frage interpretieren zu können; Inhibition, um die falsche Antwort, die ja in der Repräsentation des Szenarios auch enthalten ist zu unterdrücken; und evtl. komplexe Sprachproduktion, um die Situation zu kodieren („Ich glaube, dass Sally denkt, dass der Ball in der Kiste ist, weil sie nicht weiß, dass Anne ihn versteckt hat“). Obwohl auch diese Fähigkeiten sich erst im Vorschulalter ausbilden, ist es schwierig eine Richtung abzuleiten: es kann ja schließlich auch sein, dass die komplexe Syntax umgekehrt erst ausgebildet wird, wenn die Gedankenwelt des Kindes es erfordert, entsprechend komplexe Sachverhalte auszudrücken [48].

Ähnliche Argumente gelten natürlich auch für die Ko-Evolution von Exekutivfunktionen und ToM: Auch die Fähigkeit zur Manipulation von alternativen Sachverhalten im Arbeitsgedächtnis, und die Fähigkeit zur Inhibition von situations-irrelevanten Inhalten entwickelt sich ungefähr gleichzeitig mit ToM. Während also kognitionswissenschaftliche Theorien ToM durch die Inhalte charakterisieren würden, aber domänen-übergreifende Denk-Prozesse annehmen (z.B. Inferenz, [50]), wird in der Entwicklungs-, Neuro- oder evolutionären Psychologie oft davon ausgegangen, dass es sich tatsächlich um einen qualitativ dissoziierbaren Prozess handelt, der anderen Gesetzen folgt, als Denken und Problemlösen über nicht-soziale Inhalte. Ein Spezialfall dieses Ansatzes ist der Versuch, kontextabhängige Performanz im Problemlöseverhalten durch die sogenannte „Cheater Detection“ zu erklären (s. Abschnitt 2.2.4). Eine Vielzahl von Experimenten hat gezeigt, dass Regelverletzungen schnell und mühelos erkannt werden, wenn sie dazu führen, Individuen zu identifizieren, die sich auf Kosten anderer bereichern wollen (z.B. Varianten der Wason-Selection-Task; [30]). Während dieser Ansatz aus evolutionärer Sicht plausibel scheint, bereitet er im Rahmen eines Informationsverarbeitungsparadigmas Schwierigkeiten. Es wird interessant sein zu sehen, wie sich diese Theorien weiter entwickeln werden.

Kognition über andere: Urteile und Vorurteile

Theory-of-Mind ist die Grundlage für situationsbezogene Interpretationen der Handlungen oder Reaktionen anderer. Kognition über andere Menschen beinhaltet aber auch schon vorgefertigte Meinungen und Kenntnisse. Ein Kernthema in der sozialen Kognitionsforschung, dessen Ergebnisse auch für Politik und Pädagogik wesentlich sind, ist die Entstehung und Aufrechterhaltung von Vorurteilen. Ein Problem bei der sozialpsychologischen Forschung zu diesem Thema war lange, dass sie auf Fragebogenerhebungen und damit Selbsteinschätzungen der Versuchspersonen basierte. Wer gibt aber schon gerne zu, etwa Ressentiments gegen Minderheiten zu haben, oder mit Terroristen zu sympathisieren? Hier hat die kognitionspsychologisch begründete experimentelle Forschung deutliche Fortschritte gebracht. Der implizite Assoziationstest (IAT, [73]) basiert auf Interferenzeffekten beim Aufgabenwechsel. Versuchspersonen sollen z.B. die rechte Taste drücken, wenn sie einen ausländischen Vornamen sehen, und die linke, wenn ein deutscher Vorname präsentiert wird. In einer zweiten Aufgabe werden sie instruiert, die emotionale Valenz von Adjektiven einzuschätzen (rechts: negativ, links: positiv). Diese beiden Aufgaben werden nun in der Reihenfolge durcheinander präsentiert. Für ausländerfeindliche Versuchspersonen ist diese Antworttasten-Zuordnung kongruent mit ihrer Überzeugung (rechts: Ausländer/negativ). Wie viel schneller die Reaktionszeiten in dieser Bedingung sind, im Vergleich mit einer inkongruenten Tastenzuordnung (rechts: Ausländer / positiv), kann als Maß für die Stärke des Vorurteils genommen werden. Priming-Experimente können ebenso über Vorurteile

Aufschluss geben. Payne ([154]; s. auch [1]) instruierte Versuchspersonen, schnell eine Taste zu drücken, sobald sie das Bild einer Waffe sahen. Die Reaktionszeiten in dieser Monitoring-Aufgabe waren schneller, wenn unmittelbar vorher ein Gesicht schwarzer Hautfarbe gezeigt wurde (im Vergleich mit einer Person weißer Hautfarbe).

Diese und ähnliche Paradigmen, sowie ihre neurowissenschaftlichen Pendants, lassen sich auch auf andere Fragestellungen in der Sozialpsychologie anwenden, wie z.B. Einstellungen zu Risikoverhalten, Gesundheitsvorsorge, Aggression und viele mehr. Im Gegensatz zu den Bereichen Emotion oder ToM lässt sich dieses Vorgehen unmittelbar in einen kognitionswissenschaftlichen Ansatz integrieren.

Kognition in der Gruppe

Ein weiteres Thema an der Schnittstelle zwischen Kognition und Sozialpsychologie – das hier nur kurz angerissen werden soll – ist das Verhalten von Menschen in der Gruppe. Beeinflusst die Gruppengröße das Kommunikationsverhalten? Wie finden Menschen zu einer gemeinsamen Sprachregelung, wenn sie über ein neues Thema miteinander sprechen? Wie werden Dialoge strukturiert? Untersuchungen zu solchen und ähnlichen Fragestellungen ergeben, dass Kommunikationspartner mittels linguistischer und metakognitiver Mittel aktiv dazu beitragen, eine gemeinsame Grundlage („common ground“) zu schaffen [159].

Auch die Wissensvermittlung bzw. das Problemlöseverhalten in der Gruppe ist eine interessante angewandte Fragestellung der Kognitionswissenschaft. Wie wird Informationsaustausch in Arbeitsgruppen organisiert? Sind Teams – z.B. im Bereich der Computerprogrammierung – effizienter als einzelne Mitarbeitende? Werden Entscheidungen in einer Gruppe (z.B. in einer Jury bei Gericht) eher auf objektiven Kriterien basiert als wenn einzelne Personen ihre individuelle Meinung äußern? Oder ist es umgekehrt: unter welchen Umständen lassen sich Menschen durch Gruppenzwang in ihrem Urteil beeinflussen? Neben experimentalpsychologischen Modellen kann auch zu diesen Fragen die kognitive Modellierung theoretische Impulse liefern.

Zusammenfassung

Der Bereich der sozialen Kognition nimmt innerhalb der KW einen immer stärkeren Raum ein. Kognitionswissenschaftliche Methoden und Modelle lassen sich direkt auf sozialpsychologische Themenbereiche anwenden. Eine größere Herausforderung für die Theoriebildung sind evolutionäre Ansätze, die qualitative Unterscheidung zwischen sozial relevanten Prozessen (z.B. ToM) und kognitiver Verarbeitung postulieren. Es wird spannend zu sehen, wie sich dieses Feld weiter entwickeln wird.

2.2.6 Sprache

Die Fähigkeit zur Verwendung von Sprache gehört zu den herausragenden Eigenschaften menschlicher Kognition. Die Erforschung menschlicher Sprache und des menschlichen Sprachverhaltens ist Gegenstand aller kognitionswissenschaftlichen Einzeldisziplinen. Die Kognitionswissenschaft der Sprache wird oft als „Psycholinguistik“ bezeichnet und umfasst auch Erkenntnisse aus den Neuro- und Computerwissenschaften (Computational Psycholinguistik).

Sprache kann einerseits als Medium der Kommunikation, und anderseits als Medium des Denkens angesehen werden. Je nach Betonung dieser beiden Funktionen gelangt man zu völlig verschiedenen Forschungsansätzen, die entweder interktionale und funktionale, oder aber die

formalen Aspekte der Sprache in den Vordergrund rückt. Der weitaus größere Teil der Psycholinguistik hat sich mit den formalen Aspekten der Sprache beschäftigt, und mit den Problemen, die sich daraus für die Informationsverarbeitung ergeben. Mehr und mehr jedoch werden heute die interaktionalen Grundlagen beleuchtet und als grundlegend für die Entstehung von Sprache und den Erwerb im Kindesalter angesehen.

Sprachverarbeitung als Informationsverarbeitung

Auch wenn man Sprache als ein Problem der Informationsverarbeitung begreift, muss man sich darüber im Klaren sein, dass viele Ebenen durchlaufen werden müssen, um den Sinn einer Äußerung zu extrahieren. Bei geschriebener Sprache sind das – grob gesagt – die Ebenen der Worterkennung (Orthographie, Morphologie, Morphosyntax, Lexikalische Semantik, etc.), der Satzerkennung (Syntax, Semantik, Informationsstruktur, etc.), und der Diskursverarbeitung (Kohärenz, Referenzen, Inferenzen, Pragmatik, etc.). Bei gesprochener Sprache kommen noch die Ebenen der Phonetik und der Phonologie hinzu.

Ein zentrales Problem der Sprachverarbeitung ist das hohe Maß an Ambiguität auf allen Ebenen: Wörter können mehrere Bedeutungen haben (z.B. „Bank“), mehreren syntaktischen Kategorien angehören (z.B. „sein“), morphologisch unterschiedlich dekomponiert werden (Stau-becken vs. Staub-ecken). Bei gesprochenem Text ist der Grad an Mehrdeutigkeit um einiges höher, da hier noch Probleme der Segmentierung und der phonologischen Enkodierung hinzukommen. Jenseits der Wortebene sind es vor allem strukturelle, in der Grammatik begründete Ambiguitäten, in zunehmenden Maße aber auch semantische (z.B. Bedeutung und Skopos von Quantifizierern wie „manche“, „nur wenige“, „alle“ etc. oder referentielle Ambiguitäten; ein Überblick findet sich in [59]), die Gegenstand psycholinguistischer Forschung sind. Eine Klassifikation findet sich bei [108]. Da sich Ambiguitäten im Verlauf der Verarbeitung vom Anfang zum Ende eines Satzes bzw. Textes multiplizieren, stehen wir vor einem hochkomplexen Suchproblem. Interessanterweise hängt die beobachtbare Verarbeitungszeit einzelner Wörter im Satz bei Menschen in der Regel nicht vom Grad der Ambiguität ab und nimmt auch zum Ende eines Satzes nur unwesentlich oder gar nicht zu, manchmal sogar ab. Gleichzeitig werden Ambiguitäten in den seltensten Fällen bewusst. Auf der anderen Seite ist wiederholt gezeigt worden, dass wir Sätze hochgradig inkrementell (d.h. Wort für Wort) so verarbeiten, dass jedes neue Wort sofort im Kontext der vorangegangen syntaktisch analysiert und interpretiert wird (z.B. [88]). Offensichtlich entscheiden wir uns im Falle von auftretenden Ambiguitäten sehr früh für eine, oder nur eine geringe Auswahl der möglichen Alternativen. Die Tatsache, dass uns Ambiguitäten nur selten bewusst werden, nämlich meist dann, wenn sich im späteren Verlauf der Verarbeitung herausstellt, dass wir uns falsch entschieden haben (der sog. „Holzweg“ oder „Garden-path“ Effekt), deutet darauf hin, dass wir uns in den weitaus meisten Fällen richtig entscheiden. Eine der Hauptfragen der Psycholinguistik lautet daher: Welche Prinzipien oder Mechanismen leiten die frühzeitige Auflösung, oder Einschränkung, von Ambiguitäten? Die Beantwortung dieser Frage kann nur auf der Basis genauer Annahmen über das zugrunde liegende Wissen und die Verfahren seiner Nutzung erbracht werden.

Während wir auf der einen Seite über äußerst effiziente Mechanismen zur Verarbeitung ambigen Materials verfügen, scheitern wir gelegentlich an eindeutigen, aber zu komplexen Sätzen. Ein gängiges Beispiel sind Zentraleinbettungen:

Der Hund, der die Katze, die die Maus gefangen hatte, jagte, stolperte.

Hier sind drei Sätze so ineinander verschachtelt, dass die Verben, die die Sätze abschließen, erst nach dem jeweils eingebetteten Relativsatz verarbeitet werden können.

Erklärungen für dieses Phänomen basieren meist auf der Annahme, dass durch die Mehrfacheinbettung die Kapazität des menschlichen Arbeitsgedächtnisses überschritten wird.

Psycholinguistische Methoden

Psycholinguistische Daten werden entweder off-line, d.h. im Anschluss an die vollständige Verarbeitung eines Textes, oder on-line, d.h. während der Verarbeitung, erhoben. On-line Techniken haben den Vorteil, dass sie detaillierte Verarbeitungsprofile ermöglichen, die weit über die globale Beurteilung eines Satzes hinausgehen. Off-line Techniken hingegen sind in ihrer Anwendung wesentlich weniger aufwendig und werden oft in Normierungsstudien eingesetzt oder um einen ersten Anhaltspunkt über globale Präferenzen zu erhalten.

Typische off-line Techniken sind Fragebögen mit Akzeptabilitätsurteilen, Satzvervollständigungen oder multiple-choice Aufgaben. Typische on-line Techniken sind das selbst-getaktete Lesen (self-paced reading: Versuchspersonen drücken eine Taste um einen neuen Textabschnitt lesen zu können) sowie die Aufzeichnung von Blickbewegungen während des Lesens (z.B. [168]). Mit beiden Techniken werden Lesezeiten für jeden einzelnen Textabschnitt ermittelt, die den aktuellen kognitiven Aufwand widerspiegeln.

Inzwischen gibt es auch umfangreiche Lesekorpora, die aus mehr oder weniger natürlichen Texten bestehen, die nicht zu experimentellen Zwecken konstruiert wurden [19, 105]. Diese Texte hat man von vielen Personen lesen lassen und dabei ihre Blickbewegungen aufgezeichnet. Solche Daten sind eine wertvolle Quelle für psycholinguistische Studien, da sie unter denkbar natürlichen Umständen entstanden sind, und oft die Untersuchung neuer Fragestellungen erlauben, an die man bei der Erstellung der Korpora noch gar nicht gedacht hat.

Es gibt auch on-line Techniken, mit denen sich die Verarbeitung gesprochener Sprache untersuchen lässt. Beim *Visual-World Paradigma* etwa werden gesprochene Texte vorgespielt, und gleichzeitig die Blickbewegungen auf Abbildungen von Gegenständen oder realen Szenarien aufgezeichnet. Aus den Blickdaten lässt sich dann schließen, zu welchem Zeitpunkt welcher Teilaspekt des gesprochenen Stimulus verstanden worden ist.

Heute sind besonders neurowissenschaftliche Methoden populär, wie die Aufzeichnung von Potential- und/oder Magnetfeldveränderungen an der Schädeloberfläche während des Lesens oder Hörens von Texten. Mit Hilfe solcher Aufzeichnungen ist es möglich geworden, verschiedene Prozessklassen (wie syntaktische versus semantische Verarbeitung) in den Daten qualitativ zu unterscheiden (positive Shifts in bestimmten Zeitfenstern reflektieren eher syntaktische, negative eher semantische Prozesse), und sie in verschiedenen Arealen des Gehirns zu lokalisieren (z.B. [61, 79, 152]).

Da jedes kognitive Modell letztlich auf die neurophysiologischen Gegebenheiten des Gehirns abbildbar sein muss, werden auch aus der Neurolinguistik wesentliche Rahmenbedingungen gesetzt. Gerade die Untersuchung der Beeinträchtigungen des Sprachverhaltens bei hirnorganischen Funktionsstörungen ermöglicht Aussagen über die „normale“ Funktionsweise des Systems [130].

Die Funktion der Sprache schließlich, als Vermittlerin zwischen dem Menschen und seiner Umwelt, das Verhältnis also von Denken, Sprache und Wirklichkeit wird eingehend in der Philoso-

phie diskutiert [74, 181]. Der Rolle der Sprache im menschlichen Handeln oder als wirklichkeitskonstituierende Strukturierungshilfe bleibt auch für die Entwicklung von Modellen menschlicher Sprachverarbeitung wichtig [91].

Die Architektur des menschlichen Sprachverarbeitungssystems

Eine der Hauptfragen für eine Theorie der Sprachverarbeitung betrifft das Zusammenspiel oder die Interaktion aller Wissensquellen. Hier lassen sich zunächst zwei Extrempole ausmachen:

- die *modulare* Auffassung des Sprachverarbeitungssystems (SVS), nach der zumindest die syntaktische Analyse als autonomes Modul [55] von der Verarbeitung weiterer Information abgekoppelt ist. Dabei wird angenommen, dass sie der Verarbeitung auf höherer Ebene zeitlich und logisch vorgeschaltet ist und sich von keinerlei Information nachgeschalteter Prozesse beeinflussen lässt.
- die *interaktive* Auffassung, gemäß der Information aller Ebenen direkt oder indirekt die Verarbeitung auf jeder anderen Ebene leiten kann.

Vertreter der Auffassung, dass die Syntax weitgehend autonom ist, sind neben Fodor [55] vor allem Frazier [57, 58] und Kollegen, die mit ihrem Verarbeitungsmodell, der sogenannten *sausage machine*, und später mit ihrem *Garden-path*-Modell der Syntaxanalyse den Vorrang vor der Verarbeitung auf höheren Ebenen gibt.

Interaktive Modelle variieren enorm in der Art und Weise der angenommenen Interaktion, d.h. im angenommenen Informationsfluss zwischen den verschiedenen Ebenen der Verarbeitung. Marslen-Wilson und Tyler [128, 129] haben in sog. *Shadowing*-Experimenten gezeigt, dass Versuchspersonen, die einen Text noch während des Hörens nach- oder mitsprechen sollten, nur wenige zehntel Sekunden hinter dem Original herhinkten und dabei Korrekturen von gezielt eingestreuten kleinen Fehlern vornahmen (*compsiny* statt *company*). Solche schnellen „konzeptgesteuerten“ Verbesserungen deuten auf einen extrem schnellen Informationsaustausch zwischen allen Ebenen der Verarbeitung hin, wie er vor allem von interaktiven Modellen vorhergesagt wird (Auf der anderen Seite kann ein modulares System nich allein durch Betrachtung der Geschwindigkeit des Informationsaustausches widerlegt werden).

Zu den wichtigsten interaktiven Modellen zählen die von Just und Carpenter [100], Juliano & Tanenhaus [98] sowie MacDonald et al. [121]. Meist handelt es sich dabei um konnektionistische oder hybride Modelle, bei denen multiple Constraints zu jedem Zeitpunkt miteinander im Wettstreit stehen. Widersprechen sich die Constraints verschiedener Ebenen etwa bei der Auflösung einer Ambiguität, wird mehr Zeit für die Auflösung benötigt.

Der Klassiker: Das Holzweg-Modell

Frazier [57, 58] postuliert Verarbeitungsprinzipien, die sich ausschließlich an strukturellen Eigenschaften des Satzes orientieren. Ein Satz wird inkrementell von links nach rechts so verarbeitet, dass jedes zu verarbeitende Wort nach den Regeln der Grammatik in die bis dahin aufgebaute Satzstruktur integriert wird. Erlaubt die Grammatik mehr als eine Integrationsmöglichkeit, wird die Ambiguität auf der Basis der Prinzipien „Minimal Attachment“ (MA) und „Late Closure“ (LC) aufgelöst.

(p1) *minimal attachment principle:*

Postuliere keine Knoten, die sich später möglicherweise als unnötig erweisen.

(p2) *late closure* [57]:

Ordne nach Möglichkeit jedes neue Item der Phrase zu, die momentan verarbeitet wird.

Wird beispielsweise der Satz *The spy saw the cop with binoculars* gelesen, so wird nach den ersten Worten *The spy* und *saw* die Struktur [S [NP *The spy*] [VP *saw* ...] aufgebaut. Die Anbindung des nun folgenden Artikels *the* kann aber auf zweierlei Weise geschehen: Erstens, indem eine Nominalphrase postuliert wird, die sich nach der Regel „VP → V NP PP“ in die Verbalphrase integrieren lässt, oder aber indem nach der Regel „NP → NP PP“ zunächst eine komplexe Nominalphrase postuliert wird, die erst dann in die Verbalphrase integriert werden kann. MA entscheidet hier zugunsten des Aufbaus der sparsameren Struktur, so dass später die Präpositionalphrase *with binoculars* unmittelbar in die Verbalphrase integriert werden muss, da für die Anbindung an die vorangegangene NP die bisherige Struktur revidiert und erst ein zusätzlicher NP-Knoten eingefügt werden müsste.

Scheitert im weiteren Verlauf die gewählte Analyse, etwa weil sie sich als unplausibel erweist, wird eine Reanalyse eingeleitet, in der die komplexere, aber in diesem Fall plausiblere Struktur aufgebaut wird. Das Garden-Path-Modell sagt also für die Verarbeitung von Sätzen mit letztlich nicht minimaler Struktur einen messbar höheren Verarbeitungsaufwand vorher, der auf eine erzwungene Reanalyse zurückzuführen ist.

Crain & Steedman [32] zeigten anhand von Sätzen wie (2a,b), dass *Holzweg*-Sätze leichter zu verstehen sind, wenn durch das Weltwissen die weniger präferierte Lesart nahegelegt wird.

(2a) *The teachers taught by the Berlitz method passed the test.*

(2b) *The children taught by the Berlitz method passed the test.*

Das Wissen darüber, dass Kinder eher lernen als lehren, erleichtert es z.B. in (2b), die Phrase *taught by the Berlitz method* als Relativsatz („who were taught ...“) zu verstehen. In (2a) hingegen fehlt ein solcher Hinweis des Weltwissens, und in der Tat bleiben Leser länger bei der zunächst präferierten Hauptverb-Lesart „Die Lehrer lehrten ...“. Befunde wie diese zeigen, dass andere als syntaktische Wissenquellen, wie z.B. Hintergrundwissen über semantische Zusammenhänge in der Welt, einen so frühen Einfluss auf die Verarbeitung temporärer Mehrdeutigkeiten haben können, dass sich mit Verhaltensdaten kaum noch zwischen autonomen und interaktiven Modellen entscheiden lässt.

Prinzipien wie *Minimal Attachment* gelten als universell; d.h. es wird postuliert, dass sie für Sprecher/Hörer beliebiger Sprachen gültig sind. Diese Annahme wurde in den letzten Jahren insbesondere aufgrund von Evidenzen aus sprachvergleichenden Studien angezweifelt. Scheinbar abweichend von den universellen Vorhersagen des *Late-closure* Prinzips (p2) und Befunden aus dem Englischen findet sich bei Relativsätzen (4) in vielen Sprachen (so im Spanischen, im Französischen, im Niederländischen und im Deutschen) eine Präferenz zur „hohen“ Anbindung, d.h. zur Anbindung an den Kopf der komplexen NP (*Dienerin*).

(4) *Jemand erschoss die Dienerin der Schauspielerin, die auf dem Balkon war.*

Diese und ähnliche Befunde stehen im Widerspruch zu einer universellen Erklärung menschlicher Verarbeitungspräferenzen [139]. Die beobachtbaren Unterschiede zwischen verschiedenen Sprachen werden häufig auf die statistische Verteilung der Strukturen in der jeweiligen Sprache bzw. in der individuellen Lerngeschichte der einzelnen Sprecher/Hörer zurückgeführt (p3).

(p3) *tuning hypothesis* [139]:

Individuelle Strategien entwickeln sich, weil sie in der Vergangenheit häufiger erfolgreich waren. Strategien sind exposure based.

Arbeitsgedächtnis oder Erfahrung?

Aktuelle Ansätze versuchen heute, Präferenzen der Auflösung von Ambiguitäten als Spezialfall allgemeinerer Komplexitätsüberlegungen zu verstehen. Verschiedene Sätze sind unterschiedlich komplex, was zu mehr oder weniger stark ausgeprägten Verarbeitungsschwierigkeiten führen kann. Vergleicht man beispielsweise Satz (4) mit Satz (5)

- (4) The reporter that the senator attacked admitted the error. (Objektrelativsatz)
- (5) The reporter that attacked the senator admitted the error. (Subjektrelativsatz)

stellt man fest [106], dass (4) messbar langsamer gelesen wird als (5), obwohl die darin enthaltenen Wörter exakt die gleichen sind, und sich zudem die inhaltliche Komplexität nicht unterscheidet. Beide Sätze sind zudem zu keinem Zeitpunkt strukturell mehrdeutig. Von den vielfältigen Vorschlägen, die zur Erklärung des Unterschieds angeführt werden, sollen hier zwei erläutert werden, weil sie exemplarisch für zwei Klassen von Theorien stehen: Arbeitsgedächtnis-basierte (Abruf-) Modelle, und erfahrungsbasierte Modelle.

Arbeitsgedächtnis-basierte Ansätze gehen davon aus, dass zusätzlicher Verarbeitungsaufwand immer dann entsteht, wenn voneinander abhängige Elemente (Dependenten) eines Satzes miteinander integriert werden müssen. Wird z.B. ein Verb verarbeitet, müssen seine Dependenten (Subjekt, Objekt, etc.) aus dem Gedächtnis abgerufen und mit dem Verb integriert werden. Der Gedächtnisabruf ist jedoch um so schwieriger, je größer die Entfernung der Dependenten zueinander ist [67], oder je mehr ähnliche Gedächtniselemente beim Abruf miteinander interferieren [72, 214, 215]. Im Falle eines englischen Objektrelativsatzes (... *that the senator attacked*, in 4), müssen beim Verb *attacked* das Objekt *that*, und das Subjekt *the senator* abgerufen und integriert werden, wobei insbesondere der Abruf des Objekts über eine größere Distanz hinweg geschieht. Im Gegensatz dazu muss beim Verb *attacked* im Subjektrelativsatz (5) nur das unmittelbar benachbarte Subjekt (*that*) abgerufen werden; das Objekt steht hier noch aus. Der Integrationsaufwand ist somit beim Verb im Objektrelativsatz (4) deutlich höher. Gleichzeitig entsteht beim Abruf des Subjekts in Objektrelativsätzen (4) das Problem, dass eine ähnliche Phrase (*the reporter*) beim Abruf mit der richtigen (*the senator*) interferiert. Auch dieses Problem tritt in englischen Subjektrelativsätzen nicht auf.

Erfahrungsbasierte Ansätze stellen statistische Verteilungen von Konstruktionen oder Wortfolgen in der sprachlichen Umgebung der Sprecher in den Vordergrund. Solche Verteilungen lassen sich etwa durch Korpora schätzen und z.B. durch mit Wahrscheinlichkeiten annotierten Phrasenstrukturregeln repräsentieren. Probabilistische Grammatiken können nicht nur Präferenzen zur Auflösung von Ambiguitäten – ganz im Sinne der *Tuning-Hypothese* (p3) – abbilden [99], statistische Information kann auch genutzt werden, um das Ausmaß an inkrementellem Verarbeitungsaufwand zu berechnen. Hale [81, 82] berechnet Verarbeitungsaufwand aus dem Informationsgehalt, der sich (im informationstheoretischen Sinn) aus der Menge der Unsicherheit über mögliche grammatischen Fortsetzungen (Entropie) ableiten lässt, die mit jedem Wort reduziert wird. Levy [118] leitet das verwandte Maß für *Überraschung (surprise)* aus grammatischen Übergangswahrscheinlichkeiten ab. So lassen sich auch Anti-Lokalitätseffekten

[108] erklären, die sich z.B. in der schnelleren Verarbeitung von Verben am Ende eines Teilsatzes manifestieren. Diese haben eine größere Distanz zu ihren Dependenten, aber dadurch reduziert sich die Menge grammatischer Möglichkeiten bis zum Verb. Solche Vorfahrtssagen sind in Abruf-basierten Modellen nicht ohne weiteres möglich.

Konnektionistische Modelle stellen eine radikalere Form erfahrungsbasierter Ansätze dar, da sie nicht nur die stochastischen Eigenschaften der Sprachumgebung aufnehmen, sondern zugleich lexikalisches, syntaktisches und semantisches Wissen quasi aus dem Nichts erwerben. Im Extremfall, etwa in einer rekurrenten Netzwerkarchitektur [41] erwirbt ein System durch bloße assoziative Verkettung von Wörtern in Trainingskorpora implizites Wissen über syntaktische und semantische Kategorien. Solche *emergentistischen* neuronalen Netze lernen Generalisierung von Konstruktionen und zeigen in der Verarbeitung Transfer zwischen ähnlichen Wortfolgen. Verarbeitungsschwierigkeiten bei komplexem eindeutigem Material können so auf die geringere Regularität bestimmter Wortfolgen zurückgeführt werden. So wird der gemessene Unterschied in den Lesezeiten zwischen englischen Objektrelativsätzen (4) und Subjektrellativsätzen dadurch erklärt, dass Subjektrellativsätze (*that attacked the senator*) eine quasi-reguläre Wortreihenfolge aufweisen, die derjenigen von sehr häufigen Hauptsätzen entspricht (Subjekt-Verb-Objekt), während Objektrellativsätze mit ihrer Objekt-Subjekt-Verb Reihenfolge keinem vergleichbar regulären Muster entsprechen [120]. Durch längeres Training, d.h. durch mehr Erfahrung, verringerte sich dieser Nachteil von Objektrellativsätzen. So können inter-individuelle Unterschiede in der Verarbeitung solcher Sätze [106] auf den Grad der sprachlichen Erfahrung zurückgeführt werden [220].

Rekurrente Netze sind ein Beispiel nicht-linearer dynamischer Systeme, die in ihrem Modellierungsanspruch deutlich weiter gehen (oder tiefer, im Sinne von Marr [127]) als symbolische oder hybride Modellierungsansätze, die probabilistische Zusammenhänge häufig eher beschreiben als erklären. Varianten dynamischer Systeme erklären sprachliche Prozesse auf vielen Ebenen, etwa im Bereich des Sprachwandels [16, 205], des Spracherwerbs [42, 122], und der Sprachverarbeitung [28, 110, 112, 204]. Sie erklären zudem vermeintlich „irrationale“ Phänomene der Sprachverarbeitung, wie die Verarbeitung von Sätzen wie (6).

(6) Manfred, dem die Astronautin erfreut den Außerirdischen präsentierte, erschrak zu Tode.

In diesem Satz verbirgt sich die Wortfolge *die Astronautin erfreut den Außerirdischen*, die für sich genommen als Satz verstanden werden könnte, im Kontext der vorangegangenen Wörter *Manfred, dem ...* grammatisch aber als solcher niemals wahrgenommen werden dürfte, geht man von einer unmittelbaren inkrementellen Integration aller Wörter aus. Solche *lokalen syntaktischen Kohärenzen* führen jedoch tatsächlich zu messbaren Interferenzen [109, 111, 203]. Rekurrente Netzwerkmodelle sagen solche Interferenzen vorher, ohne einen Mehraufwand an Berechnung oder Speicherung für die lokalen Kohärenzen anzunehmen, wie es symbolische Verarbeitungsansätze zwangsläufig tun müssen. In solchen wäre dieses Phänomen nur durch Zusatzannahmen fassbar (siehe: [83, 118]).

2.2.7 Schlusswort

Im ersten Teil dieses Kapitels ist deutlich geworden, dass die Verflechtungen zwischen Kognitionswissenschaft und Informatik sehr eng sind. Dies zeigt sich schon vor der „Geburt“ der Kognitionswissenschaft am Beispiel der Carnegie-Mellon-Gruppe um Allan Newell (Informatiker)

und Herbert A. Simon (Psychologe), die das lange Zeit dominierende Berechnungsmodell, nämlich Symbolverarbeitung, gemeinsam entwickelten, aus der schließlich kognitive Architekturen wie SOAR (Newell) und ACT (Anderson) hervorgingen. Auch die charakteristische Methode des neuen Faches, die „kognitive Modellierung“ (s. Abschnitt 2.1.2.3) wäre ohne KI-Techniken schlicht nicht möglich.

Doch darin erschöpfen sich die Beziehungen nicht. Denn für die KI ist nach wie vor die menschliche Intelligenz, d.h. unsere artspezifische kognitive Leistungsfähigkeit, der gold standard, also Maßstab dessen, was die KI an technischen kognitiven Systemen entwickelt. Unser (aus Platzgründen natürlich unvollständiger und selektiver) Überblick im zweiten Teil dieses Kapitels weist allerdings aus, dass menschliche Kognition oft ganz anderen Prinzipien folgt als den in der KI üblichen. Dies hat zu beiderseitiger Befruchtung geführt: Kognitionswissenschaft und kognitive Psychologie haben viele ihrer theoretischen Modelle aus der KI-Forschung bezogen und auf dieser Grundlage Neues über die menschliche Kognition herausgefunden. Andererseits ist die KI sich der Differenz zwischen natürlicher und künstlicher Kognition bewusster geworden – zumal, seit die „leiblichen“ Voraussetzungen von Robotern wesentlich geworden sind für KI-Programme. KI „bedient“ sich also nicht einfach bei der Kognitionswissenschaft (oder umgekehrt), sondern gerade im wissenschaftlichen Austausch beider werden wir natürliche Kognition besser verstehen und zugleich erfolgreichere KI-Systeme entwickeln können. Für die Kooperation von KI-Systemen und menschlichen Partnern gilt es ohnehin, wie es die angewandte Kognitionswissenschaft anstrebt, beide Seiten zu verstehen, um ein möglichst reibungsloses Zusammenspiel zu gewährleisten.

Die Autoren danken einem anonymen Reviewer für Korrekturen und Verbesserungsvorschläge. Für die Abschnitte „Methoden“ (ohne kognitive Modellierung) und „soziale Kognition“ ist E. Ferstl verantwortlich, für den Abschnitt „Sprache“ L. Konieczny, für den Unterabschnitt „kognitive Modellierung“ und den Abschnitt „Denken und Problemlösen“ M. Ragni, und für die übrigen Abschnitte G. Strube. Wir haben gelegentlich auch Textabschnitte aus der vorigen Auflage verwendet und danken Barbara Hemforth für ihr Einverständnis, dies auch im Abschnitt „Sprache“ zu tun, den sie zusammen mit L. Konieczny verfasst hatte.

Literaturverzeichnis

- [1] Amodio, D. M., Harmon-Jones, E., Devine, P. G., Curtin, J. J., Hartley, S. L., und Covert, A. E. (2004). Neural signals for the detection of unintentional race bias. *Psychological Science*, 15:88–93.
- [2] Anderson, J. R. (1976). *Language, Memory, and Thought*. Hillsdale, NJ: Erlbaum.
- [3] Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review*, 89:369–406.
- [4] Anderson, J. R. (1983). The Architecture of Cognition.
- [5] Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* Oxford University Press, New York.
- [6] Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., und Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, 111(4):1036–1060.
- [7] Anderson, J. R. und Lebiere, C. (1998). *The atomic components of thought*. Erlbaum, Hillsdale, NJ.

- [8] Atkinson, R. C. und Shiffrin, R. M. (1968). Human memory: A proposed system and its control processes. In Spence, O. und Spence, O., editors, *Advances in the psychology of learning and motivation*, volume 2, pages 89–195. Academic Press, New York.
- [9] Baddeley, A. D. (1986). *Working memory*. Oxford University Press, Oxford.
- [10] Baddeley, A. D. (2000). The episodic buffer: A new component of working memory? *Trends in Cognitive Sciences*, 4:417–423.
- [11] Baddeley, A. D. (2002). Fractioning the central executive. In Stuss, D. T. und Knight, R. T., editors, *Principles of frontal lobe function*, pages 246–260. Oxford University Press, New York.
- [12] Baddeley, A. D. (2003). Working memory: Looking back and looking forward. *Nature Reviews Neuroscience*, 4:829–839.
- [13] Baddeley, A. D., Thomson, N., und Buchanan, M. (1975). Word length and the structure of short-term memory. *Quarterly Journal of Experimental Psychology*, 14:575–589.
- [14] Bahrick, H. P. (1984). Semantic memory content in permastore: Fifty years of memory for Spanish learned in school. *Journal of Experimental Psychology: General*, 113:1–37.
- [15] Barsalou, L. W. (1999). Perceptual symbol systems. *Behavioral and Brain Sciences*, 22:577–660.
- [16] Bates, E. A. und Elman, J. L. (2002). Connectionism and the study of change. In Johnson, M., editor, *Brain development and cognition: A reader* (2nd ed.). Blackwell Publishers, Oxford.
- [17] Beller, S. und Spada, H. (2001). Denken. In Strube, G., editor, *Wörterbuch der Kognitionswissenschaft*. Klett-Cotta-Verlag.
- [18] Biederman, I. (1981). On the semantics of a glance at a scene. In Kubovy , M. und Pomerantz, J. R., editors, *Perceptual organization*, pages 115–147. Erlbaum, Hillsdale, NJ.
- [19] Boston, M. F., Patil, U., Hale, J., Kliegl, R., und Vasishth, S. (2008). Parsing costs as predictors of reading difficulty: An evaluation using the potsdam sentence corpus. *Journal of Eye Movement Research*, 2(1):1–12.
- [20] Braine, M. D. S. und O'Brien, D. P. (1998). *Mental logic*. Erlbaum, Mahwah, NJ.
- [21] Brooks, R. (1991). Intelligence without representation. *Artificial Intelligence*, 47:139–159.
- [22] Bucciarelli, M. und Johnson-Laird, P. N. (1999). Strategies in syllogistic reasoning. *Cognitive Science: A Multidisciplinary Journal*, 23(3):247–303.
- [23] Bush, R. R. und Mosteller, F. (1951). A mathematical model for simple learning. *Psychological Review*, 58:313–323.
- [24] Byrne, R. (1989). Suppressing valid inferences with conditionals. *Cognition*, 31:61–83.
- [25] Chater, N. und Manning, C. D. (2006). Probabilistic models of language processing and acquisition. *Trends in Cognitive Sciences*, 10:335–344.
- [26] Chi, M. T. H., Bassok, M., Lewis, M. W., Reimann, P., und Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 13(2):145–182.
- [27] Christaller, T. und Metzing, D. (1983). Parsing interaction: a multilevel parser formalism based on cascaded atns. In Sparck-Jones, K. und Wilks, Y., editors, *Automatic Natural Language Parsing*. Chichester.
- [28] Christiansen, M. H. und Chater, N. (1999). Connectionist natural language processing: The state of the art. *Cognitive Science*, 23:417–437.

- [29] Chu, Y. und MacGregor, J. N. (2011). Human performance on insight problem solving: A review. *The Journal of Problem Solving*, 3(2):119–149.
- [30] Cosmides, L. (1989). The logic of social exchange: Has natural selection shaped how humans reason? studies with the wason selection task. *Cognition*, 31:187–276.
- [31] Cosmides, L. und Tooby, J. (1993). Cognitive adaptations for social exchange. In Bar-kow, J. H., Cosmides, L., und Tooby, J., editors, *The Adapted Mind: Evolutionary Psychology and the Generation of Culture*, pages 163–228. Oxford.
- [32] Crain, S. und Steedman, M. (1985). On not being led up the garden path: The use of context by the psychological syntax processor. In Dowty, D. R., Karttunen, L., und Zwicky, A. R., editors, *Natural language parsing*, pages 320–358. Cambridge University Press, Cambridge.
- [33] Dalgleish, T. und Power, M., editors (1999). *Handbook of Cognition and Emotion*. John Wiley & Sons, Hoboken, NJ.
- [34] Dennett, D. C. (1991). *Consciousness explained*. Little, Brown & Co., Boston.
- [35] Dolan, R. J. (2002). Emotion, cognition and behaviour. *Science*, 298 (5596):1191–1194.
- [36] Dörner, D., Kreuzig, H. W., Reither, F., und Stäudel, T. (1983). *Lohhausen. Vom Umgang mit Unbestimmtheit und Komplexität*. Huber.
- [37] Douven, I. (2011). Abduction. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Spring 2011 edition.
- [38] Duncker, K. (1935). *Zur Psychologie des Produktiven Denkens*. Springer, Berlin.
- [39] Ebbinghaus, E. (1885). *Über das Gedächtnis*. Duncker (Nachdruck Darmstadt: WBG, 1971), Leipzig.
- [40] Ekman, P. und Friesen, W. V. (2003). *Unmasking the face: A guide to recognizing emotions from facial expressions*. Malor Books, Los Altos, CA.
- [41] Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.
- [42] Elman, J. L. (1993). Learning and development in neural networks: the importance of starting small. *Cognition*, 48(1):71–99.
- [43] Ericsson, K. A., Chase, W. G., und Faloon, S. (1980). Acquisition of a memory skill. *Science*, 208:1181–1182.
- [44] Erman, L. D., Hayes-Roth, F., Lesser, V. R., und Reddy, D. R. (1980). The Hearsay II speech understanding system: integrating knowledge to resolve uncertainty. *Computing Surveys*, 12.
- [45] Evans, J. S. B. T., Newstead, S. E., und Byrne, R. M. J. (1993). *Human reasoning: The psychology of deduction*. Lawrence Erlbaum Associates, London.
- [46] Falkenhainer, B., Forbus, K. D., und Gentner, D. (1986). *The structure-mapping engine*. Report (University of Illinois at Urbana-Champaign. Dept. of Computer Science). Dept. of Computer Science, University of Illinois at Urbana-Champaign.
- [47] Felleman, D. J. und van Essen, D. C. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1:1–47.
- [48] Ferstl, E. C. (2006). Theory-of-Mind und Kommunikation: Zwei Seiten der gleichen Medaille? In Förstl, H., editor, *Theory of Mind: Neurobiologie und Psychologie sozialen Verhaltens*, pages 67–78. Springer, Heidelberg.
- [49] Ferstl, E. C., Neumann, J., Bogler, C., und von Cramon, D. Y. (2008). The extended language network: A meta-analysis of neuroimaging studies on text comprehension. *Human Brain Mapping*, 29:581–593.

- [50] Ferstl, E. C. und von Cramon, D. Y. (2002). What does the fronto-median cortex contribute to language processing: Coherence or theory of mind? *NeuroImage*, 17:1599–1612.
- [51] Fiedler, K. (1988). The dependence of the conjunction fallacy on subtle linguistic factors. *Psychological Research*, 50:123–129. 10.1007/BF00309212.
- [52] Fiske, S. T. und Taylor, S. T. (2007). *Social Cognition: From Brains to Culture*. McGraw-Hill, New York.
- [53] Fitts, P. M. (1964). Perceptual–motor skill learning. In Melton, A. W., editor, *Categories of human learning*, pages 234–285. Academic Press, New York.
- [54] Fodor, J. A. (1975). *The language of thought*. Crowell, New York.
- [55] Fodor, J. A. (1983). *The modularity of mind*. MIT Press, Cambridge, MA.
- [56] Förstl, H., editor (2012). *Theory of Mind: Neurobiologie und Psychologie sozialen Verhaltens* (2. Auflage). Springer, Heidelberg.
- [57] Frazier, L. (1987a). Sentence processing: a tutorial review. In Coltheart, M., editor, *The psychology of reading*. Lawrence Erlbaum, Hove.
- [58] Frazier, L. (1987b). Theories of sentence processing. In Garfield, J. L., editor, *Modularity in knowledge representation and natural-language processing*, pages 293–309. MIT Press, Cambridge, MA.
- [59] Frazier, L. (1999). *On sentence interpretation*. Kluwer Academic Press, Dordrecht.
- [60] Frensch, P. A. und Funke, J., editors (1995). *Complex problem solving: the European perspective*. Lawrence Erlbaum, Hillsdale, NJ.
- [61] Friederici, A. D. und Mecklinger, A. (1996). Syntactic parsing as revealed by brain responses: First-pass and second-pass parsing processes. *Journal of Psycholinguistic Research*, 25(1):157–176.
- [62] Funke, J. (2006). *Lösen komplexer Probleme*. Handbuch der Psychologie. Hogrefe.
- [63] Gallagher, S. und Zahavi, D. (2008). *The phenomenological mind*. Routledge, London, New York.
- [64] Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Psychology*, 7:155–170.
- [65] Gentner, D., Holyoak, K. J., und Kokinov, B. N. (2001). *The analogical mind: perspectives from cognitive science*. Bradford Books. MIT Press.
- [66] Gentner, D. und Markman, A. B. (1995). Analogy-based reasoning. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, pages 91–93. MIT Press, Cambridge, MA.
- [67] Gibson, E. (2000). The dependency locality theory: A distance-based theory of linguistic complexity. In Miyashita, Y., Marantz, A., und O’Neil, W., editors, *Image, language, brain*, pages 95–126. MIT Press, Cambridge, MA.
- [68] Gigerenzer, G. und Todd, P. (1999). *Simple heuristics that make us smart*. Oxford University Press, New York.
- [69] Gluck, M. A., Mercado, E., und Myers, C. E. (2010). *Lernen und Gedächtnis*. Spektrum, Heidelberg.
- [70] Godden, D. R. und Baddeley, A. D. (1975). Context-dependent memory in two natural environments: On land and underwater. *British Journal of Psychology*, 66:325–331.
- [71] Goldenberg, G. (2007). *Neuropsychologie: Grundlagen, Klinik, Rehabilitation*. Urban & Fischer.

- [72] Gordon, P. C., Hendrick, R., und Johnson, M. (2001). Memory interference during language processing. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 27:1411–1423.
- [73] Greenwald, A. G., McGhee, D. E., und Schwartz, J. L. K. (1998). Measuring individual differences in implicit cognition: The implicit association test. *Journal of Personality and Social Psychology*, 74:1464–1480.
- [74] Grice, H. P. (1975). William James Lectures, Harvard University, 1967. In Cole, P. und Morgan, J. L., editors, *Syntax and semantics*, volume 3. Seminar Press, New York.
- [75] Griffiths, T. L., Kemp, C., Perfors, A., und Tenenbaum, J. B. (2010). Probabilistic models of cognition: Exploring the laws of thought. *Trends in Cognitive Sciences*, 14(8):357–364.
- [76] Griffiths, T. L. und Tenenbaum, J. B. (2005). Structure and strength in causal induction. *Cognitive Psychology*, 51:354–384.
- [77] Griffiths, T. L. und Tenenbaum, J. B. (2007). From mere coincidences to meaningful discoveries. *Cognition*, 103(2):180–226.
- [78] Hagmayer, Y. und Waldmann, M. R. (2006). Kausales Denken. In Funke, J., editor, *Enzyklopädie der Psychologie ‚Denken und Problemlösen‘*, number 8 in C, chapter II, pages 87–166. Hogrefe, Göttingen.
- [79] Hagoort, P., Brown, C., und Groothusen, J. (1993). The syntactic positive shift (SPS) as an ERP measure of syntactic processing. *Language and Cognitive Processes*, 8(4):439–483.
- [80] Halbmayer, E. und Salat, J. (2012). Qualitative Methoden der Kultur- und Sozialanthropologie. <http://www.univie.ac.at/ksa/elearning/cp/qualitative/qualitative-5.html>.
- [81] Hale, J. (2003). The information conveyed by words in sentences. *Journal of Psycholinguistic Research*, 32(2):101–123.
- [82] Hale, J. (2006). Uncertainty about the rest of the sentence. *Cognitive Science*, 30(4):609–642.
- [83] Hale, J. (2011). What a rational parser would do. *Cognitive Science*, 35(3):399–443.
- [84] Harnad, S. (1990). The symbol grounding problem. *Physica*, D 42:335–346.
- [85] Hebb, D. O. (1949). *The Organization of Behavior*. John Wiley, New York.
- [86] Hélie, S. und Sun, R. (2010). Incubation, insight, and creative problem solving: A unified theory and a connectionist model. *Psychological Review*, 117(3):994–1024.
- [87] Helson, H. (1964). *Adaptation-level theory*. Harper & Row, New York.
- [88] Hemforth, B. (1993). *Kognitives Parsing: Repräsentation und Verarbeitung sprachlichen Wissens*. Infix, Sankt Augustin.
- [89] Hertwig, R. und Gigerenzer, G. (1999). The ‘conjunction fallacy’ revisited: How intelligent inferences look like reasoning errors. *Journal of Behavioral Decision Making*, 12:275–305.
- [90] Hewitt, C. (1977). Viewing control structures as patterns of passing messages. *Artif. Intell.*, 8(3):323–364.
- [91] Hörmann, H. (1976). *Meinen und Verstehen. Grundzüge einer psychologischen Semantik*. Suhrkamp, Frankfurt.
- [92] Johnson-Laird, P. N. (1972). The three-term series problem. *Cognition*, 1:57–82.

- [93] Johnson-Laird, P. N. (1983). *Mental models: Towards a cognitive science of language, inference and consciousness*. Harvard University Press, Cambridge, MA.
- [94] Johnson-Laird, P. N. (2001). Mental models and deduction. *Trends in Cognitive Sciences*, 5:434–442.
- [95] Johnson-Laird, P. N. (2006). *How we reason*. Oxford University Press, New York.
- [96] Johnson-Laird, P. N., Legrenzi, P., Girotto, V., Legrenzi, M. S., und Caverni, J. P. (1999). Naive probability: a mental model theory of extensional reasoning. *Psychological Review*, 106(1):62–88.
- [97] Jones, M. und Love, B. C. (2011). Bayesian fundamentalism or enlightenment? on the explanatory status and theoretical contributions of bayesian models of cognition. *Behavioral and Brain Sciences*, 34:169–231.
- [98] Juliano, C. und Tanenhaus, M. K. (1994). A constraint-based lexicalist account of the subject/object attachment preference. *Journal of Psycholinguistic Research*, 23:459–471.
- [99] Jurafsky, D. (1996). A probabilistic model of lexical and syntactic access and disambiguation. *Cognitive Science*, 20:137–194.
- [100] Just, M. A. und Carpenter, P. A. (1992). A capacity theory of comprehension: Individual differences in working memory. *Psychological Review*, 99:122–149.
- [101] Kahneman, D. (2003). A perspective on judgement and choice. *American Psychologist*, 58:697–720.
- [102] Kahneman, D., Slovic, P., und Tversky, A. (1982). *Judgment Under Uncertainty: Heuristics and Biases*. Cambridge University Press.
- [103] Karnath, H.-O., Ferber, S., und Himmelbach, M. (2001). Spatial awareness is a function of the temporal not the posterior parietal lobe. *Nature*, 411:950–953.
- [104] Kemmerling, A. (1990). Mentale Repräsentationen. *Kognitionswissenschaft*, 1:71–82.
- [105] Kennedy, A. (2003). The dundee corpus [cd-rom]. Psychology Department, The University of Dundee.
- [106] King, J. und Just, M. A. (1991). Individual differences in syntactic processing: The role of working memory. *Journal of Memory & Language*, 30:580–602.
- [107] Kischka, U., Wolf, G., und Wallesch, C.-W. (1997). *Methoden der Hirnforschung: Eine Einführung*. Spektrum, Heidelberg.
- [108] Konieczny, L. (2000). Locality and parsing complexity. *Journal of Psycholinguistic Research*, 29(6):627–645.
- [109] Konieczny, L. (2005). The psychological reality of local coherences in sentence processing. In *Proceedings of the 27th Annual Conference of the Cognitive Science Society*, pages 1178–1183.
- [110] Konieczny, L. und Müller, D. (2010). Das sprachliche Arbeitsgedächtnis: Kapazität oder Erfahrung? *Psychologische Rundschau*, 61(1):43–50.
- [111] Konieczny, L., Müller, D., Hachmann, W., Schwarzkopf, S., und Wolfer, S. A. (2009a). Local syntactic coherence interpretation. evidence from a visual world study. In Taatgen, N. und van Rijn, H., editors, *Proceedings of the the 31st annual conference of the cognitive science society*, pages 1133–1138, Austin, TX.
- [112] Konieczny, L., Ruh, N., und Müller, D. (2009b). What's in an error? a closer look at SRNs processing relative clauses. In Howes, A., Peebles, D., und Cooper, R. P., editors, *Proceedings of the the 9th International Conference on Cognitive Modeling*, Manchester, UK.

- [113] Laird, J. E. (2008). Extending the soar cognitive architecture. In *AGI*, pages 224–235.
- [114] Laird, J. E. (2012). *The Soar Cognitive Architecture*. MIT Press.
- [115] Laird, J. E., Newell, A., und Rosenbloom, P. S. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64.
- [116] Langley, P., Laird, J. E., und Rogers, S. (2009). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2):141–160.
- [117] LeDoux, J. E. (1992). In search of an emotional system in the brain: leaping from emotion and the amygdala. In Aggleton, J. P., editor, *The amygdala: neurobiological aspects of emotion, memory, and mental dysfunction*, pages 339–351. Wiley–Liss, New York.
- [118] Levy, R. (2008). Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.
- [119] Logothetis, N. K. (2008). What we can and cannot do with fMRI. *Nature*, 453:869–878.
- [120] MacDonald, M. und Christiansen, M. (2002). Reassessing working memory: A comment on Just & Carpenter (1992) and Waters & Caplan (1996). *Psychological Review*, 109:33–54.
- [121] MacDonald, M. C., Pearlmuter, N. J., und Seidenberg, M. S. (1994). The lexical nature of syntactic ambiguity resolution. *Psychological Review*, 101:676–703.
- [122] MacWhinney, B. (1994). The dinosaurs and the ring. In Corrigan, R., Lima, S., und Noonan, M., editors, *The reality of linguistic rules*, page 283–320. John Benjamins.
- [123] Maes, P. (1994). *Artificial Life Journal*, chapter Modeling Adaptive Autonomous Agents. MIT Press.
- [124] Mallot, H. A. (2000). *Computational vision: Information processing in perception and visual behavior*. MIT Press, Cambridge, MA.
- [125] Markman, A. B. (1999). *Knowledge representation*. L. Erlbaum.
- [126] Markowitsch, H. J. (2012). Neuroanatomie und Störungen des Gedächtnisses. In Karthäfer, H.-O. und Thier, P., editors, *Kognitive Neurowissenschaften*, pages 553–566. Springer, Berlin, 3. akt. aufl. edition.
- [127] Marr, D. (1982). *Vision. A computational investigation into the human representation and processing of visual information*. Freeman, San Francisco.
- [128] Marslen-Wilson, W. D. und Tyler, L. K. (1980). The temporal structure of spoken language understanding. *Cognition*, 8:1–71.
- [129] Marslen-Wilson, W. D. und Tyler, L. K. (1987). Against modularity. In Garfield, J., editor, *Modularity in knowledge representation and natural language understanding*. MIT Press, Cambridge, MA.
- [130] Martin, R. C. (2006). The neuropsychology of sentence processing: Where do we stand? *Cognitive Neuropsychology*, 23:74–95.
- [131] McClamrock, R. (1991). Marr’s three levels: A re-evaluation. *Minds and Machines*, 1(2):185–196.
- [132] McClelland, J. L. (1988). *Explorations in parallel distributed processing: a handbook of models, programs, and exercises*. MIT Press.
- [133] McCulloch, W. S. und Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.
- [134] Metzinger, T. (1993). *Subjekt und Selbstmodell*. Schöningh, Paderborn.
- [135] Meyer, D. E. und Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple-task performance: Part 1. basic mechanisms. *Psychological Review*, 104:3–65.

- [136] Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–96.
- [137] Minsky, M. und Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA.
- [138] Mishkin, M., Ungerleider, L. G., und Macko, K. A. (1983). Object vision and spatial vision: Two central pathways. *Trends in Neuroscience*, 6:414–417.
- [139] Mitchell, D. C. (1994). Sentence parsing. In Gernsbacher, M. A., editor, *Handbook of psycholinguistics*, pages 375–409. Academic Press, San Diego.
- [140] Miyake, A., Friedman, N. P., Emerson, M. J., Witzki, A. H., Howerter, A., und Wagner, T. D. (2000). Fractionating the central executive: Evidence for separability of executive funktions. *Cognitive Psychology*, 41:49–100.
- [141] Müller, G. E. und Pilzecker, A. (1900). Experimentelle Beiträge zur Lehre vom Gedächtnis. *Zeitschrift für Psychologie*, Suppl. 1.
- [142] Müller, J. P. (1996). *The Design of Intelligent Agents - A Layered Approach*, volume 1177 of *Lecture Notes in Computer Science*. Springer.
- [143] Neisser, U. (1976). *Cognition and reality*. W.H. Freeman, San Francisco.
- [144] Newell, A. (1990). *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA.
- [145] Newell, A. und Simon, H. (1972). *Human problem solving*. Prentice-Hall, Englewoods Cliffs, NJ.
- [146] Newell, A. und Simon, H. A. (1976). Computer science as empirical enquiry: Symbols and search. *Communications of the ACM*, 19:113–126.
- [147] Newell, A. und Simon, H. A. (1976). Computer science as empirical enquiry: symbols and search. *Communications of the ACM*, 19:113–126.
- [148] Oaksford, M. und Chater, N. (2001). The probabilistic approach to human reasoning. *Trends in Cognitive Sciences*, 5:349–357.
- [149] Oaksford, M. und Chater, N. (2007). *Bayesian rationality: The probabilistic approach to human reasoning*. Oxford University Press, Oxford.
- [150] Ochsner, K. N. und Lieberman, M. D. (2001). The emergence of social cognitive neuroscience. *American Psychologist*, 56:717–734.
- [151] Ortony, A. und Turner, T. J. (1990). What's basic about basic emotions? *Psychological Review*, 81:181–192.
- [152] Osterhout, L. und Holcomb, P. J. (1992). Event-related brain potentials elicited by syntactic anomaly. *Journal of Memory and Language*, 31:785–806.
- [153] Palmer, S. E. (1999). *Vision science – photons to phenomenology*. MIT Press, Cambridge, MA.
- [154] Payne, B. K. (2001). Prejudice and perception: The role of automatic and controlled processes in misperceiving a weapon. *Journal of Personality and Social Psychology*, 81:181–192.
- [155] Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- [156] Pearl, J. (2000). *Causality: Models, Reasoning, and Inference*. Cambridge U.P.
- [157] Pezzuolo, G., Barsalou, L. W., Cangelosi, A., Fischer, M. H., McRae, K., und Spivey, M. J. (2011). The mechanics of embodiment: a dialog on embodiment and computational modeling. *Frontiers in Psychology*, 2 (5):1–21.
- [158] Piaget, J. (1944). *Die geistige Entwicklung des Kindes*. Zürich.

- [159] Pickering, M. J. und Garrod, S. (2004). Toward a mechanistic psychology of dialogue. *Behavioral Brain Sciences*, 27:169–225.
- [160] Poldrack, R. A. (2006). Can cognitive processes be inferred from neuroimaging data? *Trends in Cognitive Science*, 10:59–63.
- [161] Prinz, W. (1983). *Wahrnehmung und Tätigkeitssteuerung*. Springer, Berlin.
- [162] Putnam, H. (1979). *Mind, Language and Reality*. Philosophical Papers. Cambridge University Press.
- [163] Putz-Osterloh, W. (1981). Über die Beziehung zwischen Testintelligenz und Problemlöseerfolg. *Zeitschrift für Psychologie*, 189:79–100.
- [164] Pylyshyn, Z. (1984). *Computation and cognition*. MIT Press, Cambridge, MA.
- [165] Ragni, M. (2008). *Räumliche Repräsentation, Komplexität und Deduktion: Eine kognitive Komplexitätstheorie*[*Spatial representation, complexity and deduction: A cognitive theory of complexity*]. PhD thesis, Albert-Ludwigs-Universität Freiburg.
- [166] Rauh, R., Hagen, C., Knauff, M., Kuß, T., Schlieder, C., und Strube, G. (2005). Preferred and alternative mental models in spatial reasoning. *Spatial Cognition and Computation*, 5:239–269.
- [167] Rayner, K. (1998). Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, 124:372–422.
- [168] Rayner, K. und Sereno, S. (1994). Eye movements in reading: Psycholinguistic studies. In Gernsbacher, M. A., editor, *Handbook of psycholinguistics*, pages 57–82. Academic Press, San Diego.
- [169] Reason, J. (1990). *Human error*. Cambridge University Press, Cambridge.
- [170] Reichel, W. (2005). *Der große Intelligenztest: IQ-+ EQ-Test-Training mit mehr als 600 Fragen und Antworten*. Klett-Cotta.
- [171] Rescorla, R. A. und Wagner, A. R. (1972). A theory of pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In Black, A. H. und Prokasy, W. F., editors, *Classical conditioning II*. Appleton–Century–Crofts, New York.
- [172] Riesenhuber, M. und Poggio, T. (2002). Neural mechanisms of object recognition. *Current Opinion in Neurobiology*, 12:162–168.
- [173] Rips, L. J. (1994). *The psychology of proof: Deductive reasoning in human thinking*. The MIT Press, Cambridge, MA.
- [174] Rizzolatti, G., Fadiga, L., Gallese, V., und Fogassi, L. (1996). Premotor cortex and the recognition of motor actions. *Cognitive Brain Research*, 3:131–141.
- [175] Rosch, E. H. (1973). Natural categories. *Cognitive Psychology*, 4:328–350.
- [176] Rösler (2011). *Psychophysiologie der Kognition: Eine Einführung in die kognitive Neurowissenschaft*. Spektrum Akademischer Verlag, Heidelberg.
- [177] Russell, S. und Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition.
- [178] Sager, S., Barth, C. M., Diedam, H., Engelhart, M., und Funke, J. (2011). Optimization as an analysis tool for human complex problem solving. *SIAM Journal on Optimization*, 21(3):936–959.
- [179] Schank, R. C. und Abelson, R. P. (1977). *Scripts, Plans, Goals and Understanding*. Erlbaum, Hillsdale, NJ.
- [180] Searle, J. (1983). *Intentionality: an essay in the philosophy of mind*. Cambridge University Press, Cambridge.
- [181] Searle, J. R. (1969). *Speech acts: an essay in the philosophy of language*. Cambridge University Press, Cambridge.

- [182] Searle, J. R. (1980). Minds, brains, and programs. *Behavioral and Brain Sciences*, 3:417–457.
- [183] Shepard, R. N. und Metzler, J. (1971). Mental rotation of three-dimensional objects. *Science*, 171:701–703.
- [184] Simon, H. und Wallach, D. (1999). Cognitive modeling in perspective. *Kognitionswissenschaft*, 8:1–4.
- [185] Simon, H. A. und Wallach, D. (1999). editorial: Cognitive modeling in perspective. *Kognitionswissenschaft*, 8:1–4.
- [186] Simons, D. J. und Chabris, C. F. (1999). Gorillas in our midst: sustained inattentional blindness for dynamic events. *Perception*, 28:1059–1074.
- [187] Singer, W. (1999). Neuronal synchrony: A versatile code for the definition of relations? *Neuron*, 24:49–65.
- [188] Skinner, B. F. (1938). *The behavior of organisms: An experimental analysis*. Appleton Century-Crofts, New York.
- [189] Smith, E. E. und Kosslyn, S. M. (2007). *Cognitive Psychology*. Pearson Prentice Hall, Upper Saddle River, NJ.
- [190] Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11:1–74.
- [191] Sperling, G. (1960). The information available in brief visual presentations. *Psychological Monographs: General and Applied*, 74:1–29.
- [192] Steels, L. (2008). The symbol grounding problem has been solved. So what's next? In de Vega, M., Glenberg, A., und Graesser, A., editors, *Symbols and Embodiment: Debates on Meaning and Cognition*, pages 223–244. Oxford University Press, New York.
- [193] Stenning, K. und Lambalgen, M. (2008). *Human reasoning and cognitive science*. Bradford Books. MIT Press.
- [194] Steyvers, M., Griffiths, T. L., und Dennis, S. (2006). Probabilistic inference in human semantic memory. *Trends in Cognitive Sciences*, 10:327–334.
- [195] Steyvers, M., Tenenbaum, J. B., Wagenmakers, E. J., und Blum, B. (2003). Inferring causal networks from observations and interventions. *Cognitive Science*, 27(3):453–489.
- [196] Stroop, J. R. (1935). Studies of interference in serial verbal reactions. *Journal of Experimental Psychology*, 18:643–662.
- [197] Strube, G. (1992). The role of cognitive science in knowledge engineering. In Schmalhofer, F., Strube, G., und Wetter, T., editors, *Contemporary knowledge engineering and cognition*, pages 161–175. Springer, Berlin, Heidelberg, New York.
- [198] Strube, G. (2001). Cognitive modeling: research logic in cognitive science. In Smelser, N. J. und Baltes, P. B., editors, *International encyclopedia of the social and behavioral sciences*, pages 2124–2128. Elsevier Science, Oxford.
- [199] Suchman, L. A. (1987). *Plans and situated actions: The problem of human-machine communication*. Cambridge University Press, New York.
- [200] Sun, R. (2001). *Duality of the mind – a bottom-up approach toward cognition*. Lawrence Erlbaum.
- [201] Sun, R. (2006). *Cognition And Multi-agent Interaction: From Cognitive Modeling To Social Simulation*. Cambridge University Press.
- [202] Taatgen, N. und Anderson, J. R. (2010). The past, present, and future of cognitive architectures. *Topics in Cognitive Science*, 2(4):693–704.
- [203] Tabor, W., Galantucci, B., und Richardson, D. (2004). Effects of merely local syntactic coherence on sentence processing. *Journal of Memory and Language*, 50(4):355–370.

- [204] Tabor, W., Juliano, C., und Tanenhaus, M. K. (1997). Parsing in a dynamical system: An attractor-based account of the interaction of lexical and structural constraints in sentence processing. *Language and Cognitive Processes*, 12:211–271.
- [205] Tabor, W. und Traugott, E. (1998). Structural scope expansion and grammaticalization. In Ramat, A. G. und Hopper, P. J., editors, *The Limits of Grammaticalization*, pages 229–272. John Benjamins, Amsterdam.
- [206] Tack, W. H. (1997). Kognitionswissenschaft: eine Interdisziplin. *Kognitionswissenschaft*, 6:2–8.
- [207] Tambe, M., Johnson, W. L., Jones, R. M., Koss, F. V., Laird, J. E., Rosenbloom, P. S., und Schwamb, K. (1995). Intelligent agents for interactive simulation environments. *AI Magazine*, 16(1):15–39.
- [208] Tenenbaum, J. B., Griffiths, T. L., und Kemp, C. (2006). Theory-based Bayesian models of inductive learning and reasoning. *Trends in Cognitive Science*, 10:309–318.
- [209] Tenenbaum, J. B., Kemp, C., Griffiths, T. L., und Goodman, N. D. (2011). How to Grow a Mind: Statistics, Structure, and Abstraction. *Science*, 331(6022):1279–1285.
- [210] Thibadeau, R., Just, M. A., und Carpenter, P. (1982). A model of the time course and content of reading. *Cognitive Science*, 6:157–203.
- [211] Tomasello, M. (2003). *Constructing a language*. Harvard Univ. Press, Cambridge, MA.
- [212] Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265.
- [213] Tversky, A. und Kahneman, D. (1983). Extensional Versus Intuitive Reasoning: The Conjunction Fallacy in Probability Judgment. *Psychological Review*, 90(4):293–315.
- [214] Van Dyke, J. A. (2007). Interference effects from grammatically unavailable constituents during sentence processing. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, 33(2):407–430.
- [215] Vasishth, S. und Lewis, R. L. (2006). Argument-head distance and processing complexity: Explaining both locality and antilocality effects. *Language*, 82(4):767–794.
- [216] Ward, J. (2010). *The Student's Guide to Cognitive Neuroscience* (2nd ed.). Psychology Press, Hove, UK.
- [217] Ward, J. (2012). *The Student's Guide to Social Neuroscience*. Psychology Press, Hove, UK.
- [218] Wason, P. (1968). Reasoning about a rule. *Quarterly Journal of Experimental Psychology*, 20(3):273–281.
- [219] Watson, J. B. und Rayner, R. (1920). Conditioned emotional reactions. *Journal of Experimental Psychology*, 3:1–15.
- [220] Wells, J., Christiansen, M. H., Race, D. S., Acheson, D., und MacDonald, M. C. (2009). Experience and sentence processing: Statistical learning and relative clause comprehension. *Cognitive Psychology*, 58:250–271.
- [221] Wilson, M. (2002). Six views of embodied cognition. *Psychonomic Bulletin & Review*, 9 (4):626–636.
- [222] Wittgenstein, L. (1922). *Tractatus logico–philosophicus*. Routledge & Kegan Paul, London.
- [223] Xu, F. und Tenenbaum, J. B. (2007). Word learning as bayesian inference. *Psychological Review*, 114(2):245–72.
- [224] Zwaan, R., Stanfield, R. A., und Yaxley, R. H. (2002). Language comprehenders mentally represent the shape of objects. *Psychological Science*, 13:169–171.

3 Suche

Clemens Beckstein

3.1 Problemlösen als Suche

3.1.1 Problemrepräsentation mit Zuständen und Operatoren

Viele Probleme der Künstlichen Intelligenz lassen sich als Probleme reformulieren, bei denen Suche eine Rolle spielt. Deshalb sollte jeder, der sich näher mit dem Gebiet der Künstlichen Intelligenz befaßt, die wichtigsten Suchverfahren und deren Eigenschaften kennen. Das entsprechende Grundwissen versuchen die folgenden Ausführungen zu vermitteln.

Bevor man ein Suchverfahren zur Lösung eines Problems einsetzen kann, muß dieses Problem in eine Form gebracht werden, die es einer Verarbeitung durch Suchverfahren zugänglich macht. Eine solche Form ist eine (i. allg. nicht eindeutige) *Zustandsraumrepräsentation* des Problems. Zu ihrer Herstellung muß man den Problemlöseprozeß zustandsorientiert rekonstruieren, d.h.

- (*Problemlöse-)**Zustände* identifizieren, die den Fortschritt der Problemlösung zu ausgewählten Zeitpunkten darstellen und
- dazu passend *Zustandsübergangsoperatoren* festlegen, die elementare Problemlöseschritte darstellen, also beschreiben, wie man von einem ausgewählten Problemlösezustand zu seinen unmittelbaren Nachfolgern gelangen kann.

Außerdem hat man bestimmte Problemlösezustände besonders kennzuzeichnen: die Zustände, von denen aus die Problemlösung starten kann, als Startzustände und jene Zustände, in denen das Problem als gelöst betrachtet werden kann, als Zielzustände. Von den Operatoren wird üblicherweise und auch im Folgenden angenommen, dass es derer nur endlich viele gibt (in der Praxis stellt das keine Einschränkung dar) und dass deren Ausführung mit i. allg. operatorspezifischen (positiven) Kosten verbunden ist. Für Szenarien, bei denen die Kosten der einzelnen Operatoren für die Problemlösung irrelevant sind, setzt man der einheitlichen Repräsentation halber für alle Operatoren gleiche (Einheits-)Kosten an.

Hat man erst einmal für ein Problem eine derartige Zustandsraumrepräsentation gefunden, weiß man, dass das Problem eine Lösung besitzt, sofern in diesem Zustandsraum einer der Startzustand mit Hilfe der verfügbaren Operatoren in einen Zielzustand überführt werden kann (für eine erschöpfende Diskussion dieser Sicht von Problemlösung siehe z.B. [20] und [16]), d.h. falls sich (mindestens) eine Folge von hintereinander ausführbaren Operatoren finden lässt, die im Endeffekt genau so eine Zustandstransformation bewirkt. Entsprechende Operatorfolgen können daher als Plan zur Lösung des Problems betrachtet werden und das ursprünglich zu lösende Problem wird damit auf eines der Suche nach entsprechenden Pfaden im Zustandsraum zurückgeführt.

Damit diese Suche auch durch einen Rechner durchgeführt werden kann, der ja keine Vorstellung von den konkreten Zuständen und Operatoren hat, relativ zu denen in der realen Welt das Problem gelöst werden soll, muß dieser Zustandsraum zunächst geeignet im Rechner repräsentiert werden. Dazu liegt es nahe, den Zustandsraum im Rechner durch einen markierten Graphen zu repräsentieren, in dem die Knoten für Zustände und gerichtete Kanten für Übergangsooperatoren des Zustandsraumes stehen. Knoten, die Start- bzw. Zielzustände repräsentieren, sind dabei entsprechend als Start- bzw. Zielknoten zu kennzeichnen und die Kanten, neben dem Namen des Operators, für den sie jeweils stehen, auch mit dessen Kosten zu annotieren¹. Jeder Pfad von einem Startknoten zu einem der Zielknoten repräsentiert damit einen Lösungspfad im Zustandsraum, also eine Lösung des Problems. Aufgabe der im Folgenden beschriebenen Suchverfahren ist es, solche Pfade unter einem möglichst geringen Einsatz von Ressourcen zu finden.

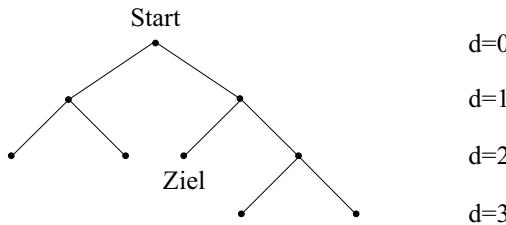


Abbildung 3.1: Grundbegriffe der Suche.

In Abbildung 3.1 ist ein derartiger *Suchgraph* (genauer ein *Suchbaum*) dargestellt – wie in der Informatik üblich, auf dem Kopf, d.h. mit der Wurzel des Baums am oberen und den Blättern am unteren Ende. Die Kanten sind demnach immer als von oben nach unten gerichtet zu sehen. In der Abbildung ist die Wurzel als Startknoten und eines der Blätter als Zielknoten gekennzeichnet. Die restlichen Blätter stellen Sackgassen dar, von denen aus mit den vorhandenen Operatoren (deren Kosten hier nicht interessieren) keine Lösung mehr erzielbar ist. Die einzelnen Schichten des Baums – eine Schicht enthält jeweils Knoten gleicher *Tiefe* d , d.h. mit gleich langen Pfaden von der Wurzel zum fraglichen Knoten – sind auf der rechten Seite der Abbildung mit $d = 0, \dots, 3$ durchnumeriert. Im dargestellten Graphen hat jeder Knoten einen *Verzweigungsgrad* von 2, d.h. genau 2 Nachfolger.

Es ist im Allgemeinen nicht ratsam, das Suchprogramm den kompletten Suchraum in einer Datenstruktur aufzubauen zu lassen, bevor es mit der eigentlichen Suche beginnt. Selbst wenn der entsprechende Suchgraph endlich ist, wird er nämlich häufig trotzdem sehr groß sein und damit seine vollständige Erzeugung im Rechner die Problemlösung bereits vor der eigentlichen Suche unverhältnismäßig verteuern: Lösungen, falls sie existieren, erstrecken sich ja i. allg. nur über einen sehr kleinen Teil dieses Graphen. Statt dessen wird man das Suchverfahren mit einer Prozedur parametrisieren, die zu einem vorgegebenen Suchraumknoten *der Reihe nach* all dessen Nachfolger ermitteln kann (also die *Übergangsfunktion* zum Suchgraphen berechnet) und dafür sorgen, dass das Verfahren den Suchraum schrittweise und nur so weit generiert, wie es für den Fortgang der Suche unbedingt nötig ist. Dabei entsteht dann intern im einfachsten Fall ein Baum, dessen Wurzel der Startknoten ist (oder ein Wald solcher Bäume, falls es mehr als einen Startknoten gibt) und dessen Wurzelpfade zusammengenommen den bisher erforschten Teil des Suchraums konstituieren.

¹ Im Falle von Einheitskosten spart man sich die Kostenangabe.

Um die Übergangsfunktion kodieren zu können, wird man die Knoten des Suchraums mit Datenstrukturen markieren, die den Zustand, für den sie jeweils stehen, genauer charakterisieren und darüberhinaus reichhaltig genug sind, um passend zu den Operatoren zu den Kanten, die aus einem Knoten herausführen, die Beschreibungen der Zustände zu den Folgeknoten (und damit die Folgeknoten selbst) berechnen zu können. Außerdem wird man das Suchprogramm in der Regel noch mit Instrumenten versorgen, die eine genauere Fokussierung des Suchprozesses erlauben. Dazu gehören einerseits Funktionen, die die Knoten des Suchraums entsprechend den von ihnen repräsentierten Zuständen bewerten – z.B. als besonders interessante und damit vorrangig zu explorierende Knoten – und andererseits solche, die die Kosten abschätzen, die nach dem momentanen Stand der Suche mindestens schon und höchstens noch für eine Lösung fällig werden können (dazu gleich mehr). Zur Durchführung der Knotenbewertungen wird das Suchverfahren auf die zustandscharakterisierenden Informationen zurückgreifen, mit denen die Knoten markiert sind und für die genannten Abschätzungen der Kosten potentieller Lösungen auf die Markierungen der Kanten, die zu diesen Lösungen nach momentanem Kenntnisstand beitragen.

Die Start- bzw. Zielknoten werden üblicherweise dadurch als solche gekennzeichnet, dass man in ihren Beschreibungen algorithmisch überprüfbare Eigenschaften aufführt, die sie entsprechend charakterisieren. Im einfachsten Fall legt man dazu die Mengen der Start- bzw. Zielknoten durch eine explizite (also insbesondere endliche) Auflistung ihrer jeweiligen Elemente fest und versieht die betroffenen Knoten mit einem entsprechenden Flag. Derart kodierte Zustandsmengen bezeichnen wir im Folgenden als *explizit vorgegeben* und Zustandsmengen, die nicht auf diese Weise festgelegt oder festlegbar sind, als (lediglich) *implizit vorgegeben*.²

Beispiel 3.1.1. Schiebepuzzle mit 8 Steinen (für eine formale Betrachtung siehe [4]):

Bei diesem Suchproblem sind 8 horizontal und vertikal bewegliche Steine zusammen mit einem Leerraum in einer ebenen 3×3 Matrix angeordnet. Ein Stein, der zum Leerraum benachbart ist, kann auf die Stelle des Leerraums verschoben werden (dabei wandert natürlich der Leerraum an die alte Position des Steines). Die Aufgabe besteht darin, eine vorgegebene Ausgangskonfiguration (etwa die linke in der Abbildung 3.2) durch Steinbewegungen der genannten Art in die Zielkonfiguration zu überführen, die auf der rechten Seite von Abbildung 3.2 gezeigt ist (Anordnung der Steine gemäß ihrer Numerierung im Uhrzeigersinn, beginnend links oben mit Stein 1).



Abbildung 3.2: Das Schiebepuzzle.

In jeder Konfiguration gibt es mindestens zwei und höchstens vier Möglichkeiten, einen Stein zu verschieben. Unter der Annahme einer durchschnittlichen Anzahl von drei Nachfolgern pro Konfiguration und einer typischen Lösungsweglänge von 20, also eines uniformen Suchbaums des Verzweigungsgrads 3 und der Tiefe 20, erhält man einen Baum mit fast 3,5 Milliarden Knoten.

² Bei einer lediglich implizit vorgegebenen Zielzustandsmenge ist übrigens die Aufzählung auch nur eines ihrer Elemente durch ein Programm im ungünstigsten Fall genauso aufwendig, wie die Lösung des Suchproblems selbst.

Die Übergangsfunktion des Zustandsraums zum Schiebepuzzle kann übrigens einfacher angegeben werden, wenn man nicht Bewegungen von Steinen, sondern Verschiebungen des Leerraums als die Operatoren ansieht, die den Zustandsraum aufspannen: Die möglichen Nachfolger einer Puzzlekonfigurationen können dann alleine aufgrund der alten Position des Leerraums, ohne Berücksichtigung der Nummern der ihn umgebenden Steine ermittelt werden. Dadurch werden insbesondere die Fälle vereinfacht, bei denen der Leerraum am Rand der Konfiguration liegt.

3.1.2 Generische Suche

Das generische Verfahren zum Lösen von Suchproblemen

Algorithmus 3.1 zeigt das Skelett eines Suchverfahrens, das wir im Folgenden auch als *generisches Suchverfahren* bezeichnen werden, weil es noch so allgemein gehalten ist, dass beliebige Suchverfahren durch Spezialisierung und Konkretisierung aus ihm gewonnen werden können.

1. Sei L die Liste der Startknoten für das Problem (ab jetzt wird L immer die Liste der noch nicht überprüften Knoten darstellen).
2. Ist L leer, so melde einen Fehlschlag. Andernfalls wähle einen Knoten n aus L .
3. Stellt n einen Zielknoten dar, so melde Erfolg und liefere den Pfad vom Startknoten zu n .
4. Andernfalls ersetze in L den Knoten n durch seine Nachfolgeknoten. Markiere dabei die neuen Knoten mit dem jeweils zugehörigen Pfad vom Startknoten.
5. Weiter mit Schritt 2!

Algorithmus 3.1: Generisches Verfahren zum Lösen von Suchproblemen.

Dieses Verfahren verwaltet in der Liste L die *Knoten*, d.h. Repräsentanten von Suchraumzuständen. Bei diesen Knoten werden auch jeweils die Pfade vermerkt, auf denen sie ausgehend vom Startknoten von dem Suchverfahren erreicht wurden. Falls es nicht vorher bereits einen anderen Zielknoten findet, wird das Suchverfahren jeden Knoten der Liste L daraufhin überprüfen, ob er einen Zielknoten darstellt (Schritt 3). Die Liste L heißt daher auch *open list* oder *Liste offener Knoten*. Wir wollen sie im Folgenden als *Agenda* bezeichnen. Nachdem sie die Blätter des von dem Suchverfahren durch die Pfade in L implizit definierten Baumes (bzw. Waldes im Falle multipler Startknoten) repräsentiert, nennen wir sie auch manchmal – in Analogie zum Saum eines Kleides – den (momentanen) *Saum* der Suche.

Solange die Agenda L noch mindestens einen Knoten enthält, besteht noch eine Chance, dass ein Zielknoten gefunden wird. Wird sie jedoch leer, so bedeutet dies, dass der von dem Verfahren prinzipiell explorierbare Teil des Suchraums vergeblich auf einen Zielknoten hin durchforstet wurde (jeder offene Knoten wurde bereits getestet und es gibt keine Nachfolger mehr). Das Verfahren signalisiert dann einen Fehlschlag.

Knoten gelangen in die Liste L , wenn sie Nachfolger eines Knotens n sind, von dem sich das Suchverfahren bereits überzeugt hat, dass er kein Zielknoten ist. Sie ersetzen dann in L den Knoten n (Schritt 4). Dieser Vorgang wird als *Expansion* des Knotens n bezeichnet. Knoten werden von dem skizzierten einfachen Suchverfahren erst beim Expandieren und nicht schon zum Zeitpunkt ihres Eintragens in die Agenda auf Zieleigenschaft geprüft. Dies vereinfacht die

nachfolgenden Überlegungen zur Komplexität von Konkretisierungen des generischen Verfahrens.

Die Art und Weise, wie die Wahl eines Knotens aus L in Schritt 2 des Suchverfahrens 3.1 vorgenommen wird, bestimmt die sog. *Suchstrategie*, die die wesentlichen Eigenschaften dieser Konkretisierungen festlegt (mehr dazu im Abschnitt 3.1.3). Wird L als eine geordnete lineare Datenstruktur (eine Liste im programmiersprachlichen Sinne) repräsentiert, so kann man eine beliebige Auswahlstrategie dadurch erreichen, dass man Elemente grundsätzlich am vorderen Ende der Liste entnimmt und zusätzlich eine Funktion festlegt, die bestimmt, an welcher Stelle neue Elemente in L eingeordnet werden.

Erfolgt die Auswahl eines Knotens aus der Agenda L *alleine aufgrund seiner Position im soweit generierten Baum* zu L , also insbesondere unabhängig von seinem Inhalt (der bei ihm gespeicherten Zustandsbeschreibung), so spricht man auch von *uninformierter* oder *blinder* Suche. Spielt dagegen bei der Auswahl auch der *Inhalt* des Knotens eine Rolle, so spricht man von *informierter* oder *heuristischer* Suche. Beiden Arten von Suche ist im Folgenden ein eigener Abschnitt gewidmet.

Vorwärts- vs. Rückwärtssuche

Häufig sind Zielzustände nur implizit gegeben (beispielsweise beim Lösen eines Kreuzworträtsels), weshalb das geschilderte generische Suchverfahren den Suchraum auch *vorwärts*, beginnend bei den typischerweise explizit vorgegebenen Startknoten exploriert, bis es einen Knoten findet, der die Zieleigenschaft aufweist.

Bei manchen Suchproblemen sind jedoch neben den Start- auch die Zielzustände explizit vorgegeben oder mit einem – im Vergleich zur Lösung des Suchproblems – geringen Aufwand durch ein Programm aufzählbar. Wenn dann auch noch die den Suchraum definierenden Operatoren *einfach invertierbar* sind (was ganz und gar nicht selbstverständlich ist, wenn diese Operatoren nur prozedural gegeben sind – vgl. unsere Anmerkungen zum Schiebepuzzle), so kann man eine sog. *zielorientierte Suche* durchführen. Dazu exploriert man den Suchraum *rückwärts*, beginnend bei einem der Zielknoten, indem man zu einem vorgegebenen Zustand des Suchraums iteriert jeweils alle möglichen direkten *Vorgänger* generiert, bis man einen Startknoten antrifft. Man macht bei zielorientierter Suche also im Endeffekt eine reguläre (Vorwärts-) Suche in demjenigen Suchraum, der aus dem ursprünglichen Suchraum entsteht, indem man die Operatorkanten umdreht, die Rolle von Start- und Zielknoten miteinander vertauscht und Lösungspfade dieses neuen Suchproblems in umgekehrter Kantenreihenfolge als Lösungspfade des ursprünglichen Suchproblems ausgibt.

Ein einfaches Suchproblem, bei dem zielorientierte Suche leicht möglich ist, ist das Schiebepuzzle-Problem aus Beispiel 3.1.1: Es gibt genau einen, explizit bekannten Zielzustand (der, bei dem die Steine, links oben, mit 1 beginnend, gemäß ihrer Numerierung im Uhrzeigersinn angeordnet sind) und zu jeder Schiebeoperation existiert eine eindeutige, sie kompensierende, inverse Schiebeoperation.

Haben die Knoten eines Unterraums des Suchraums typischerweise einen kleineren Eingangs- als Ausgangsgrad, so verhält sich für diesen Unterraum Rückwärtssuche besser als Vorwärts-suche. Deshalb bieten sich auch Varianten der generischen Vorgehensweise an, bei denen nicht der ganze Suchraum ausschließlich vorwärts bzw. rückwärts durchsucht wird, sondern – je nach seiner Gestalt – der eine Unterraum vorwärts und der andere rückwärts exploriert wird. So wird

z.B. bei *inselgesteuerter Suche* zunächst eine „Insel“ (ein Knoten auf dem halben Wege vom Start- zum Zielknoten) bestimmt und dann nach einem Lösungspfad gesucht, der über diese Insel läuft (dazu muß insbesondere der Wurzelpfad zur Insel bestimmt werden). Findet man keinen solchen Pfad, so löst man das ursprüngliche Problem unter Ignorieren der Insel.

Suche in Graphen

Eventuell repräsentieren unterschiedliche Knoten des Baums, der von der Suchprozedur definiert wird, ein und denselben Zustand im Suchraum, der nur auf *verschiedenen Wegen erreichbar* ist. So läßt sich im vorausgehenden Beispiel 3.1.1 jede Teillösung des Problems durch entsprechende inverse Schiebeoperationen in die Ausgangssituation rücküberführen, d.h. jeder Punkt des Suchraums zu diesem Problem kann auf beliebig vielen und beliebig langen Wegen erreicht werden, die beim Startknoten beginnen.

Führen zu einem Knoten im Suchraum unterschiedliche Wege, so hat dies im Allgemeinen zur Folge, dass er vom Suchverfahren wiederholt auf die *Agenda L* gesetzt wird. Enthält der Suchraum Zyklen, so kann dies je nach verwendeter Suchstrategie sogar zur Nichtterminierung der Suche führen: Das Verfahrenwickelt dann einen Zyklus unendlich oft als unendlich langen Pfad im generierten Baum ab. Auf jeden Fall trägt jedoch ein Suchraum, der nicht Baumgestalt hat, auch bei Abwesenheit von Zyklen die Gefahr in sich, dass das Suchverfahren Teile von ihm, wenn auch nicht unendlich oft, so doch unnötig wiederholt exploriert.

Um dies zu verhindern, modifiziert man deshalb bei Suchräumen, die keine Bäume darstellen, das generische Suchverfahren so, dass es einen Knoten nur dann zur Liste *L* der noch zu explorierenden Knoten hinzufügt, wenn er nicht schon in *L* oder der Liste *bereits erschlossener Knoten* ist. Die Liste *C* der bereits erschlossenen Knoten (engl. *closed list*) enthält all jene Knoten, die schon geprüft und aus *L* entfernt wurden³. Selbst wenn dann vom Suchverfahren Knoten mehrfach erzeugt werden, werden so die entsprechenden Teilaräume nicht erneut durchsucht.

Beispiel 3.1.2. Missionare und Kannibalen (siehe auch [15]):

Zwei Missionare und zwei Kannibalen müssen von einer Uferseite auf die andere gelangen. Dazu steht ihnen ein Ruderboot zur Verfügung, das maximal 2 Personen faßt. Wann immer aber die Kannibalen auf einer Uferseite in der Überzahl sind, müssen die Missionare um ihr Leben fürchten. Wie kann das Ruderboot so eingesetzt werden, dass alle sechs Personen lebend auf die andere Uferseite gelangen?

Abbildung 3.3 zeigt den zugehörigen Suchraum, der entsteht, wenn man bei seiner Festlegung folgende vereinfachende, den Suchraum klein haltende, Annahmen trifft: In den Knoten wird nur jeweils die Gemengelage auf einer der beiden Uferseiten (z.B. immer der linken) repräsentiert (da damit die jeweils andere Seite festliegt) und von den konkreten Missionaren bzw. Kannibalen abstrahiert (nur deren jeweilige Zahl ist für das Problem interessant). Außerdem wird ein Nachfolgezustand nur dann in die Agenda aufgenommen, wenn er nicht schon einmal vorher erzeugt wurde (andernfalls wäre der Suchraum zyklisch und der zugehörige Baum unendlich groß) und auch dann nur, wenn er zulässig ist, also für die Missionare keine Bedrohung darstellt. Ein mit (xM, yC, b) markierter Knoten repräsentiert dabei den Zustand, in dem x Missionare und y Kannibalen auf der repräsentierten Uferseite versammelt sind und das Boot auf ihrer Seite

³ In der Fabel von Ariadne und Minotaurus spielt der Faden, mit dem sich Ariadne im Labyrinth zurechtfindet, die Rolle der Liste der bereits erschlossenen Knoten.

haben, falls b nicht leer ist und eine Kanten zwischen zwei Knoten repräsentiert die Überfahrt, die aus dem Zustand des Vaterknotens den des Tochterknotens macht.

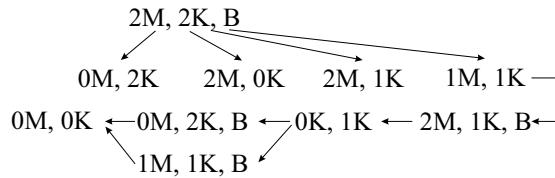


Abbildung 3.3: Missionare und Kannibalen.

Wie man sieht, enthält der Suchraum mit den genannten Vereinfachungen nur einige wenige Knoten und weist auch nur einen kleinen Verzweigungsgrad auf. Außerdem sind für dieses Suchproblem der Vorwärts- und Rückwärtssuchraum identisch, da es sich um ein symmetrisches Problem handelt: Zu jedem Bootstransfer gibt es einen inversen Bootstransfer und es macht keinen Unterschied, ob man die gesamte Gesellschaft sicher von einem Ufer zum anderen oder umgekehrt bringen will.

Natürlich darf nicht übersehen werden, dass das Verwalten der Liste C und die Redundanztests mit den Listen der noch offenen (L) bzw. bereits erschlossenen Knoten (C) ebenfalls nicht zu unterschätzende Kosten verursachen: Das Suchverfahren muß dazu ständig den gesamten soweit explorierten Teil des Suchraums im Speicher halten und bei jedem Erzeugen eines Knotens jeweils sowohl die Liste L als auch die immer länger werdende Liste C der bereits erschlossenen Knoten daraufhin überprüfen, ob der fragliche Knoten bereits in ihr enthalten ist.

3.1.3 Suchstrategien und deren Bewertung

Die Suchstrategie, d.h. die Auswahl in Schritt 2 des generischen Suchverfahrens 3.1 bestimmt die wesentlichen Eigenschaften von Konkretisierungen des Verfahrens. Sie bestimmt, welcher Teil des Suchraums in welcher Reihenfolge exploriert wird.

Von der Suchstrategie hängt z.B. ab, ob das Suchverfahren *vollständig* ist, d.h. schließlich jeden Knoten des Suchraums expandiert, sofern nicht vorher ein Zielknoten gefunden wird. In diesem Fall spricht man manchmal auch von *erschöpfender* Suche. Ein vollständiges Suchverfahren findet garantiert eine Lösung, wenn es nur mindestens eine Lösung gibt – sogar dann, wenn der Suchraum Zyklen enthält, die zu unendlichen Wurzelpfaden in dem Baum führen könnten, der von dem Verfahren in L aufgebaut wird. Im Gegensatz dazu kann es bei einer *unfairen* Suchstrategie passieren, dass das Verfahren eifrig einen unendlich tiefen Ast des generierten Baums immer weiter weiterfolgt und Zielknoten zwar auf die Agenda plazierte, aber nie expandiert, weil sie nicht auf dem vorrangig verfolgten unendlichen Pfad liegen. Enthält der Suchraum mehr als eine Lösung, so ist es natürlich wünschenswert, eine möglichst gute Lösung zu finden. Dabei definiert das Verfahren, was als Güte einer Lösung anzusehen ist. Üblicherweise wird hier als Güte einer Lösung ein Maß für die Kosten genommen, die durch das Herstellen der Zielzustände aus dem jeweiligen Startzustand verursacht werden. Im einfachsten Fall – wenn man unterstellt, dass jedes Anwenden eines Operators die Einheitskosten 1 verursacht – ist dieses Maß durch die Länge des kürzesten Pfads gegeben, der vom Start- zum Zielknoten führt. Ein Suchverfahren heißt *optimal*, wenn es immer eine bezüglich des jeweiligen Gütemaßes optimale Lösung findet, sofern es überhaupt eine Lösung gibt.

Neben dem Aufwand für die Herstellung einer gefundenen Lösung in der realen Problemlösewelt ist natürlich auch der Aufwand für das eigentliche Finden der Lösung (Zielknoten und zugehöriger Wurzelpfad zu seiner Herstellung aus dem Startknoten) ein wichtiges Gütekriterium für ein Suchverfahren. Unterschiedliche Konkretisierungen haben im Allgemeinen auch eine unterschiedliche algorithmische Komplexität, gehen also unterschiedlich mit den Ressourcen Zeit und Speicher um.

Bei der nachfolgenden Diskussion konkreter Suchverfahren legen wir bei der Untersuchung ihrer algorithmischen Komplexität – sofern jeweils nicht ausdrücklich etwas anderes erwähnt wird – die folgenden einfachen Komplexitätsmaße zugrunde:

- Unter dem *Speicherbedarf* $Space(X)$, den das Suchverfahren X für ein konkretes Suchproblem hat, wird die maximale Anzahl von während der Suche gleichzeitig in der Agenda zu speichernden Knoten verstanden.
- Der *Zeitbedarf* $Time(X)$, den das Suchverfahren X für ein konkretes Suchproblem hat, gibt an, wie viele Knoten bei der Suche vom Suchverfahren auf ihre Zieleigenschaft hin untersucht werden.

Wenn nicht anders angegeben, nehmen wir bei den entsprechenden Komplexitätsbetrachtungen auch jeweils (stark) vereinfachend an, dass der Suchraum *uniform* ist, d.h. einen konstanten Verzweigungsgrad b und eine einheitliche Tiefe d hat, wobei der Wurzelknoten die Tiefe $d = 0$ haben soll. Die l Kinder eines Knotens n_i bezeichnen wir dann als $n_{i,1}, \dots, n_{i,l}$ (damit lässt sich am Multiindex eines Knotens direkt dessen Wurzelpfad im Suchbaum ablesen).

Außerdem unterstellen wir zunächst, dass es im Suchraum nur eine Lösung g gibt, die mit gleicher Wahrscheinlichkeit an jeder Stelle in der Maximaltiefe d liegen kann und geben dann für die Suchverfahren, abhängig von der Lage der Lösung, jeweils den günstigsten (engl. best case), mittleren (engl. average case) und ungünstigsten (engl. worst case) Speicher- bzw. Zeitbedarf an.

3.2 Uninformierte Suchverfahren

3.2.1 Breitensuche

Das erste uninformed Suchverfahren, dem wir uns zuwenden wollen, entsteht aus dem generischen Suchverfahren 3.1 durch jene Konkretisierung des Auswahlschritts 2, die in Algorithmus 3.2 gezeigt wird. Die Agenda wird hier als eine linear geordnete Datenstruktur verwaltet, in die neue Knoten am hinteren Ende der Agenda eingefügt und in der Agenda vorhandene Knoten am vorderen Ende der Agenda entnommen werden. Sie verhält sich damit wie eine Warteschlange.

Verwaltet man die Agenda auf die geschilderte Art und Weise, so verfährt das Suchverfahren nach folgender Auswahlmaxime:

Ein Knoten auf der Suchbaumtiefe k darf erst dann expandiert werden, wenn *alle* Knoten der Tiefe $k - 1$ bereits expandiert sind.

1. Sei L die Liste der Startknoten für das Problem.
2. Ist L leer, so melde einen Fehlschlag.
Andernfalls wähle den *ersten* Knoten n aus L .
3. Stellt n einen Zielknoten dar, so melde Erfolg und liefere den Pfad vom Startknoten zu n .
4. Andernfalls entferne n aus L und füge seine Nachfolgeknoten *am Ende* von L an.
Markiere dabei wieder die neuen Knoten mit dem jeweils zugehörigen Pfad vom Startknoten.
5. Weiter mit Schritt 2!

Algorithmus 3.2: *Breitensuche*.

Das Suchverfahren durchforstet damit den Baum Schicht für Schicht von oben nach unten. Es wird deshalb als *Breitensuche* (engl. breadth first, BF) bezeichnet.

Die genannte Auswahlmaxime legt die Suchstrategie noch nicht eindeutig fest, nachdem in Schritt 4 noch offengelassen ist, in welcher Reihenfolge die Nachfolger eines Knotens in die Agenda einzufügen sind, wenn ein Knoten mehr als einen Nachfolger besitzt. Die sich anschließenden Überlegungen haben aber unabhängig davon Gültigkeit.

Wir hatten vereinfachend angenommen, dass der Suchraum uniform ist und der Zielknoten auf der Tiefe d (dem Saum des vollständig generierten Suchbaums) liegt. Nachdem von der Breitensuche alle Knoten auf der Tiefe $k - 1$ expandiert sein müssen, bevor der erste Knoten auf der Tiefe k getestet werden darf, enthält die Agenda L vor der ersten Überprüfung (und ggf. Expansion) eines Knotens der Tiefe k *alle* Knoten der Tiefe k . Aufgrund der Uniformität des Suchraums und der Annahme, dass die Lösungen auf dem Saum des Suchraums, in Tiefe d , liegen, ist der *Speicherbedarf* für die Breitensuche – unabhängig von der Lage der Lösung im Saum des Suchraums – immer

$$\text{Space(BF)} = b^d,$$

also exponentiell in der Tiefe des Suchbaums!

Ein Knoten wird von der Breitensuche erst untersucht, wenn er expandiert, nicht schon, wenn er in L gespeichert wird. Um auf die Tiefe d des Zielknotens zu gelangen, müssen also vorher alle Knoten auf den höheren Ebenen $0, \dots, d - 1$ untersucht werden. Von diesen gibt es (wieder aufgrund der Uniformitätsannahme) $\sum_{k=0}^{d-1} b^k = \frac{b^d - 1}{b - 1}$. Bis der Zielknoten angetroffen wird, sind dann vom Saum des Suchbaums bestenfalls einer (nämlich der Lösungsknoten) und schlimmstenfalls alle b^d Knoten der Tiefe d auf ihre Zieleigenschaft hin zu untersuchen (wenn der Lösungsknoten der letztuntersuchte Knoten der Tiefe d ist). Der günstigste Zeitbedarf ist demnach $\frac{b^d + b - 2}{b - 1} = O(b^{d-1})$ und der ungünstigste $\frac{b^{d+1} - 1}{b - 1} = O(b^d)$. Daraus resultiert ein *mittlerer Zeitbedarf*

$$\text{Time(BF)} = \frac{b^{d+1} + b^d + b - 3}{2(b - 1)} = O(b^d)$$

für die Breitensuche, der – wie schon der Speicherbedarf der Breitensuche – exponentiell in der Tiefe des Suchbaums ist.

Die oben genannte Auswahlmaxime stellt sicher, dass Breitensuche vollständig ist, also auf jeden Fall eine Lösung findet, wenn es überhaupt eine Lösung gibt. Im schlimmsten Fall muß dazu der ganze Suchraum durchsucht werden. Nachdem der Suchraum aufgrund der Uniformitätsannahme von der Größe b^d ist, entschuldigt dies in gewisser Weise das geschilderte kostspielige Zeitverhalten der Breitensuche als im Prinzip unvermeidlich.

Ärgerlich ist dagegen, dass die Breitensuche im schlimmsten Fall Speicher in der Größe des gesamten Suchraums abzüglich des Saums benötigt, um eine Lösung zu finden. Dafür garantiert die Auswahlmaxime dieses Verfahrens, dass von ggf. mehreren vorhandenen eine Lösung zuerst gefunden wird, die an höchster Stelle im Suchbaum liegt, also mit einer minimalen Anzahl von Operatoranwendungen aus dem Startzustand hergestellt werden kann.

3.2.2 Gleiche-Kosten-Suche

Bei dem folgenden Suchverfahren kann man sich streiten, ob es sich nach unserer Definition um ein uninformiertes oder ein informiertes Suchverfahren handelt. Bei diesem Verfahren werden zur Steuerung der Suche zwar keine Informationen über die Zustände verwendet, die von den Knoten repräsentiert werden. Jedoch müssen für jeden Operator die Kosten $c(n \rightarrow n')$ bekannt sein, mit denen er den vom Knoten n repräsentierten Zustand in den durch n' repräsentierten Folgezustand transformiert.

Im Folgenden sei davon ausgegangen, dass Operatoren nur positive Kosten verursachen, es also keine Operatoren gibt, deren Anwendung umsonst ist oder gar Gewinne (negative Kosten) bringt. Außerdem nehmen wir im Folgenden an, dass es eine (positive) untere Schranke ϵ für die Kosten der Operatoren gibt, d.h. für alle Knotenpaare n, n' die Beziehung

$$c(n \rightarrow n') \geq \epsilon > 0$$

gilt.⁴ Die zugehörige Kostenfunktion auf Operatoren (Kanten im Suchraum) läßt sich dann in Suchräumen, die Bäume darstellen, auf naheliegende Weise zu einer (strikt) *monotonen* Kostenfunktion $g(n_k)$ auf Pfaden $\langle n_0, \dots, n_k \rangle$ vom Startknoten n_0 zum Knoten n_k erweitern, indem man die Kosten der beteiligten Operatoren einfach aufaddiert:

$$g(n_k) := \sum_{i=0}^{k-1} c(n_i \rightarrow n_{i+1}).$$

Gleiche-Kosten-Suche (engl. uniform cost search, branch & bound) ist dann ein Suchverfahren, das jene Knoten der Agenda L vorrangig expandiert, deren Wurzelpfade die geringsten Kosten besitzen.

Ein Spezialfall von Gleiche-Kosten-Suche ist Breitensuche, da die Breitensuche jene Zielknoten zuerst expandiert, die weit oben im Suchbaum liegen, also die wenigstens Operatoranwendungen zu ihrer Herstellung aus dem Startknoten erfordern. Ordnet man nämlich allen Operatoren Einheitskosten der Größe 1 zu, so stimmt die Länge des Pfades zu einer Lösung mit deren Kosten überein, d.h. Breitensuche findet zuerst solche Lösungen, die im genannten Sinne kostenoptimal sind.

⁴ Eine solche untere Schranke gibt es natürlich immer schon dann, wenn die Zahl der zur Problemlösung zur Verfügung stehenden Operatoren – wie vorne unterstellt – endlich ist.

Nachdem die Kostenfunktion auf den Wurzelpfaden strikt monoton ist, lässt sich die Argumentation für die Vollständigkeit der Breitensuche direkt auf die Gleiche-Kosten-Suche übertragen. Gleiche-Kosten-Suche ist also nicht nur optimal sondern auch vollständig.

Im Fall von Suchräumen, die keine Bäume sind, also Zusammenführungen von Pfaden oder gar Zyklen enthalten (ein Knoten ist dann nicht mehr auf eindeutigem Wege vom Startknoten aus zu erreichen), ordnet man einem Knoten natürlich jeweils die Kosten des Pfades zu, auf dem der Knoten mit minimalen Kosten erreichbar ist und geht ansonsten genauso vor wie für Suchbäume.

Technisch lässt sich dies elegant dadurch erreichen, dass man vor dem Einfügen eines Knotens in die Agenda prüft, ob der dort bereits mit einem Pfad enthalten ist, der größere Kosten verursacht und dann ggf. diesen Pfad destruktiv durch den neu gefundenen, kostengünstigeren Pfad ersetzt und den Knoten nicht neu auf die Agenda plaziert. Die Agenda enthält damit zu jedem Zeitpunkt für jeden auf ihr gespeicherten Knoten n einen Pfad, der relative zum bereits explorierten Teil des Suchraums kostenoptimal ist und die soweit ermittelten Kosten $g(n)$ stellen damit eine obere Schranke für die *tatsächlichen* Kosten $g^*(n)$ des Knotens dar

$$g^*(n) \leq g(n).$$

Das auf allgemeine Suchräume angepaßte Suchverfahren wird manchmal auch *verbesserte Gleiche-Kosten-Suche* genannt.

Gleiche-Kosten-Suche ist übrigens ein Spezialfall des noch vorzustellenden informierten Suchverfahrens, das durch den sog. A*-Algorithmus realisiert wird und teilt deshalb auch wesentliche Eigenschaften von A*-Suche (mehr dazu siehe Abschnitt 3.3.3).

3.2.3 Tiefensuche

Im Gegensatz zur Breitensuche wird die Agenda bei dem in Algorithmus 3.3 gezeigten Verfahren der sog. *Tiefensuche* (engl. depth-first, DF) wie ein Keller verwaltet: Neue Knoten werden am gleichen (vorderen) Ende der Agenda eingeordnet, an dem der jeweils als nächstes zu überprüfende Knoten vom Suchverfahren entnommen wird. Die Auswahlmaxime zur Tiefensuche lautet also:

Bevor im Suchbaum Geschwister eines Knotens expandiert werden dürfen, müssen erst alle Kinder und Kindeskinder dieses Knotens expandiert worden sein.

Das Suchverfahren durchforstet damit immer erst den kompletten Suchbaum unter dem zuletzt expandierten Knoten der Agenda, bevor es im Suchbaum wieder aufsteigt – daher auch der Name Tiefensuche.

Auch die Maxime zur Tiefensuche legt die Suchstrategie noch nicht vollständig fest, da – ähnlich wie bei der Breitensuche – noch zusätzlich spezifiziert werden muß, in welcher Reihenfolge jeweils die Kinder eines Knotens an der alten Stelle ihres Elternknotens in die Agenda einzufügen sind.

Repräsentiert n den momentanen Suchzustand, ist also n der erste Knoten auf der Agenda, dann müssen – der Auswahlmaxime für Tiefensuche zur Folge – alle *unexpandierten Geschwister*

1. Sei L die Liste der Startknoten für das Problem.
2. Ist L leer, so melde einen Fehlschlag. Andernfalls wähle den *ersten* Knoten n aus L .
3. Stellt n einen Zielknoten dar, so melde Erfolg und liefere den Pfad vom Startknoten zu n .
4. Andernfalls entferne n aus L und füge seine Nachfolgeknoten *am Anfang* von L an. Markiere dabei wieder die neuen Knoten mit dem jeweils zugehörigen Pfad vom Startknoten.
5. Weiter mit Schritt 2!

Algorithmus 3.3: *Tiefensuche.*

von Knoten auf dem *Wurzelpfad* vom Startknoten zum Knoten n in der Agenda gespeichert sein (sie werden ja erst beim Aufstieg expandiert). Nachdem wir vereinfachend angenommen hatten, dass der einzige Lösungsknoten im Saum des Suchbaums, d.h. auf der untersten Ebene d liegt, und dass der Suchbaum einen konstanten Verzweigungsgrad b hat, ist der *Speicherbedarf Space(DF)* für die Tiefensuche – unabhängig von der Lage der Lösung im Saum – linear in d :

$$\text{Space(DF)} = d(b - 1) + 1 = O(d)$$

(dieser Speicher wird erstmals bei Expansion des ersten Knotens auf Tiefe d benötigt).

Der mittlere Zeitbedarf für die Tiefensuche ist nicht ganz so leicht zu ermitteln – wir folgen hier der Analyse in [7]. Dazu legen wir uns zunächst auf eine Ordnung der Kinder eines jeden Knotens fest und stellen uns den Suchbaum so gezeichnet vor, dass Knoten, die in dieser Ordnung vor ihren Geschwistern liegen, links von ihren Geschwistern angeordnet werden. Den so skizzierten Baum traversiert dann das Tiefensuchverfahren von links unten nach rechts oben. Außerdem markieren wir jeden Knoten in diesem Baum mit dem Zeitpunkt, zu dem er expandiert wird, wenn vorher nicht ein Zielknoten angetroffen wird. Das Resultat ist für den Spezialfall $b = 3$ in Abbildung 3.4 (mit kleiner Änderung aus [7]) zu sehen.

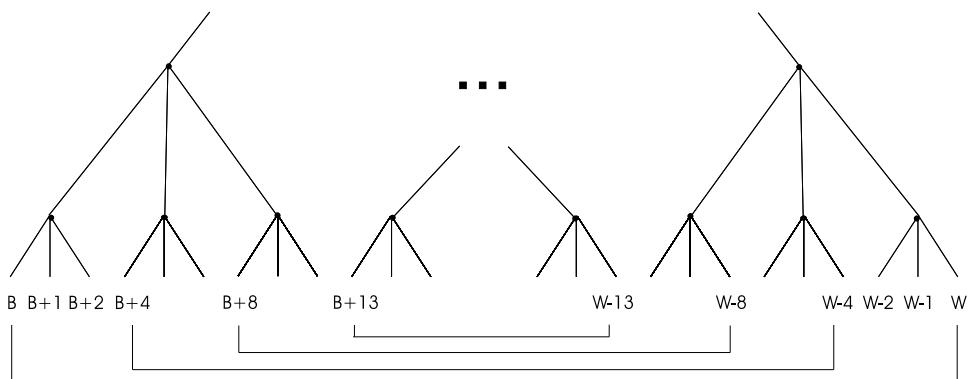


Abbildung 3.4: *Zeitbedarf Tiefensuche.*

Wenn das Verfahren nach B Zeiteinheiten zum ersten mal den Saum betritt und nach W Zeiteinheiten den Suchraum komplett durchsucht hat, so vergeht aufgrund unserer vereinfach-

chenden Annahmen mindestens die Zeit $B = d + 1 = O(d)$ und höchstens die Zeit $W = \sum_{k=0}^d b^k = \frac{b^{d+1}-1}{b-1} = O(b^d)$ bis der Zielknoten gefunden ist. Weil außerdem in der von uns gewählten Anordnung Saumknotenpaare mit gleichem Abstand zum linken bzw. rechten Saumrand jeweils die gleiche Summe $B + W$ ihrer Knotenmarkierungen aufweisen (in der Abbildung 3.4 sind solche Paare jeweils durch eine horizontale Klammer verbunden), hat die Tiefensuche einen mittleren *Zeitbedarf* von

$$\text{Time(DF)} = \frac{b^d/2 \times (B + W)}{b^d} = \frac{B + W}{2} = \frac{b^{d+1} + bd + b - d - 2}{2(b-1)} = O(b^d).$$

Wie bereits das Breitensuchverfahren weist also das Tiefensuchverfahren einen Zeitbedarf auf, der exponentiell in der Lösungstiefe des Suchraums wächst. In Bezug auf das Zeitverhalten kann also keinem der beiden Verfahren von vorneherein der Vorzug gegeben werden.

Anders liegen die Verhältnisse jedoch bei dem von uns ermittelten Speicherbedarf. Eine Suchstrategie, die *schnell zu Endzuständen* (Zuständen, die durch Knoten im Saum repräsentiert werden) vorstößt, ist *speicherökonomisch*, da allein die Endzustände für eine Verkleinerung der Agenda sorgen. Damit entscheidet also die Form des Suchraums, ob Breiten- oder Tiefensuche für ein vorgelegtes Suchproblem in Bezug auf den Umgang mit Speicher besser abschneidet: Solange der Suchbaum nicht im Vergleich zu seiner Breite sehr lange Wurzelpfade enthält und der Zielknoten weit oben und rechts von diesen langen Wurzelpfaden im Suchbaum liegt, wird die Tiefensuche der Breitensuche in Bezug auf das Speicherverhalten überlegen sein.

Fatal ist die Situation allerdings, wenn die erste Lösung rechts oberhalb eines Unterraums des Suchraums liegt, in dem ein unendlicher Wurzelpfad liegt. In diesem Fall wird die Tiefensuche diesem unendlichen Pfad in die Tiefe folgen und nicht terminieren. Tiefensuche ist also kein faires Suchverfahren. Sie ist i. allg. auch nicht optimal, da sie von zwei Lösungen immer diejenige zuerst finden wird, die links unterhalb der anderen liegt.

Häufig unangemessen ist auch das uninformierte Verhalten von Tiefensuche beim Antreffen von *Sackgassen*, d.h. von Knoten des Suchraums, die nicht mehr expandierbar und auch kein Ziel sind. Blinde Tiefensuche steigt in so einem Fall grundsätzlich zum Vaterknoten der Sackgasse auf, nimmt also die zeitlich zuletzt getroffene Entscheidung zurück und versucht dann über dessen nächstes Geschwister, d.h. mit einer Alternative für die letzte Entscheidung, wieder abzusteigen. Dieses sog. *chronologische Rücksetzen* ist aber z.B. beim Lösen eines Kreuzworträtsels fast immer kontraproduktiv: Meist ist nämlich nicht der *letzte* Versuch, ein Wort zu vervollständigen, sondern eine schon viel frühere Festlegung beim Ausfüllen des Rätsels für das spätere Scheitern des Lösungsversuchs verantwortlich und es wäre viel klüger, gleich *alle* Entscheidungen von der eigentlich verantwortlichen bis zur letzten auf einmal zurückzunehmen und die Suche mit einer Alternative für die verantwortliche Entscheidung fortzusetzen. Dafür wird aber i. allg. Information benötigt, über die ein blindes Suchverfahren nicht verfügt. In der Literatur werden zur Lösung des Sackgassenproblems im Wesentlichen zwei Lösungsvorschläge gemacht, die wir hier nicht weiter ausführen können und für die wir den Leser deshalb lediglich auf die Originalquellen verweisen: *schrittweise verbreitende Suche* (engl. iterative broadening, [8]) und *Suche mit abhängigkeiten-gesteuertem Rücksetzen* (engl. dependency-directed backtracking, [5, 23] und [3]).

3.2.4 Schrittweise vertiefende Suche

Wir hatten in den vorausgegangenen Abschnitten gesehen, dass das Tiefensuchverfahren meist ein etwas besseres Zeit- und ein deutlich besseres Speicherverhalten als das Breitensuchverfahren aufweist, dafür aber gelegentlich zum Nichtterminieren neigt und nicht optimal ist. Es wäre daher schön, wenn man nach der Aschenputtelmethode die guten Eigenschaften der beiden Verfahren kombinieren könnte ohne dabei gleichzeitig die schlechten Eigenschaften in Kauf nehmen zu müssen. Dieses Verfahren müsste also im Wesentlichen den Raumbedarf von Tiefensuche und den Zeitbedarf sowie die Robustheit und Optimalität von Breitensuche aufweisen.

Ein solches Suchverfahren gibt es tatsächlich. Die Schlüsselidee ist dabei, im Prinzip Breitensuche durchzuführen, dabei jedoch nicht jeweils alle Knoten einer Zwischenebene zwischenzuspeichern, sondern diese jeweils zum Expansionszeitpunkt ggf. erneut zu erzeugen. Wie wir gleich sehen werden, rechnet sich diese wiederholte Erzeugen, solange die Zeit für das erneute Erzeugen nur kleiner ist, als die für das Überprüfen des Saums. Nachdem sich Tiefensuche außerdem vor allem dann schlecht verhält, wenn der Zielknoten höher liegt als der Suchraum tief ist, ist es gut, wenn der Suchraum nicht tiefer durchsucht würde, als das oberste Ziel liegt. Dem wird in Algorithmus 3.4 Rechnung getragen.

1. Sei $c = 1$ (c steht für die maximale Suchtiefe).
2. Sei L die Liste der Startknoten für das Problem.
3. Ist L leer, so erhöhe c um 1 und mache weiter mit Schritt 2! Andernfalls sei n der erste Knoten in L .
4. Stellt n einen Zielknoten dar, so melde Erfolg und liefere den Pfad vom Startknoten zu n .
5. Andernfalls entferne n aus L . Befand sich n auf einer Tiefe kleiner als c , so füge an den Anfang von L die Nachkommen von n an. Markiere dabei die neuen Knoten jeweils mit dem zugehörigen Wurzelpfad, der beim Startknoten beginnt.
6. Weiter mit Schritt 3!

Algorithmus 3.4: *Schrittweise vertiefende Suche.*

Wie man sieht, führt der Algorithmus im Wesentlichen eine *iterierte Tiefensuche* durch, bei der nur bis zu einer bestimmten *maximalen Suchtiefe* c im Suchraum vorgestoßen wird, die bei jedem neuen Suchvorgang um 1 erhöht wird. Daher wird dieses Suchverfahren auch als Suchverfahren der *schrittweisen Vertiefung* (engl. iterative deepening, ID) bezeichnet.

Offensichtlich findet das Verfahren ID wie die Breitensuche immer zuerst die *höchstgelegene Lösung*, ist also optimal und vollständig und kann sich nicht wie die Tiefensuche in unendlichen Wurzelpfaden verirren. Aufgrund seiner Realisierung hat es jedoch – unabhängig von der Lage der Lösung im Saum – den gleichen Raumbedarf, den auch die Tiefensuche aufweist:

$$\text{Space(ID)} = \text{Space(DF)}.$$

Der Speicherbedarf von ID stimmt also mit dem Speicherbedarf für die letzte Iteration überein.

Der Zeitbedarf für die letzte Iteration der schrittweisen vertiefenden Suche ist der, den eine Tiefensuche mit einer Tiefenbegrenzung von d aufweisen würde. Vorher werden aber noch

alle Teilbäume zu den Tiefen $1, \dots, d - 1$ jeweils komplett aber vergeblich durchsucht, wobei die Zeit $T = \sum_{j=0}^{d-1} \frac{b^{j+1}-1}{b-1} = \frac{b^{d+1}-bd-b+d}{(b-1)^2}$ verstreicht. Zusammen ergibt das im günstigsten Fall einen Zeitbedarf von $T + (d + 1) = O(b^{d-1})$, im ungünstigsten Fall von $T + \frac{b^{d+1}-1}{b-1} = O(b^d)$ und im Mittel von $\frac{b^{d+2}+b^{d+1}+b^2d+b^2-4bd-5b+3d+2}{2(b-1)^2}$, was für große d von $\frac{(b+1)b^{d+1}}{2(b-1)^2}$ dominiert wird, also in etwa mit dem Zeitbedarf $Time(DF)$ der Tiefensuche übereinstimmt. Insgesamt erhalten wir damit für das Verhältnis des mittleren Zeitbedarfs für die schrittweise vertiefende und die reguläre Tiefensuche:

$$1 \leq \frac{Time(ID)}{Time(DF)} = \frac{b+1}{b-1} \leq 3.$$

Das *Iterieren* stellt also (für große d) – entgegen dem ersten Augenschein – *nicht wirklich ein Problem* dar.

3.3 Heuristische Suche

3.3.1 Heuristische Schätzfunktionen

Nicht nur für die von uns erörterten blinden Suchverfahren X gilt, dass sie ein Zeitverhalten von

$$Time(X) = O(b^d)$$

aufweisen. In [10] wird gezeigt, dass dies für jedes blinde Suchverfahren X gilt, das Anspruch auf Vollständigkeit erhebt. Damit sind aber solche blinden Verfahren *praktisch nicht einsetzbar*: Bereits bei niedrigen und schlanken Suchbäumen wächst der Zeitbedarf für die Suche in's Unermeßliche.

Abhilfe schaffen hier erst Suchverfahren, die Wissen über das konkrete Suchproblem, d.h. die zu den Knoten gespeicherten, zustandscharakterisierenden Daten auswerten, um den als nächstes zu expandierenden Knoten zu bestimmen. Um eine möglichst gute Auswahl sicherzustellen, wählen daher sog. *heuristische Suchverfahren* (der Klassiker zu diesem Thema ist [19]) bevorzugt solche Knoten, die vermutlich *nahe am Ziel* liegen. Die Grundidee ist dabei dieselbe wie bei dem Kinderspiel „Blinde Kuh“, bei dem ein Mitspieler mit verbundenen Augen, geleitet von unterschiedlich intensiven „heiß“- und „kalt“-Hinweisen der Mitspieler, einen bestimmten Gegenstand oder eine bestimmte Person finden soll: Eine höhere Temperatur signalisiert hier größere Nähe zum Ziel.⁵

Was genau bei heuristischer Suche unter „Nähe“ zu verstehen ist, hängt natürlich vom konkreten Problem ab. Selbst dann, wenn das Ziel gut bekannt ist, fällt es häufig schwer, ein gutes Maß für die Nähe zum Ziel festzulegen. So ist z.B. beim „Blinde Kuh“-Spiel das Repertoire an Verbalisierungsmöglichkeiten für unterschiedliche „heiß“- und „kalt“-Grade selbst bei recht phantasievollen Mitspielern ziemlich eingeschränkt.

⁵ Das Spiel müßte demzufolge besser „Heuristisch suchende Kuh“ heißen.

Natürlich würde man zur Steuerung der Suche gerne über eine Funktion h^* verfügen, die für beliebige Knoten n des Suchraums, auf der Grundlage der dort gespeicherten Informationen über den von n repräsentierten Zustand, die tatsächlichen Kosten $h^*(n)$ eines kürzesten Pfads (im einfachsten Fall also die kürzeste Distanz) von n zum Ziel angibt, falls das Ziel von n aus erreichbar ist. Diese Funktion h^* im Vorhinein zu bestimmen, ist aber im Allgemeinen genauso kostspielig wie die Bestimmung der eigentlichen Lösung des Suchproblems. Deshalb setzt man üblicherweise Daumenregeln zur *Abschätzung* h des tatsächlichen Abstands h^* zum nächstgelegenen Ziel ein und versucht dann mit Hilfe dieser Abschätzung einen möglichst guten, als nächstes zu expandierenden Knoten zu bestimmen.

Von dieser *heuristischen (Schätz-)Funktion h*, die wir im Folgenden manchmal auch salopp als *Heuristik* bezeichnen werden, erwartet man nun einerseits, dass sie den Suchprozeß nicht an der Nase herumführt, also bei tatsächlicher Annäherung an ein Ziel auch größere Nähe signalisiert und andererseits, dass sie für eine gegebene Situation mit möglichst geringem Aufwand eine Schätzung dieses Abstands ermöglicht – zwei Ziele, die offensichtlich nur schwer gleichzeitig zu erfüllen sind. Formal verlangt man von einer heuristischen Funktion lediglich, dass sie immer nicht-negative Werte liefert und zumindest die Zielknoten mit der Schätzung 0 bewertet. Gilt für zwei Heuristiken h_1 und h_2 die Beziehung $\forall n : h_1(n) \leq h_2(n)$, so heißt h_2 *besser informiert* als h_1 . Die am schlechtesten informierte Heuristik h ist offensichtlich die mit der Eigenschaft $\forall n : h(n) = 0$, die sog. „Ich-bin-schon-da“-Heuristik. Diese Heuristik macht offensichtlich gar keinen Gebrauch von den bei den Knoten gespeicherten, zustandscharakterisierenden Informationen.

Eine besonders einfache Heuristik h_1 für das Schiebepuzzle-Problem in Beispiel 3.1.1 besteht z.B. darin, die Anzahl der Steine zu zählen, die in der momentanen Situation noch nicht am richtigen Platz sind – die Situation in Abbildung 3.2 wäre danach mit 3 zu bewerten. Offensichtlich ist eine Schiebepuzzle-Stellung umso weiter vom Ziel entfernt, je mehr derartige Fehlplazierungen in ihr bestehen. Eine bessere, aber auch etwas aufwendiger zu bestimmende, Heuristik h_2 für das Schiebepuzzle erhält man durch Berechnen der sog. *Manhattan-Distanz*. Dabei bestimmt man für jeden Stein den kürzesten Weg zu seiner Zielposition und summiert diese Werte zu einer Bewertung der Gesamtstellung auf – für die gezeigte Beispielsituation erhält man damit ebenfalls den Wert 3. Die fehlplazierten Steine einer Situation gehen in die Manhattan-Distanz also umso stärker ein, je weiter sie von ihrer jeweiligen Zielposition entfernt sind. Die Heuristik der Manhattan-Distanz ist besser informiert als die der Fehlplazierungsanzahl.

Heuristische Suche involviert also Aktivität auf zwei Ebenen:

1. zur Ermittlung dessen, *was zu tun ist* (Meta-Ebene) und
2. zu dessen *Ausführung* (Objekt-Ebene).

Dabei soll sich der Aufwand für die Ermittlung dessen, was zu tun ist, bei der Ausführung bezahlt machen. Bei blinder Suche finden im Prinzip nur Aktivitäten auf der Objekt-Ebene statt. Sich alleine auf die Meta-Ebene zu verlassen, kann aber ebenfalls unproduktiv sein, vor allem dann, wenn die Bestimmung des richtigen nächsten Schrittes schwierig ist – in unserem „Blinde Kuh“-Beispiel etwa, weil einige Mitspieler ungenaue oder gar falsche und damit irreführende Temperaturangaben machen.

3.3.2 Suche mit schrittweiser lokaler Verbesserung

Das Vorhandensein einer heuristischen Schätzfunktion versetzt uns in die Lage, die blinden Suchverfahren dadurch zu verbessern, dass wir von ihr gut bewertete Knoten weiter vorne in die Agenda einordnen. Dieser Einordnungsvorgang findet jeweils dann statt, wenn ein Knoten expandiert, d.h. durch seine Nachfolger ersetzt wird. Wird dabei lediglich eine (nicht notwendigerweise echte) Teilmenge der neuen Knoten gemäß der Schätzfunktion aufsteigend geordnet und als Block an einer bestimmten Stelle in die Agenda gesetzt, so spricht man von *Suche mit schrittweiser lokaler Verbesserung*.

Bergsteigen (mit Zurücksetzen)

Algorithmus 3.5 stellt das einfachste Suchverfahren mit lokaler Verbesserung dar, das sog. *Bergsteigerverfahren* (engl. hill climbing with backtracking, BTHC). Bei diesem Verfahren wird eine Tiefensuche durchgeführt, bei der alle Nachfolger eines Knotens bei seiner Expansion als bezüglich der Heuristik *geordneter* Block an den Anfang der Agenda gesetzt werden.

1. Sei L die Liste der Startknoten für das Problem, *sortiert* nach der jeweils geschätzten Distanz h zum Ziel.
2. Ist L leer, so melde einen Fehlschlag.
Andernfalls wähle den *ersten* Knoten n aus L .
3. Stellt n einen Zielknoten dar, so melde Erfolg und liefere den Pfad vom Startknoten zu n .
4. Andernfalls entferne n aus L und füge seine Nachfolgeknoten, sortiert nach der jeweils geschätzten Distanz h zum Ziel, *am Anfang* von L an. Markiere dabei die neuen Knoten mit dem jeweils zugehörigen Pfad vom Startknoten.
5. Weiter mit Schritt 2!

Algorithmus 3.5: Das Bergsteigerverfahren.

Für die Schiebepuzzle-Spielsituation, die in Abbildung 3.2 gezeigt ist, liefert z.B. die Fehlplatzierungsheuristik für die drei möglichen Nachfolgezustände, die durch ein Bewegen des Leerraums nach links (l), rechts (r) bzw. oben (o) gegeben sind, heuristische Schätzungen von 2, 4 bzw. 3 und die Zugfolge $l \rightsquigarrow o \rightsquigarrow r$, die mit dem bestbewerteten Zug beginnt, führt auch direkt zum Ziel.

Lässt man die heuristische Schätzfunktion jeden Knoten mit 0 bewerten und macht sie damit effektiv blind, so degeneriert die Bergsteigersuche zur reinen Tiefensuche. In diesem Fall hat das Bergsteigerverfahren also die gleiche algorithmische Komplexität wie die Tiefensuche. Aus dem gleichen Grund ist die Bergsteigersuche für unendliche Suchräume im Allgemeinen auch nicht vollständig (unendliche Wurzelpfade!).

Optimistisches Bergsteigen

Wenn man sich in einer unbekannten Gegend verlaufen hat, ist auch schon eine vereinfachte Form des Bergsteigerverfahrens nützlich. Einer Pfadfinderregel zufolge soll man dann permanent versuchen, in der Landschaft abzusteigen und keinesfalls bergauf gehen: So hat man eine gute Chance, früher oder später in einem Tal und an einem (natürlich abwärts fließenden) Gewässer anzugelangen, wo bevorzugt Siedlungen anzutreffen sind. Als Verbesserung der eigenen

Situation wird bei dieser Pfadfinderregel also die lokale Verminderung der Höhe über dem Meeresspiegel genommen.

Als Suchverfahren ließe sich dieses Verhalten dadurch realisieren, dass man statt aller Nachfolger eines Knotens nur den jeweils bestbewerteten Nachfolger an den Anfang der Agenda setzt und hofft, dass trotz dieser lokalen Festlegung auf einen Nachfolger noch ein Ziel gefunden wird – siehe Algorithmus 3.6.

1. Sei L die Liste der Startknoten für das Problem, *sortiert* nach der jeweils geschätzten Distanz h zum Ziel.
2. Ist L leer, so melde einen Fehlschlag. Andernfalls wähle den *ersten* Knoten n aus L .
3. Stellt n einen Zielknoten dar, so melde Erfolg und liefere den Pfad vom Startknoten zu n .
4. Andernfalls entferne n aus L und füge (nur) den besten Nachfolgeknoten von n (den mit minimaler geschätzter Distanz h zum Ziel) *am Anfang* von L an. Markiere dabei den neuen Knoten mit dem zugehörigen Pfad vom Startknoten.
5. Weiter mit Schritt 2!

Algorithmus 3.6: Optimistisches Bergsteigen.

Wir bezeichnen dieses Verfahren auch als *optimistisches Bergsteigerverfahren* (engl. strict hill climbing, SHC).⁶ Es führt im Wesentlichen eine Tiefensuche mit heuristischer Schätzfunktion durch, bei der – im Gegensatz zum Bergsteigerverfahren – kein Aufsteigen im Suchbaum mehr möglich ist, wenn man in einer Sackgasse angelangt ist: Der Bergsteiger merkt sich hier nicht wie beim regulären Bergsteigen bei der Expansion eines Knotens auch dessen weniger gute Nachfolger.⁷ Ist der Ausgangsverzweigungsgrad des Suchbaums beschränkt, so hat deshalb optimistisches Bergsteigen lediglich einen *konstanten Speicherbedarf* von

$$\text{Space(SHC)} = O(b),$$

wobei dann b der maximale Verzweigungsgrad ist.

Die optimistische Vorgehensweise der SHC-Suche hat natürlich auch ihren Preis. Um dies zu verdeutlichen, reinterpretieren wir das optimistische Bergsteigerverfahren als ein *Optimierungsproblem*, bei dem ein Knoten im Suchraum gefunden werden soll, für den die heuristische Schätzfunktion minimal wird. Enthält der Suchraum Zielknoten, die von einem der Startknoten aus erreichbar sind, so stellen diese Zielknoten (globale) Minima der Schätzfunktion mit der Bewertung 0 dar (heuristische Schätzfunktionen liefern ja immer nicht-negative Werte). Andernfalls sind die Lösungen des Optimierungsproblems Knoten, die aus Sicht der Schätzfunktion so nahe wie vom jeweiligen Startzustand aus erreichbar, an einem Zielknoten liegen.

Das optimistische Bergsteigerverfahren vollzieht damit auf der heuristischen Schätzfunktion im Wesentlichen eine diskrete Variante dessen, was ein Gradientenabstieg, beginnend beim Startknoten des Suchraums, leisten würde, wenn die Schätzfunktion differenzierbar wäre (was

⁶ Wird in englischsprachiger Literatur schlicht von „hill climbing“ (ohne den Zusatz „strict“ oder „with backtracking“) gesprochen, so ist damit fast immer das hier beschriebene „strict hill climbing“ gemeint.

⁷ Typischerweise wird sich ein menschlicher Bergsteiger diese Nachfolger dadurch leicht merken, dass er schrittweise mit der Suche eine immer feinere Karte des bereits explorierten Teils der Landschaft zeichnet.

bei einem per Definition diskreten Suchraum natürlich nicht möglich ist) und leidet damit auch an den gleichen drei Hauptproblemen wie das Gradientenabstiegsverfahren: lokale Minima, Plateaus und Talsohlen mit steilen Flanken.

Optimistisches Bergsteigen steuert immer dasjenige *lokale Minimum* an, in dessen Attraktionsbecken sich der Startpunkt der Suche befindet und kann dieses dann nicht mehr verlassen (engl. *foothill problem*), Dieses Minimum muß nicht gleichzeitig ein (Suchraum-)globales Minimum, also insbesondere kein Zielknoten sein. So kann optimistisches Bergsteigen z.B. beim Schiebepuzzle bei Verwendung der Fehlplazierungsheuristik einen einmal korrekt plazierten Stein nicht mehr (von seiner korrekten Position weg-)bewegen und so die Lösung des Puzzles unmöglich machen. Optimistisches Bergsteigen ist also selbst für endliche Suchräume im Allgemeinen nicht vollständig.

In Teilen des Suchraums, in denen sich der Wert der Schätzfunktion nicht ändert (sog. *Plateaus*) vollzieht optimistisches Bergsteigen im Wesentlichen blinde Suche und führt mangels jeglicher Anhaltspunkte nur zufällige lokale Richtungsänderungen aus. Deshalb ist man z.B. mit dem optimistischen Bergsteigerverfahren in flachen Wüstenlandschaften hoffnungslos verloren.

Die durch die heuristische Schätzfunktion definierte Landschaft enthält unter Umständen Täler mit steilen Flanken, die nur langsam in Richtung des nächstgelegenen Minimums abfallen. Die Sohlen solcher Täler können dann mit optimistischem Bergsteigen zwar schnell erreicht, aber nur schwer verfolgt werden, wenn es nicht zufällig Operatoren gibt, die ein exaktes Gehen entlang der Talsohle gestatten. Ein bekanntes Beispiel für dieses Problem ist der Zauberwürfel von Rubik, bei dem eine Lösungsstrategie darin besteht, zunächst eine Zustandsraumrepräsentation mit (Makro-)Operatoren zu konstruieren, die aus den elementaren Operatoren des naheliegenden, aber für das Suchverfahren suboptimalen Zustandsraumes (Drehen kompletter Würfelschichten) zusammengesetzt sind und dann eine Lösung des Problems (einheitliche Färbung jeder Würfelseite) mit einfacherem optimistischen Bergsteigen erlauben (für Details zu diesen sog. *Makrooperatoren* siehe z.B. [13]).

Randomisierte Verfeinerungen des optimistischen Bergsteigens

Eine naheliegende Möglichkeit, die genannten drei Probleme mit optimistischem Bergsteigen zu lindern, besteht darin, das Bergsteigerverfahren an einem zufällig gewählten neuen Punkt neu zu starten, sobald es an einem Punkt angelangt ist, von dem aus kein Fortschritt mehr erzielt wird (engl. random-restart hill climbing) und sich für die soweit erfolgten Suchvorgänge das bisher beste Endergebnis zu merken. Hat man genug Zeit für solche Neustarts, so wird man auf diese Weise früher oder später ein globales Minimum finden. Dabei benötigt man im Mittel umso mehr Durchgänge, je mehr lokale Minima in der durch die Schätzfunktion definierten Landschaft enthalten sind.

Einen anderen Weg zur Lösung der Probleme beschreitet das Verfahren des sog. *Simulierten Ausglühens* (engl. simulated annealing, [11]), bei dem nicht der Startpunkt zufällig variiert wird, sondern während der Suche mit einer bestimmten Wahrscheinlichkeit auch Schritte in eine Richtung unternommen werden dürfen, die *keine lokale Minimierung der geschätzten Distanz* bewirken. Auf diese Weise können lokale Minima mit einer im Verlauf der Zeit sinkenden Wahrscheinlichkeit wieder verlassen werden. Die Wahrscheinlichkeit wird proportional zu einer globalen Größe T , der sog. *Temperatur* gewählt, die von Iteration zu Iteration *langsam reduziert* wird. Irgendwann ist die Wahrscheinlichkeit dann so klein, dass das Verfahren nur

noch abwärts schreiten darf und sich wie das optimistische Bergsteigerverfahren verhält, also das nächste lokale und hoffentlich auch globale Minimum ansteuert.

Nachdem der zum simulierten Ausglühen gehörende stochastische Prozeß die Markov-Eigenschaft besitzt, kann man z.B. zeigen, dass das Verfahren *asymptotisch konvergiert*. Gilt etwa für die Temperatur T_k beim k -ten Schritt $T_k \geq \frac{T_0}{\log(1+k)}$, fällt die Temperatur also *sehr langsam* und ist T_0 eine hinreichend große Starttemperatur, so gelangt die Suche schließlich mit einer Grenz-Wahrscheinlichkeit von 1 in ein globales Minimum (siehe [6] – andere hinreichende Konvergenzbedingungen finden sich z.B. in [1]).

Das Verfahren bezieht seinen Namen nach einer Erfahrung beim Kochen von Stahl, wonach bekannt ist, dass man einen besonders guten und harten Stahl dadurch erhält, dass man den rohen Stahl anfangs stark erhitzt und dann langsam abkühlen lässt.

3.3.3 Bestensuche

Gierige Suche

Das Problem der lokalen Minima kann man auch dadurch lösen, dass man für die Entscheidung, welcher Knoten als nächstes zu expandieren ist, nicht nur die Bewertungen der Nachfolger des zuletzt expandierten Knotens heranzieht, sondern unter *allen Knoten des Saums* des soweit generierten Suchbaums (also der Agenda L) den heuristisch *global* optimalen Knoten nimmt. Dies leistet Algorithmus 3.7.

1. Sei L die Liste der Startknoten für das Problem.
2. Ist L leer, so melde einen Fehlschlag. Andernfalls wähle denjenigen Knoten n aus L , der dem Ziel *schätzungsgemäß am nächsten* ist.
3. Stellt n einen Zielknoten dar, so melde Erfolg und lieferne den Pfad vom Startknoten zu n .
4. Andernfalls entferne n aus L und füge alle seine Nachfolgeknoten in die Liste L ein. Markiere dabei wieder die neuen Knoten mit dem jeweils zugehörigen Pfad vom Startknoten.
5. Weiter mit Schritt 2!

Algorithmus 3.7: *Gierige Suche.*

Diese einfache Variante von *Bestensuche* nennen wir im Folgenden auch *gierige Suche* (engl. greedy search, GS), weil sie nur nach vorne, in die Richtung des aus agenda-lokal Sicht nächstgelegenen Ziels strebt, ohne dabei in Betracht zu ziehen, welche Kosten das Streben in die betreffende Richtung schon verursacht hat.

Wollten wir den im vorangegangenen Abschnitt gezogenen Bergsteigervergleich auch hier weiterspinnen, so müßten wir den Bergsteiger nicht nur (wie beim regulären Bergsteigen) mit Papier und Bleistift zur schritthalrenden Aktualisierung einer Karte der soweit explorierten Landschaft ausstatten, sondern ihm darüberhinaus auch noch eine Vorrichtung zum Beamen zur Verfügung stellen, wie sie aus der Science Fiction Serie „Enterprise“ bekannt ist: Er muß jetzt seinen nächsten Schritt ja jeweils an der, nach momentanem Wissen optimalen Stelle der so weit kartierten Landschaft machen und unabhängig davon, ob seine alte und die neue Position durch Täler und

Berge getrennt und vielleicht weit voneinander entfernt sind. Die von ihm verwaltete Landkarte wird damit im Allgemeinen auch nicht mehr einen einzigen zusammenhängenden Ausschnitt der Landschaft darstellen, sondern aus mehreren, eventuell getrennten Teilkarten bestehen.

Offensichtlich ist auch die gierige Suche im Allgemeinen nicht vollständig – die Argumentation aus Abschnitt 3.3.2 für das Bergsteigen (mit Zurücksetzen) lässt sich sinngemäß auf die gierige Suche übertragen. Unter der Annahme, dass man das Algorithmenskelett 3.7 durch Verwalten einer Liste besuchter Knoten zur Suche in eventuell zyklisch behafteten Graphen verfeinert und außerdem eine faire Schätzfunktion verwendet, kann die gierige Suche jedoch vollständig gemacht werden. Dabei heißt eine heuristische Schätzfunktion *fair* (für ein Suchproblem), wenn es im (zugehörigen) Suchraum zu einem beliebigen $k \geq 0$ nur endlich viele Knoten n gibt, für die $h(n) \leq k$ gilt und damit jeder Knoten nach endlicher Zeit vom Suchverfahren expandiert wird.⁸

Analog zu den Überlegungen aus dem vorangegangenen Abschnitt lässt sich auch die gierige Suche als ein Optimierungsverfahren betrachten – jedoch nicht mehr als eines mit lokaler schrittweiser Verbesserung. Gierige Suche wird also zielsicherer auf eine ggf. vorhandene Lösung zusteuern als die (ebenfalls vollständige) reguläre Bergsteigersuche. Sie hat dafür aber im Allgemeinen einen *wesentlich größeren* Speicherbedarf. Außerdem ist natürlich die Entscheidung selbst, welcher Knoten als nächster zu expandieren ist, zeitaufwendiger, da hierfür die ganze Agenda und nicht nur jeweils der Block der neuen Nachfolger sortiert werden muß.

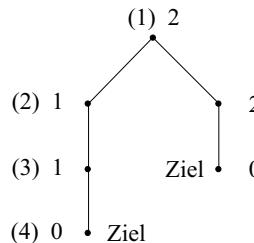
A*-Suche

Der A-Algorithmus

Wie das folgende Beispiel 3.3.1 zeigt, gibt es Suchprobleme, für die die gierige Suche keine optimale Lösung findet, wenn man eine Lösung als umso besser ansieht, je geringer die Kosten zu ihrer Herstellung sind. Diese Herstellungskosten haben wir bei den bis jetzt dargestellten informierten Suchverfahren, Einheitskosten für die Operatoren unterstellend, als die Anzahl der Operatoranwendungen definiert, mit der sie aus der Startsituation heraus erzeugbar sind. Im Falle baumartiger Suchräume stimmen diese Kosten also mit der Tiefe überein, in der die Lösungen im Suchbaum liegen.

Beispiel 3.3.1. (Gierige Suche findet nicht zuerst das Optimum)

In dem nachfolgenden Suchbaum findet gierige Suche zwar eine der beiden Lösungen, jedoch nur die schlechtere. In der Abbildung bezeichnen Zahlen in Klammern den Expansionszeitpunkt und solche ohne Klammern den jeweils geschätzten Abstand $h(n)$ eines Knotens n zum nächsten Zielknoten.



⁸ Für *endliche* Suchräume ist also jede Schätzfunktion fair.

Nachdem Kosten der genannten Art weder beim Bergsteigen noch bei der gierigen Suche thematisiert, geschweige denn für die Entscheidung herangezogen werden, welcher Knoten als nächstes zu expandieren ist, kann man eigentlich auch nicht erwarten, dass die beiden Verfahren immer kostenoptimale Lösungen liefern.

Abhilfe schafft hier – unter der weiteren, noch zu erläuternden Voraussetzung, dass eine zulässige Schätzfunktion eingesetzt wird – der sog. A-Algorithmus (siehe [9]). Dem A-Algorithmus liegt die folgende Auswahlmaxime zugrunde:

Expandiere den Knoten n zuerst, der (aus Sicht von n) *schätzungsgemäß am nächsten* an einem (gemäß d) *möglichst hochliegenden* Ziel liegt, d.h. den mit *minimalem* $f(n) := d(n) + h(n)$.

Suche mit dem A-Algorithmus lässt sich daher als Breitensuche verstehen, bei der die Größe $d(n) + h(n)$ als „Tiefe“ für die Auswahl des als nächstes zu expandierenden Knotens benutzt wird, in die neben der geschätzten Distanz $h(n)$ von n zum Ziel auch die Mindestkosten $d(n)$ für die Herstellung einer über n laufenden Lösung eingeht.

Im Allgemeinen verwendet man statt $d(n)$ die Schätzung $g(n)$ der tatsächlichen Kosten $g^*(n)$, die man benötigt, um n zu erreichen. $g(n)$ wird eventuell besser, während man den Suchraum parallel zur Suche generiert – siehe unsere spätere Diskussion des A*-Algorithmus für Graphen sowie unsere Erörterungen zu allgemeinen Suchräumen im Abschnitt 3.2.2 über Gleiche-Kosten-Suche.

Zulässige Schätzfunktionen

Solange nichts weiter über die im A-Algorithmus verwendete Schätzfunktion bekannt ist, kann nur wenig über das Verhalten des A-Algorithmus ausgesagt werden. Dies ändert sich schlagartig, wenn wir verlangen, dass Schätzfunktionen *zulässig* sein sollen. Eine Schätzfunktion h wird als *optimistisch* oder *zulässig* bezeichnet, falls

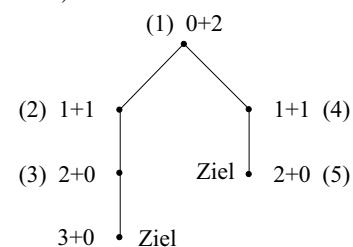
$$\forall n : h(n) \leq h^*(n)$$

gilt, die Distanz von einem Knoten n zum nächsten Zielpunkt also *nicht überschätzt* wird. Der Spezialfall des A-Algorithmus, bei dem eine zulässige Schätzfunktion zum Einsatz kommt, heißt **A*-Algorithmus**. Er wurde das erste Mal 1968 von Peter Hart, Nils J. Nilsson und Bertram Raphael beschrieben [9].

Wie man sich leicht überlegen kann, sind z.B. die beiden für das Schiebepuzzle vorgestellten Schätzfunktionen („Anzahl Fehlplazierungen“ bzw. „Manhattan-Distanz“) zulässig.

Beispiel 3.3.2. (Optimales Ergebnis bei zulässiger Schätzfunktion)

Rechts ist wieder der Suchraum aus Beispiel 3.3.1 zu sehen – diesmal aber mit einer zulässigen Schätzfunktion. Damit liefert der A-Algorithmus die optimale Lösung (5). In der Abbildung steht in einer Knotenbezeichnung der Form $(x)y + z$ die Größe x für den Expansionszeitpunkt, y für die Tiefe und z für die heuristische Bewertung des Knotens.



Das Bergsteigerverfahren findet im Suchraum zu Beispiel 3.3.2 das Optimum nur dann zuerst, wenn bei Expansion des Startknotens der rechte Nachfolgeknoten vor dem linken in die Agenda eingesortiert wird und die gierige Suche nur, falls von den beiden gleichbewerteten Nachfolgern des Startknotens zufällig der rechte vor dem linken aus der Agenda ausgewählt wird.

Dass der A-Algorithmus für das Beispiel 3.3.2 eine optimale Lösung findet, ist kein Zufall. Wie wir in Kürze zeigen werden, gilt vielmehr allgemein, dass der A-Algorithmus schon immer dann garantiert zuerst eine optimale Lösung berechnet, wenn die Schätzfunktion zulässig ist. Außerdem hat die Schätzfunktion einen wesentlichen Einfluss auf die Geschwindigkeit, mit der der A*-Algorithmus so eine optimale Lösung findet:

Lemma 3.3.1. *Sind zwei Schätzfunktionen h_1 und h_2 zulässig und gilt $\forall n : h_1(n) \leq h_2(n)$, dann exploriert der A*-Algorithmus mit h_2 nie mehr Knoten als mit h_1 (in so einem Fall sagt man, dass h_2 mehr Information als h_1 hat; für einen Beweis konsultiere man [5] bzw. [18]).*

Das eine Geschwindigkeitsextrem stellt hier der Fall $\forall n : h(n) = 0$ dar, bei dem A* zu Breitensuche (bzw. *Gleiche-Kosten-Suche* bei variablen Operatorkosten) entartet. Das andere Extrem resultiert, falls die Schätzfunktion zur Verfügung steht, die den Abstand zum nächsten Zielknoten immer exakt angibt; in diesem Fall steuert der A*-Algorithmus ohne Umwege, d.h. mit maximaler Geschwindigkeit, auf das nächstgelegene Ziel zu.

Es kann übrigens *keine zulässige* Schätzfunktion geben, die mit A*-Suche *Tiefensuche* realisiert, da Tiefensuche im Allgemeinen nicht die optimale Lösung eines Suchproblems findet, ja nicht einmal vollständig ist.

A*-Suche auf Graphen

Abbildung 3.8 zeigt eine Verallgemeinerung des A*-Algorithmus, die auch auf Suchräumen operieren kann, die nicht die Gestalt eines Baums haben (aus [18], mit kleinen Änderungen). Dieses Verfahren verwaltet neben einer Liste L von offenen Knoten (der Agenda bzw. open list) zusätzlich eine Liste C bereits erschlossener Knoten (die sog. closed list). Unseren Ausführungen in Abschnitt 3.2.2 folgend, wird von diesem Verfahren außerdem in Schritt 8 beim Einfügen eines Knotens in die Agenda überprüft, ob er oder einer seiner eventuell schon bekannten Nachkommen dort schon mit einem zu teuren Pfad vorhanden ist und ggf. entsprechend optimiert.

Der Algorithmus entwickelt zu diesem Zweck in G eine Graphen-Repräsentation des soweit explorierten Suchraums, den sog. *Explorationsgraphen*. Die Knotenmenge von G ist dabei zu jedem Zeitpunkt $L \cup C$ und die Kantenmenge besteht aus den vom Algorithmus berechneten Nachfolgekanten und Elternzeigern. G enthält einen ausgezeichneten Untergraphen T , den sog. *Explorationsbaum*. T besteht aus den gleichen Knoten wie G und ist durch die Elternzeiger aus Schritt 8 definiert: Abgesehen vom (hier der Einfachheit halber eindeutigen) Startknoten s verweist jeder Knoten n in G über seinen Elternzeiger auf genau einen seiner Vorgängerknoten – den Vater von n in T . Der Explorationsgraph G beinhaltet also alle soweit entdeckten Pfade von s zu einem beliebigen Knoten $n \in L \cup C$, wobei jeweils einer dieser Pfade (der mit den nach momentanem Kenntnisstand geringsten akkumulierten g -Kosten) durch die Elternzeiger in G (also T) ausgezeichnet wird.

1. Initialisiere L (die *open list*) mit dem Startknoten s für das Problem:

$$L := \{s\}.$$

2. Initialisiere die *closed list* C mit \emptyset und erzeuge einen Explorationsgraphen G , der nur s enthält.
3. Ist L leer, so melde einen Fehlschlag.
4. Andernfalls wähle denjenigen Knoten n aus L , für den

$$f(n) = g(n) + h(n)$$

minimal ist (dabei bezeichne $g(n)$ die Kosten des günstigsten Pfads vom Startknoten s zu n , der bis jetzt gefunden wurde), entferne n aus L und trage n in C ein.

5. Stellt n einen Zielknoten dar, so melde Erfolg und liefere durch Verfolgen der in G vorhandenen Elternzeiger von n nach s (siehe Schritt 8) den Pfad vom Startknoten s zu n .
6. Andernfalls expandiere n durch Generieren derjenigen seiner Nachfolger, die nicht schon Vorgänger von n in G sind und installiere die Knoten aus der so generierten Menge M als Kinder von n in G (Nachfolgekanten).
7. Etabliere in G für alle neuen Knoten aus M (also solche, die nicht schon in L oder C enthalten sind), einen Elternzeiger auf n und füge die neuen Knoten aus M in die Liste L ein.
8. Entscheide für jeden alten Knoten m aus M , ob sein Elternzeiger auf n geändert werden muß, weil der aktuell betrachtete Wurzelpfad von m (mit dem Vater n) billiger als der zum alten Elternzeiger von m gehörende Vater ist (Kostenkorrektur).
Verfahren entsprechend für die Nachkommen derjenigen alten Knoten aus M , die schon in C lagen und gerade eine Kostenkorrektur erfahren haben (rekursive Propagierung der Kostenkorrektur).
9. Weiter mit Schritt 3!

Algorithmus 3.8: *A*-Suche auf Graphen*.

Wird in Schritt 8 der Elternzeiger eines (alten) Knotens m aus M geändert, der schon in der closed list C steht, so bedeutet dies, dass soeben ein kostengünstigerer Pfad p für m gefunden wurde. Nachdem aber p Anfangsstück eines kostengünstigeren Pfads zu einem der Nachfolger von m in G sein kann, müssen dann ggf. auch die Elternzeiger einiger Nachfolger von m in G korrigiert werden.

Beispiel 3.3.3. (Kostenkorrekturen)

Abbildung 3.5 zeigt auf der linken Seite einen Explorationsgraphen, wie er aufgrund eines Suchprozesses entstanden sein kann, der mit s gestartet wurde und bei dem nun Knoten 1 zur Expansion ansteht: Gefüllte Knoten im Explorationsgraphen seien in C , hohle Knoten in L enthalten und die dicken Pfeile an den Nachfolgekanten (dargestellt mit dünnen Pfeilen) mögen für Elternzeiger stehen, die der *A*^{*}-Algorithmus soweit etabliert hat. Als Kosten für die Operatoren seien uniform Einheitskosten unterstellt.

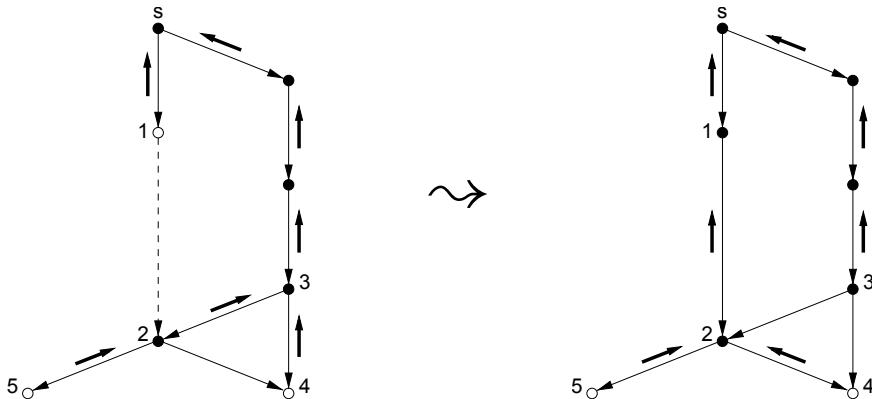


Abbildung 3.5: Aktualisierung des A*-Explorationsgraphen.

Bei der Expansion von Knoten 1 wird dessen einziges Kind, Knoten 2 erzeugt (gestrichelter Pfeil). Knoten 2 und auch sein T-Vater, der Knoten 3, sind alt und Knoten 2 liegt zudem in C und hat die Nachfolger 4 und 5. Der T-Vater von Knoten 4 ist vor der Expansion von Knoten 1 noch der Knoten 3.

Nach der Expansion von Knoten 1 läuft aber der kürzeste Pfad von Knoten s zum Knoten 2 nicht mehr über Knoten 3, sondern über Knoten 1. Also wird zunächst der T-Vater von Knoten 2 auf den Knoten 1 korrigiert. Damit müssen aber die Kosten für die Nachfolger von Knoten 2 neu berechnet und im vorliegenden Fall als Konsequenz der T-Vater von Knoten 4 auf den Knoten 2 korrigiert werden und es entsteht der auf der rechten Seite von Abbildung 3.5 gezeigte neue Explorationsgraph.

Das Propagieren der Kosten- und Elternzeiger-Änderungen im Explorationsgraphen G terminiert immer schon nach endlicher Zeit, da G zu jedem Zeitpunkt endlich und in Bezug auf die Nachfolgekanten, die ja die Propagierungsrichtung vorgeben, dank des Vorgängertests in Schritt 6 azyklisch ist.

Wendet man den A*-Algorithmus auf einen Suchbaum an, so kann man sich natürlich die closed list C und die Verwaltung der Elternzeiger sparen – es gibt dann in Schritt 8 keine alten Knoten in M und G und T fallen zusammen.

Wichtige Eigenschaften des A*-Algorithmus

Es liegt nun natürlich die Frage nahe, ob der A*-Algorithmus auch für Suchräume mit Zyklen und unendlich vielen Knoten optimal und sicher anwendbar ist. In den diesbezüglich folgenden Betrachtungen verwenden wir – wie wir das schon für die Funktionen g und h getan haben – die entsprechenden ungestörten Größen wieder für die Schätzungen ihrer gestörten Gegenstücke.

$$f^*(n) := g^*(n) + h^*(n)$$

bezeichnet also im Folgenden die (tatsächlichen) Kosten eines billigsten Pfades, der vom Startknoten s über n zu einem optimalen Zielknoten führt.

Wir beginnen unsere Überlegungen mit dem einfachen Fall, bei dem es sich bei dem Suchraum um einen endlichen, aber eventuell zyklen-behafteten Graphen handelt:

Lemma 3.3.2. *Der A*-Algorithmus auf Graphen terminiert für endliche Graphen.*

Bei jeder Iteration wird nämlich ein Knoten aus L entfernt und durch eine endliche Anzahl von *neuen* Nachfolgern (die nicht schon in der Agenda enthalten sind) ersetzt. In endlichen Graphen gibt es nur endlich viele neue Nachfolger. Also terminiert der Algorithmus nach endlicher Zeit entweder in Schritt 3 mit einem Fehlschlag ($L = \emptyset$) oder in Schritt 5 mit einem Erfolg (erster Knoten aus L ist eine Lösung).

Bezeichnen wir einen Pfad als optimal, wenn er mit den geringsten Kosten vom Startknoten zu einem besten Zielknoten führt, so lässt sich auch die folgende Eigenschaft des A*-Algorithmus leicht nachweisen:

Lemma 3.3.3. *Zu jedem Zeitpunkt bevor der A*-Algorithmus terminiert, gibt es auf L einen Knoten n' , der sich 1. auf einem optimalen Pfad vom Startknoten s zu einem besten Zielknoten befindet und für den 2. gilt: $f(n') \leq f^*(s)$.*

Ein ausführlicher Beweis, in den natürlich die Zulässigkeit der Schätzfunktion wesentlich ein geht, findet sich z.B. in [18].

Für den Fall unendlicher Graphen nehmen wir, eine reductio ad absurdum startend, an, dass der Suchgraph mindestens einen Zielknoten enthält, der A*-Algorithmus aber nicht terminiert. Dann kann die Terminierung nur dadurch verhindert werden, dass immer neue Knoten zu L hinzugefügt werden, weil immer längere, azyklische Pfade, die bei s beginnen, betrachtet werden. In diesem Fall würden aber sogar die kleinsten f -Werte von Knoten in L unmöglich groß wachsen, denn: Sei $d^*(n)$ die Länge des kürzesten Pfades im soweit generierten Explorationsgraphen, der von s nach n führt. Wir hatten in Abschnitt 3.2.2 vorausgesetzt, dass jeder Operator mindestens die positiven Kosten ϵ verursacht. Damit gilt $g^*(n) \geq d^*(n) \times \epsilon$ und wegen $g(n) \geq g^*(n)$ und $h(n) \geq 0$ sogar $f(n) \geq g(n) \geq d^*(n) \times \epsilon$. Insbesondere trifft dies für jeden Knoten $n \in L$ zu. Obwohl A* den Knoten aus L mit dem kleinsten f -Wert zur Expansion auswählt, wird dieser Knoten schließlich einen beliebig großen d^* - und damit auch einen beliebig großen f -Wert aufweisen, falls A* nicht terminiert. Diese Beobachtung zusammen mit Lemma 3.3.3 liefert den gewünschten Widerspruch und erhärtet damit gleichzeitig das folgende wichtige Korollar aus [18]:

Korollar 3.3.1. *Falls es einen Pfad vom Startknoten s zu einem Zielknoten gibt, so terminiert der A*-Algorithmus (auch für unendliche Graphen).*

Damit folgt auch die zentrale Eigenschaft des A*-Algorithmus:

Satz 3.3.1. *Der A*-Algorithmus ist zulässig, d.h. falls es einen Pfad von s zu einem Zielknoten gibt, so terminiert A* mit einem kosten-optimalen Pfad.*

Gibt es nämlich mindestens einen Zielknoten, so terminiert A* nicht mit einem Fehlschlag in Schritt 3 sondern in Schritt 5 mit einem Erfolg, weil L wg. Lemma 3.3.3 nicht leer werden kann (danach gibt es ja sogar einen Knoten in L , der auf einem optimalen Zielpfad liegt). Nehmen wir nun an, dass A* mit einem Zielknoten t terminiert, ohne einen optimalen Pfad gefunden zu haben. Dann müßte gelten: $f(t) = g(t) > f^*(s)$. Aufgrund von Lemma 3.3.3 gab es aber unmittelbar bevor A* terminiert hat, in L einen Knoten n' auf einem optimalen Pfad vom Startknoten s mit $f(n') \leq f^*(s)$, also gilt insgesamt $f(n') \leq f^*(s) < f(t)$. Damit hätte aber A* statt t den Knoten n' zur Expansion auswählen müssen; A* kann also nicht mit t terminiert haben und es folgt die Behauptung.

A*-Algorithmus bei Monotoniebeschränkung

Der A*-Algorithmus, so wie wir ihn skizziert haben, verwaltet den Explorationsgraphen G , um auch mit Suchräumen klarzukommen, die keine Baumgestalt haben, also Wegzusammenführungen oder gar Zyklen enthalten. Wie wir gleich sehen werden, lässt sich für Suchprobleme, bei denen für die heuristische Schätzfunktion und die Kosten der Kanten die sog. Monotoniebeschränkung gilt, die Verwaltung des Explorationsgraphen erheblich vereinfachen.

Für ein heuristisches Suchproblem gilt die *Monotoniebeschränkung*, wenn für zwei beliebige Knoten n und n' , von denen n' im Suchgraph Kind des Knotens n ist und $c(n \rightarrow n')$ die Kosten der Kante von n nach n' angibt, immer schon

$$h(n) - h(n') \leq c(n \rightarrow n')$$

zutrifft (also die geschätzte Distanz zum Ziel bei einem vorgegebenen Schritt nicht schneller schrumpft, als die Kosten durch diesen Schritt wachsen) und die heuristische Schätzfunktion (wie üblich) für alle Zielknoten den Wert 0 liefert.

Beispiele für Heuristiken, die die Monotoniebeschränkung erfüllen, sind die „Ich-bin-schon-da“-Heuristik und die einfache Fehlplazierungsheuristik (aus Beispiel 3.1.1) für das Schiebe-puzzle mit 8 Steinen, bei der $h(n)$ mit der Anzahl der in Stellung n fehlplazierten Steine übereinstimmt und Einheitskosten für die Züge unterstellt werden.

Eine heuristische Schätzfunktion, die die Monotoniebeschränkung erfüllt, ist offensichtlich schon zulässig. Außerdem lässt sich leicht zeigen, dass für solche Schätzfunktionen die vom A*-Algorithmus eingesetzte Priorisierungsfunktion f auf Wurzelpfaden des Suchraums monoton steigend ist. Nilsson zeigt darüberhinaus in [18] den folgenden Satz:

Satz 3.3.2. *Erfüllt die Schätzfunktion h die Monotoniebeschränkung, dann hat der A*-Algorithmus immer schon einen optimalen Pfad für jeden Knoten n gefunden, den er zur Expansion auswählt (es gilt dann also $g(n) = g^*(n)$).*

Unter der Voraussetzung der Monotoniebeschränkung muß der A*-Algorithmus also auch dann weder Pfadkosten aktualisieren, noch irgendwelche Elternzeiger im Explorationsgraphen G verwalten, wenn der Suchraum nicht die schlichte Gestalt eines Baumes aufweist. Aber Vorsicht: Um Terminierung zu garantieren, muß i. allg. leider trotzdem immer noch eine Liste der bereits erschlossenen Knoten verwaltet werden!

Eine unmittelbare Konsequenz dieses Satzes ist außerdem das folgende Korollar:

Korollar 3.3.2. *Erfüllt die Schätzfunktion h die Monotoniebeschränkung, so ist die Folge der f -Schätzwerte zu den Knoten, die vom A*-Algorithmus während der Suche der Reihe nach expandiert werden, monoton steigend.*

Es legt die folgende, genauso einfache wie nützliche Modifikation des A*-Algorithmus nahe, bei der man den Algorithmus eine wie folgt definierte Knotenmenge $K \subset L$ sowie die Größe

$$F := \text{Maximum aller } f\text{-Werte bereits expandierter Knoten}$$

verwalten lässt (offensichtlich gilt dann $F \leq f^*(s)$ und F ist mit 0 zu initialisieren). Erfüllt h die Monotoniebeschränkung *nicht* (und nur dann), so kann es sein, dass der A*-Algorithmus

einen Knoten n mit $f(n) < F \leq f^*(s)$ auf die open list L setzt. Sei K die Menge aller solcher Knoten (K ist also konstant leer, wenn die Monotoniebeschränkung gilt). Muß der A*-Algorithmus nun in Schritt 4 einen Knoten aus L zur Expansion auswählen, so gibt es zwei Möglichkeiten:

1. Fall: K ist zu diesem Zeitpunkt leer.

Dann wird, wie gehabt, ein Knoten $n \in L$ mit minimalem f -Wert zur Expansion ausgewählt (in diesem Fall gilt natürlich $F \leq f(n) \leq f^*(s)$ und F muß ggf. auf $f(n)$ korrigiert werden).

2. Fall: K ist nicht leer.

Dann muß der A*-Algorithmus früher oder später jeden der Knoten aus K zur Expansion auswählen (weil jeder Knoten $n \in K$ auch in L liegt und für ihn $f(n) < f^*(s)$ gilt), d.h. man kann getrost den Knoten $n \in K$ als nächsten zur Expansion bestimmen, der den minimalen g -Wert, aber nicht notwendig den minimalen f -Wert hat (und muß F anschließend nicht aktualisieren).

Diese, bereits in [18] beschriebene (nur bei Nichtzutreffen der Monotoniebeschränkung wirksame) Variante des A*-Algorithmus erhöht für den Fall, dass die Monotoniebeschränkung nicht gilt (und deshalb Elternzeiger verwaltet und Pfadkosten aktualisiert werden müssen), die Chance, dass bereits der erste zu einem Knoten gefundene Pfad der optimale Pfad ist und sorgt damit häufig für eine Reduktion des Aufwandes für rekursive Kostenaktualisierungen, die vom Algorithmus ggf. in Schritt 8 durchzuführen sind.

Speicherbeschränkte Varianten der A*-Suche

Wie wir in Abschnitt 3.3.3 gesehen haben, wird A*-Suche für die schlechteste aller zulässigen Schätzfunktionen (also die „Ich-bin-schon-da“-Heuristik $h(n) = 0$, die für alle Knoten n schätzt, man sei bereits am Ziel) zur Gleiche-Kosten-Suche, die ihrerseits für uniforme Kosten zur Breitensuche degeneriert. Der A*-Algorithmus entwickelt also im ungünstigsten Fall den selben unangenehmen Speicherhunger wie die Breitensuche.

Dieses Problem läßt sich aber mit dem gleichen Kunstgriff entschärfen, mit dem schon der Breitensuche ihr Speicherproblem genommen wurde. Dazu führt man im Wesentlichen eine schrittweise vertiefende Suche durch, bei der jedoch als maximale Tiefenbeschränkung die Größe $f(n)$ anstelle der Größe $g(n)$ verwendet wird. Der so modifizierte Algorithmus heißt *A*-Suche mit schrittweiser Vertiefung* (engl. iterative deepening A*, IDA* – siehe [12]) und ist in Abbildung 3.9 für die Nutzung auf Suchbäumen dargestellt.

A*-Suche mit schrittweiser Vertiefung hat aufgrund ihrer Konstruktion offensichtlich den gleichen Speicherbedarf wie Tiefensuche

$$\text{Space(IDA}^*) = O(d).$$

Trotzdem expandiert sie genau die gleichen Knoten, die auch von A*-Suche expandiert würden und liefert immer eine kostenoptimale Lösung, solange nur die Schätzfunktion h zulässig ist.

Nachdem A*-Suche mit schrittweiser Vertiefung zwischen je zwei Iterationen nur die aktuelle f -Kosten-Schranke c memoriert, werden vor allem in Suchräumen, die nicht Baumform haben, viele Expansionen mehrfach vorgenommen – für solche Suchräume wird also der sparsame Umgang mit Speicher mit einem gestiegenen Zeitbedarf bezahlt. A*-Suche kann jedoch so

1. Sei $c = 1$ (c steht für die maximale Suchtiefe).
2. Sei L die Liste der Startknoten für das Problem.
3. Ist L leer, so erhöhe c um 1 und mache weiter mit Schritt 2!
Andernfalls sei n der erste Knoten in L .
4. Stellt n einen Zielknoten dar, so melde Erfolg und liefere den Pfad vom Startknoten zu n .
5. Andernfalls entferne n aus L .
Füge an den Anfang von L diejenigen Nachkommen n' von n an, für die gilt:

$$f(n') \leq c.$$
- Markiere dabei die neuen Knoten jeweils mit dem zugehörigen Wurzelpfad, der beim Startknoten beginnt.
6. Weiter mit Schritt 3!

Algorithmus 3.9: *A*-Suche mit schrittweiser Vertiefung (IDA*-Suche)*

modifiziert werden, dass sie im Rahmen des verfügbaren Speichers so wenig Expansionen wie möglich durchführt, vollständig ist, wenn der verfügbare Speicher ausreicht, um den optimalen Lösungspfad zu merken und zeitoptimal, wenn genug Speicher für den kompletten Suchraum zur Verfügung steht. Die entsprechende Variante heißt SMA*-Suche (engl. simplified memory-bounded A*). Weil uns hier der Raum zu einer detaillierten Darstellung der SMA*-Suche fehlt, verweisen wir den Leser stattdessen auf [21]), wo das Verfahren samt seiner Eigenschaften ausführlich erörtert wird.

Literaturverzeichnis

- [1] Aarts, E. und Korst, J. (1989). *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. Wiley, New York.
- [2] Barr, A. und Feigenbaum, E. A. (1981). *The Handbook of Artificial Intelligence*, volume 1. William Kaufmann, Inc.
- [3] Beckstein, C. (1996). *Begründungsverwaltung: Grundlagen, Systeme und Algorithmen*. Teubner-Texte zur Informatik. Teubner-Verlag, Leipzig.
- [4] Berlekamp, E. R., Conway, J. H., und Guy, R. K. (1985 und 1986). *Gewinnen: Strategien für mathematische Spiele*, volume 1–4. Vieweg, Braunschweig.
- [5] Gaschnig, J. (1979). Performance measurement and analysis of certain search algorithms. Technical Report CMU-CS-79-124, Carnegie-Mellon University.
- [6] Geman, S. und Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-6*, pages 721–741.
- [7] Ginsberg, M. L. (1993). *Essentials of Artificial Intelligence*. Morgan Kaufmann.
- [8] Ginsberg, M. L. und Harvey, W. D. (1990). Iterative broadening. In *Proc. of the Eighth National Conference on Artificial Intelligence*, pages 216–220.

- [9] Hart, P., Nilsson, N., und Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on SCC*, 4.
- [10] Hopcroft, J. E., Motwani, R., und Ullman, J. D. (2006). *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading, MA, third edition.
- [11] Kirkpatrick, S., Gelatt, D., und Vercchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671–680.
- [12] Korf, R. E. (1985a). Depth-first iterative deepening: An optimal admissible tree search. *Artificial Intelligence*, 27:97–109.
- [13] Korf, R. E. (1985b). Macro-operators: A weak method for learning. *Artificial Intelligence*, 26:35–77.
- [14] Langley, P. (1992). Systematic and nonsystematic search strategies. In *Artificial Intelligence Planning Systems: Proc. of the First International Conference*, pages 145–152, San Mateo, CA. Morgan Kaufmann.
- [15] McCarthy, J. (1980). Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39.
- [16] Michalewicz, Z. und Vogel, D. B. (2010). *How to Solve It: Modern Heuristics*. Springer Verlag, second, revised and extended edition.
- [17] Minton, S., Johnston, M. D., Philips, A. B., und Laird, P. (1990). Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In *Proc. of the Eighth National Conference on Artificial Intelligence*, pages 17–24.
- [18] Nilsson, N. J. (1982). *Principles of Artificial Intelligence*. Springer-Verlag.
- [19] Pearl, J. (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, Reading, MA.
- [20] Polya, G. (1973). *How to Solve it*. Princeton University Press, New Jersey, second edition.
- [21] Russell, S. und Norvig, P. (2010). *Artificial Intelligence, A Modern Approach*. Prentice Hall Series in Artificial Intelligence, third revised edition.
- [22] Selman, B., Levesque, H., und Mitchell, D. (1992). A new method for solving hard satisfiability problems. In *Proc. of the Tenth National Conference on Artificial Intelligence*.
- [23] Stallman, R. und Sussman, G. (1977). Forward reasoning and dependency directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence*, 9:135–196.

4 Wissensrepräsentation und -verarbeitung

Bernhard Nebel und Stefan Wölfl

4.1 Einleitung und Motivation

Wissensrepräsentation und -verarbeitung (engl. *knowledge representation and reasoning*) ist eines der Kerngebiete der Künstlichen Intelligenz. In diesem Kapitel wollen wir einige der grundlegenden Annahmen dieses Gebiets präsentieren und Techniken sowie Methoden der Wissensrepräsentation skizzieren.

4.1.1 Wissen – wozu?

Wir leben in einer Wissensgesellschaft. Diese Aussage wird häufig angeführt, um die Bedeutung von Wissen für die moderne Gesellschaft hervorzuheben. Etwas konkreter besagt die Aussage, dass „Wissen“ eine essentielle Ressource für die sozialen wie ökonomischen Prozesse in hoch entwickelten Gesellschaften darstellt. Wenn in diesem Zusammenhang von *Wissen* die Rede ist, so in einem weiten Sinne des Wortes: „Wissen“ meint hier nicht nur reines Faktenwissen, sondern subsumiert eine ganze Bandbreite von wissensbasierten Kontexten, etwa wissenschaftliche und industrielle Forschung, Technologien und Verfahrenstechniken, Bildung und Lernkompetenzen. Die Bedeutung von Wissen wird auch daran deutlich, dass Wissensinhalte heute in einem weltumspannenden Informationsnetzwerk, dem *World Wide Web (WWW)*, jederzeit Millionen von Nutzern, Web-Diensten und künstlichen Agenten zur Verfügung stehen. Ein eindrucksvolles Beispiel stellt die Online-Enzyklopädie *Wikipedia* dar, die allein in ihrer deutschsprachigen Ausgabe mehr als 1,4 Millionen Einträge (Stand Mai 2012) umfasst und täglich um etwa 400 neue Artikel anwächst [35].

Aus Sicht der Informatik und speziell der Künstlichen Intelligenz stellt sich daher die Frage, wie diese Wissensinhalte so repräsentiert werden können, dass aus der großen Zahl an Informationsressourcen und -quellen neue Wissensinhalten extrahiert, aggregiert und schließlich wieder anderen Nutzern zur Verfügung gestellt werden können. Kurz gesagt benötigt man *Technologien* zur Repräsentation und Verarbeitung von Wissen. Der technologische Aspekt ist hierbei wichtig: wie wir in diesem Kapitel sehen werden, beeinflusst die Wahl der Repräsentation (oder wie wir später sagen werden, des *Repräsentationsformalismus*) entscheidend, wie leicht oder wie schwierig sich bestimmte Problemstellungen bei der Wissensverarbeitung lösen lassen. Damit hängt die Güte eines Repräsentationsformalismus auch davon ab, welche Art von Problemen in einer gegebenen Anwendung gelöst werden soll. Im englischen Sprachgebrauch hat sich daher auch der Term *knowledge engineering* eingebürgert, der diesen technologischen Aspekt des Forschungsgebiets Wissensrepräsentation und -verarbeitung unterstreicht.

Neben der Frage, wie wir Wissensinhalte repräsentieren können, sind in der praktischen Anwendung natürlich auch Probleme der Verwaltung, der Vorhaltung und Sicherung von Wissensinhalten (engl. *knowledge management*) von Bedeutung, auch wenn wir dieses Thema im Rahmen dieses einführenden Kapitels nicht vertiefen können. Eine typische Fragestellung des Wissensmanagements ergibt sich zum Beispiel daraus, dass viele Wissensinhalte auch zukünftigen Generationen (von Menschen und Technologien) zur Verfügung stehen sollen. Dies stellt insofern ein Problem dar, als Wissensinhalte, wie sie etwa im WWW zur Verfügung stehen, „kurzlebig“ sind, d.h. es ist absehbar, dass diese Inhalte aktualisiert werden (und damit die alten Inhalte nicht mehr zugänglich sind), dass Web-Ressourcen durch andere ersetzt werden, dass die Art der Wissensrepräsentation sich verändert, etc.

Wir haben diesen Abschnitt mit der Frage „Wissen – wozu?“ überschrieben. Im Grunde versteht sich eine Antwort auf diese Frage von selbst. Wissen benötigen wir nicht nur, um etwa neue Informationen oder Ereignisse zu verstehen und zu bewerten, sondern auch, wenn wir uns in einer gegebenen Situation entscheiden müssen, wie wir handeln sollen (und wollen). Je *informierter* unsere Entscheidungen sind, desto wahrscheinlicher werden wir in der Regel unsere Ziele verwirklichen können. Offensichtlich spielen hierbei ganz verschiedene Formen von Wissen eine Rolle, zum Beispiel Wissen, das uns hilft, die gegebene Situation einzuschätzen, Wissen über die Alternativen, die man gerade hat, und Wissen darüber, mit welchen Handlungen gesteckte Ziele erreicht werden können. Damit dieses (Hintergrund-)Wissen, also z.B. faktisches Wissen, kausales Wissen, Wissen über Normen und Gesetze, aber bei Entscheidungen einfließen kann, ist es notwendig, dieses Wissen zu verarbeiten. Was folgt etwa aus unserem generellen Hintergrundwissen für die spezielle Situation, in der wir eine Entscheidung zu treffen haben? Wie können wir neue Erfahrungen auf der Basis unserer Wissensannahmen erklären? Wie integrieren wir neues Wissen in unsere bestehenden Annahmen? Und, ist all das, was wir zu wissen glauben, denn überhaupt miteinander konsistent?

Solche und ähnliche Fragen sind typische Problemstellungen, die auf dem Gebiet der Wissensrepräsentation und Wissensverarbeitung betrachtet werden. Wie wir im Weiteren ferner sehen werden, lassen sich viele dieser Probleme einer Lösung zuführen, indem man spezielle Inferenzmethoden entwickelt, mit denen sich neue Wissensinhalte aus bestehenden Wissensannahmen generieren oder herleiten lassen. In diesem Sinne sind Themen der Wissensrepräsentation und Wissensverarbeitung typischerweise eng mit Theorien des Schlussfolgerns und somit mit formalen Logiksystemen verknüpft.

4.1.2 Wissensformen

Wir haben bisher etwas vage von Wissen und Wissensinhalten gesprochen, ohne eine genaue Definition dieser Begriffe anzugeben. Was aber ist überhaupt Wissen? Um diese Frage zu beantworten, ist es sinnvoll, den Wissensbegriff zu klassifizieren, d.h. rein begrifflich, terminologisch, verschiedene Arten von Wissen zu unterscheiden.

Ein erstes Unterscheidungsmerkmal betrifft den Gegenstand des Wissens, also die Art des Wissensinhalts (dessen, was gewusst wird). Von einem *deskriptiven Wissen* (oder auch *propositio-nalem Wissen*) spricht man, wenn der Gegenstand des Wissens eine Proposition oder Sachverhalt ist. Sprachlich drücken wir deskriptives Wissen in der Form „*X* weiß, dass *P*“ oder auch in

der Form „*X weiß, ob P*“ aus, wo *X* der Wissensträger und *P* eine Proposition ist.¹ Wenn man nun sagt, dass eine Person *weiß, dass* eine Proposition *P* wahr ist, so wird man dieser Person zunächst einmal zusprechen, dass sie davon überzeugt ist, dass *P* der Fall ist. Im gewöhnlichen Sinne des Wortes wird man ferner davon ausgehen, dass die Person sich in dieser Überzeugung nicht irrt, d.h. dass *P* auch tatsächlich der Fall ist, die Proposition *P* also wahr ist.²

Vom deskriptiven Wissen ist das *prozedurale Wissen* zu unterscheiden. Eine Person kann zum Beispiel wissen, dass sich eine Festplatte in Partitionen aufteilen lässt, ohne zu wissen, *wie* sich die Festplatte in Partitionen aufteilen lässt. Ein Informatikstudent im Grundstudium mag vielleicht das Pumping Lemma für reguläre Sprachen kennen, aber nicht imstande sein, dieses Lemma auch anzuwenden, um für eine gegebene formale Sprache zu zeigen, dass sie nicht regulär ist. Umgekehrt kann eine Person wissen, *wie* man im Prinzip prüft, ob eine gegebene Zahl eine Primzahl ist, auch wenn sie nicht weiß, dass die Zahl 2311 eine Primzahl ist. In Gegensatz zum propositionalen Wissen basiert prozedurales Wissen also auf Fertigkeiten und Kompetenzen, die man erlernt und eingeübt hat.

Beide bisher besprochenen Wissensformen lassen sich nun jeweils weiter unterscheiden in einerseits *explizites* und andererseits *implizites* Wissen. Von einem *expliziten* Wissen spricht man dann, wenn der Wissensinhalt in einer sprachlich fixierten Form vorliegt und damit insbesondere auch kommuniziert werden kann. Andernfalls spricht man von *implizitem Wissen*.

Im Falle deskriptiven Wissens ist die Unterscheidung „explizit“ vs. „implizit“ ganz offensichtlich. Eine Person, die explizit weiß, dass die Summe der Innenwinkel in einem beliebigen Dreieck 180° beträgt, muss nicht auch explizit wissen, dass jeder Innenwinkel in einem gleichseitigen Dreieck gerade 60° beträgt, zum Beispiel deshalb, weil sie nicht weiß, dass die gleichseitigen Dreiecke gerade die gleichwinkligen Dreiecke sind, oder einfach deshalb, weil sie sich noch nie überlegt hat, dass aus der Tatsache, dass die gleichseitigen Dreiecke gerade die gleichwinkligen sind, folgt, dass jeder Innenwinkel gerade 60° beträgt, sie also diesen Sachverhalt nicht explizit, sondern nur implizit weiß.

Prozedurales Wissen ist häufig, aber nicht immer, implizit. Selbst langjährigen Autofahrern wird es zum Beispiel schwer fallen, explizit zu erläutern, *wie* man ein Auto fährt; einer Fahrschullehrerin dagegen sollte dies möglich sein, d.h. sie sollte in der Lage sein, Subroutinen des Autofahrens zu erklären, also sprachlich zu vermitteln. Typische Beispiele für explizites, prozedurales Wissen begegnen uns in Gebrauchsanweisungen oder in Kochrezepten.

Wenn der Wissensträger nicht nur eine einzelne Person, sondern eine Gruppe von Personen (oder künstlichen Agenten) ist, so lassen sich ferner eine Reihe von besonderen Wissensbegriffen unterscheiden. Typische Beispiele sind die Begriffe *verteiltes* und *gemeinsames* Wissen.

¹ Tatsächlich lässt sich „wissen, ob“ auf „wissen, dass“ zurückführen. Dass ein Agent *X* weiß, ob *P* der Fall ist, heißt ja nicht anderes als, dass *X* weiß, dass *P* der Fall ist, oder weiß, dass *P* nicht der Fall ist.

² Zur Frage, ob diese Gleichsetzung „Wissen = wahre Überzeugung“ einer sinnvolle Explikation des Wissensbegriffs ist, gibt es in der philosophischen Literatur eine ausführliche Diskussion. Manche Argumente lassen diese Gleichsetzung als *zu eng* erscheinen. So ist es durchaus sinnvoll zu sagen, dass ein Student weiß, dass die Lichtgeschwindigkeit 299 792 458 m/sec beträgt, obgleich diese Aussage einen empirischen Sachverhalt darstellt und sich somit als falsch herausstellen kann (zumindest dann, wenn man die Aussage nicht als eine Folgerung aus einer Definition der Maßeinheit Meter mittels der Lichtgeschwindigkeit ansieht). Andere Argumente weisen dagegen die Definition von Wissen als wahre Überzeugung als *zu weit* aus. Hintergrund ist hier, dass man Überzeugungen, die eine Person nur zufälligerweise hat, für die die Person also zum Beispiel keine Begründung hat, nicht als Wissen auszeichnen möchte. Zu einer eingehenden Diskussion siehe z.B. [15, 33].

Verteiltes Wissen ist eine spezielle Form des impliziten Wissens. Man kann zum Beispiel sagen, dass eine Gruppe eine Proposition weiß, wenn diese Proposition aus der Gesamtheit dessen folgt, was die einzelnen Mitglieder der Gruppe wissen. Wenn also eine Person weiß, dass Erde, Merkur, Venus, Mars, Jupiter, Saturn, Uranus und Neptun Planeten in unserem Sonnensystem sind, aber unsicher ist, ob Pluto ein Planet ist, eine andere aber Person weiß, dass unser Sonnensystem acht Planeten hat, diese aber nicht aufzählen kann, so weiß die Gruppe dieser beiden Personen (im Sinne der Begriffs verteilten Wissens), dass unser Sonnensystem genau aus den genannten Planeten besteht, obwohl keine der beiden Personen dies (explizit oder implizit) weiß. Von einem *gemeinsamen Wissen* in einer Gruppe spricht man dagegen dann, wenn jeder in der Gruppe diesen Sachverhalt weiß und darüber hinaus auch weiß, dass alle anderen in der Gruppe den Sachverhalt wissen.³

Der Bereich des deskriptiven Wissens lässt sich nun nochmals weiter untergliedern, wenn man berücksichtigt, von welcher Art die gewussten Sachverhalte sind. Die einfachste Art von Propositionen sind solche, die einem Objekt eine bestimmte Eigenschaft zusprechen („Dieser Ball ist rot“) oder aber ausdrücken, dass mehrere Objekte in einer bestimmten Relation zueinander stehen („Dieser Ball ist kleiner als jener“). Bezieht sich Wissen auf solche Sachverhalte, so sprechen wir von *faktischem Wissen* (oder kurz: *Faktenwissen*). Neben Faktenwissen unterscheidet man häufig *terminologisches* und *konditionales/kausales Wissen*. Terminologisches Wissen (oder auch *begriffliches Wissen*) ist dadurch charakterisiert, dass die Wahrheit des gewussten Sachverhalts allein davon abhängt, wie bestimmte Begriffe in einer Sprache, besser in einer Sprachgemeinschaft (z.B. einer Expertengruppe), verwendet werden. Beispiele hierfür sind etwa „Junggesellen sind unverheiratet“, „Pinguine sind Vögel“, „Pilze sind weder Pflanzen noch Tiere“.

Von konditionalem Wissen spricht man dann, wenn der Wissensinhalt eine Wenn-dann-Beziehung zum Gegenstand hat, also Sachverhalte, die sich ausdrücken lassen durch Aussagen wie z.B. „Wenn die Komponente X des Systems S ausfällt, so fällt das System S selbst ebenfalls aus“ oder „Wäre die Komponente X des Systems S nicht ausgefallen, so wäre auch das System S nicht ausgefallen“. Unter kausalem Wissen versteht man Wissen, das sich auf kausale Sachverhalte bezieht, deren Bestehen also von der Gültigkeit kausaler Beziehungen abhängt und die sich daher durch Aussagen ausdrücken lassen, in denen von Ursachen, Wirkungen, kausalen Korrelationen oder Abhängigkeiten die Rede ist. Oftmals sind kausale Sachverhalte eine spezielle Form konditionaler Sachverhalte, etwa in „Übergewicht begünstigt Bluthochdruck“ oder „Der Ausfall der Komponente X könnte eine Ursache für den Ausfall des Systems S sein“. Damit ist die Liste der verschiedenen Wissenstypen (aufgeschlüsselt nach der Art der gewussten Sachverhalte) aber bei Weitem nicht abgeschlossen: es gibt normatives Wissen, moralisches und ästhetisches Wissen, Wissen über Präferenzen, temporales Wissen, räumliches Wissen, usw.

Schließlich kann man deskriptives Wissen auch danach charakterisieren, wie sich die Wahrheit des Wissensinhalt überprüfen lässt. So ist etwa empirisches Wissen ein Wissen von Sachverhalten, deren Bestehen sich durch Beobachtung, Experiment, oder andere auf Erfahrung basierenden Verfahren überprüfen lässt. Dagegen ist theoretisches Wissen keiner direkten empirischen Verifikation zugänglich: theoretische Annahmen lassen sich nur falsifizieren, etwa in einem Experiment, das zeigt, dass sie mit den gemachten experimentellen Ergebnissen unverträglich

³ Die logische Analyse des gewöhnlichen Wissensbegriffs wie auch der Begriffe *verteiltes* und *gemeinsames Wissen* ist Gegenstand der *epistemischen/doxastischen Logik*. Als eine gut lesbare Einführung in die epistemische Logik verweisen wir auf [11].

sind. Empirisches wie theoretisches Wissen ist synthetisch, d.h. es könnte sich im Prinzip als falsch herausstellen. Im Gegensatz dazu ist analytisches Wissen ein Wissen von Sachverhalten, die sich vollständig unabhängig von empirischen Ergebnissen verifizieren (z.B. durch Angabe eines formalen, mathematischen Beweises) lassen. Insbesondere wird man terminologisches Wissen als eine Form analytischen Wissens auffassen.

4.1.3 Repräsentation

Damit ein künstlicher Agent oder ein wissensbasiertes System Wissen verarbeiten kann, muss dieses Wissen zunächst in irgendeiner Form repräsentiert sein. Ein künstlicher Agent, der Skat spielt, muss die grundlegenden Spielregeln des Skat „beherrschen“ und auch über geeignete Funktionen verfügen, die die gegebene Spielsituation bewerten und es somit dem Spieler erlauben, vernünftige (erfolgswahrscheinliche) Aktionen auszuführen (etwa beim Reizen oder beim Ausspielen einer Karte). Tatsächlich wird ein Skatspieler Wissen über das Skatspiel oftmals nur implizit repräsentieren (z.B. in der Implementierung der Spielregeln oder der Bewertungsfunktion). Während ein künstlicher Skat-Agent aber einfach nur gut Skat spielen soll, werden an ein Expertensystem ganz andere Anforderungen gestellt. Ein solches System soll Anfragen beantworten können, wobei aus den im System explizit oder implizit repräsentierten Informationen „neues“, explizites Wissen *generiert* wird. Ein medizinisches Expertensystem, das die verschiedenen Körperorgane des Menschen sowie die medizinisch relevanten (z.B. funktionale und dysfunktionale) Beziehungen zwischen diesen Organen abbildet, sollte Anfragen über den Blutkreislauf beantworten können, auch wenn die Antworten selbst nicht schon explizit im System abgelegt sind. Die Idee wissensbasierter Systeme (wie etwa Expertensysteme) ist nun, solche Anfragen unter Benutzung eines Korpus von explizit repräsentierten Wissensinhalten, einer sogenannten *Wissensbasis*, zu verarbeiten.

Eine explizite Repräsentation, wie sie in einer Wissensbasis verwendet wird, kann man verstehen am Beispiel einer thematischen Karte, die einen bestimmten geographischen Aspekt, zum Beispiel die durchschnittlichen Tagestemperaturen aufgeschlüsselt nach Regionen, abbildet. Eine solche thematische Karte stellt eine Repräsentation dar: Regionen werden durch zweidimensionale Flächen repräsentiert, Einfärbungen dieser Regionen repräsentieren die durchschnittlichen Tagestemperaturen (entsprechend einer fest gewählten Klassifikation von Temperaturwerten).

Wie dieses einfache Beispiel bereits zeigt, geht es bei einer Repräsentation um die Darstellung eines bestimmten Weltausschnittes, d.h. eine Repräsentation beruht auf einer Auswahl von Objekten und einer Auswahl von zu repräsentierenden Aspekten. Wenn wir also eine bestimmte Domäne D von Entitäten repräsentieren wollen, so müssen wir zunächst auch angeben, welche der auf D gegebenen Eigenschaften, Relationen, Operationen von Interesse sind und daher repräsentiert werden sollen. Ist eine solche Menge \mathcal{F} von zu repräsentierenden Relationen (aus Einfachheitserwägungen beschränken wir uns hier auf Relationen) identifiziert, so benötigen wir für eine Repräsentation eine Menge von Repräsentanten und für jede Relation F aus \mathcal{F} eine Repräsentation dieser Relation, die auf der Menge der Repräsentanten definiert ist. Will man zum Beispiel die Elternbeziehungen („ist Mutter von“ und „ist Vater von“) in einer Mäusepopulation repräsentieren, so kann man analog zu einer Ahnentafel einen gerichteten Graphen angeben, der eine solche Repräsentation liefert: die Knoten des Graphen repräsentieren die Mäuse; eine gerichtete Kante zwischen zwei Mäusen repräsentiert die Elternbeziehung und um „Mutter-“ und

„Vater-“ Maus unterscheiden zu können, versehen wir die gerichteten Kanten noch mit jeweils unterschiedlichen Farben oder versehen sie mit einer Beschriftung „Mutter“ oder „Vater“.

Auch wenn wir bei dieser graphentheoretischen Repräsentation gerichtete (und gefärbte) Kanten verwendet haben, um eine bestimmte Relation zu repräsentieren, so ist doch klar, dass die Kanten wieder eine Relation definieren, nun nicht zwischen Mäusen, sondern zwischen den Knoten des Graphen, die die Mäuse aus unserer Population repräsentieren. Wir können daher unter einer Repräsentation einer Domäne $(D, \{F_i\}_{i \in I})$ eine Struktur $(S, \{F'_i\}_{i \in I}, R)$ verstehen, wobei S eine Menge von Repräsentanten ist, F'_i eine zur Relation F_i assoziierte Relation auf S ist und $R \subseteq D \times S$ eine Repräsentationsrelation zwischen Domänenentitäten und Repräsentanten ist. An diese Repräsentationsstruktur wird man sinnvollerweise weitere Forderungen stellen: (a) ein Element der Domäne wird durch höchstens einen Repräsentanten repräsentiert, verschiedene Domänenentitäten haben verschiedene Repräsentanten und jeder Repräsentant repräsentiert auch etwas, (b) wann immer Domänenentitäten $x_1, \dots, x_n \in D$ Repräsentanten besitzen und in einer der Eigenschaften aus $F_i \in \mathcal{F}$ zueinander stehen, so sollen die Repräsentanten in der dazu assoziierten Relation F'_i zueinander stehen, und umgekehrt (entprechendes gelte für Funktionen).

In obigem Beispiel hatten wir eine graphentheoretische Repräsentation betrachtet. Eine solche Repräsentation hat Vorteile: sie ist nicht nur einfach, sondern erlaubt es unmittelbar, graphentheoretische Algorithmen zu verwenden, um zum Beispiel die Menge aller Vorfahren einer Maus in der Population bestimmen zu können. Sie hat aber auch Nachteile. So verliert man die Einheitlichkeit der Darstellung sofort, wenn zum Beispiel nicht nur zweistellige Relationen, sondern höherstellige Relationen dargestellt werden sollen. Ferner ist man oftmals nicht nur an der Repräsentation einer spezifischen Situation, sondern allgemeiner an einem Repräsentationsschema interessiert, das sich einfach auf unterschiedliche Kontexte anwenden lässt. Daher liegt es nahe, anstelle graphentheoretischer Repräsentationen symbolische Repräsentationen zu betrachten, bei denen Domänenentitäten und die Relationen zwischen ihnen durch Symbole repräsentiert werden. Eine besondere Form der symbolischen Repräsentation, nämlich die Repräsentation mittels logischer Formalismen, werden wir im zweiten Kapitel besprechen.

4.1.4 Wissensverarbeitung = Schlussfolgern

Wenn wir von Wissensrepräsentation reden, dann geht es natürlich auch immer darum, dieses Wissen zu nutzen, um bestimmte Aufgaben zu lösen. Das kann die Beantwortung einer Frage sein, wie z.B. welche Bücher Hector Levesque geschrieben hat, was ein Junggeselle ist oder welche Krankheit ich habe, wenn mir die Nase läuft und ich Gliederschmerzen habe. Die erste Frage betrifft faktisches Wissen und lässt sich leicht beantworten, wenn man eine Datenbank hat, die alle Bücher zusammen mit ihren Autoren enthält. Die zweite Frage betrifft offensichtlich terminologisches Wissen und es geht hier darum, in einem geeigneten Lexikon die Definition des Begriffs nachzuschlagen. Die dritte Frage betrifft kausales Wissen über den Zusammenhang von Symptomen und Krankheiten.

In all den genannten Fällen mag die Antwort der Frage direkt bereits in der Wissensbasis repräsentiert sein und es geht nur darum, diese Antwort zu finden. Generell ist es aber nicht praktikabel, Antworten auf alle möglichen Fragen explizit zu repräsentieren. Stattdessen setzt man *Schlussfolgerungsprozeduren* ein, um auch in Fällen, in denen die Wissensbasis keine explizite Antwort enthält, die Fragen beantworten zu können.

Wollen wir beispielsweise wissen, ob Hans im Wissensrepräsentationskurs einen Freund trifft, so können wir aus der Tatsache, dass Peter ein Freund von Hans ist und Peter ebenfalls den Wissensrepräsentationskurs besucht, die Schlussfolgerung ziehen, dass man die Frage positiv beantworten kann.

Um solche Schlussfolgerungen durchzuführen, ist es hilfreich, wenn das Wissen in einer Art und Weise repräsentiert ist, die es erlaubt, durch rein syntaktische Manipulationen der für die Repräsentation verwandten Symbole, solche Schlussfolgerungen zu erhalten. Ähnlich wie in der Arithmetik, wo wir die arithmetischen Verknüpfungen auf mechanisch auszuführende Operationen reduzieren, die einfach auf dem Computer ausgeführt werden können, wollen wir Operationen auf repräsentierten Wissensinhalten ausführen können. Wie wir sehen werden, ist die formale Logik ein mächtiges Hilfsmittel, um uns an dieser Stelle weiterzuhelfen.

Die Beantwortung von Fragen ist natürlich nur eine mögliche Nutzung von repräsentiertem Wissen. Jede andere Aufgabe, wie z.B. die Navigation eines Roboters durch einen Raum, die Planung einer Vorgehensweise zur Produktion eines Artefakts, das Verstehen gesprochener Sprache oder auch die Fehlerdiagnose von technischen Geräten benutzt in irgendeiner Art repräsentiertes Wissen auf die eine oder andere Art. Oft wird dabei nur explizit repräsentiertes, faktisches Wissen genutzt. Zum Beispiel bei der Roboternavigation wird man Wissen über die gegebene Umgebung nutzen, für das keine weitere Verarbeitungsschritte erforderlich sind.

Wird aber auch implizit repräsentiertes Wissen benötigt, das durch Schlussfolgerungsprozesse generiert werden soll, so müssen diese ggf. angestoßen werden. In diesem Kontext betrachtet man ein Wissensrepräsentationssystem oft als abstrakte Datenstruktur, mit der man nur mittels Operationen wie *tell* und *ask* kommuniziert. Man teilt dem System in einer geeigneten formalen Sprache per *tell*-Operationen neue Wissensinhalte mit und fragt per *ask*-Operation an, ob Wissensinhalte in der Wissenbasis implizit oder explizit repräsentiert sind. Zusätzlich kann natürlich eine Operation wie *forget* wünschenswert sein, um falsches oder nicht mehr gültiges Wissen aus einer Wissenbasis zu entfernen. Wir werden diese Operation, die in den Bereich der sogenannten *Wissensrevision* führt, allerdings im Weiteren nicht betrachten.

4.2 Deklarative Wissensrepräsentation

Wie bereits bemerkt, kann Wissen implizit oder explizit sein. Innerhalb von KI-Systemen ist es ebenfalls so, dass Wissen entweder implizit beim Schreiben des Programmes mit eingeflossen ist oder dass es explizit unabhängig vom Programm repräsentiert wurde und auch unabhängig von dem speziellen Programm eine Bedeutung hat. In diesem Zusammenhang spricht man dann von deklarativer Wissensrepräsentation.

4.2.1 Wissensbasierte Systeme

Von einem wissensbasierten System spricht man, wenn das für die Funktion des Systems notwendige Wissen explizit und unabhängig von dem System in einer sogenannten *Wissenbasis* erfasst wurde. Damit erhält man zum einen die Möglichkeit, das Wissen auch unabhängig vom

System zu inspizieren, verifizieren und modifizieren. Zum anderen wird es möglich, das repräsentierte Wissen auch in anderen Kontexten einzusetzen.⁴

Um diese Ideen zu illustrieren, wollen wir ein kleines Beispiel betrachten. In Abb. 4.1 sehen wir ein kleines Prolog-Programm, das in der Lage ist, für einige der Mitglieder der Zähringer, eines im 11. und 12. Jahrhundert in Süddeutschland und in der Schweiz bedeutenden Fürstengeschlechts, die jeweiligen Eltern zu benennen.

```
printE(ludwig_i) :- !,
    write("karl_friedrich und caroline_luise").
printE(karl_ludwig) :- !,
    write("karl_friedrich und caroline_luise").
printE(karl) :- !,
    write("karl_ludwig und amalie").
printE(alexander) :- !,
    write("karl und stephanie").
printE(X) :-
    write("Keine Ahnung, wer die Eltern sind!").
```

Abbildung 4.1: Stammbaum der Zähringer

Während dieses kleinen Programms seine Funktion erfüllt, kann man es nur in einem sehr eingeschränkten Sinne als wissensbasiertes System bezeichnen. Das Programm in Abb. 4.2 erfüllt diese Bedingung sicherlich viel besser.

```
printE(X) :-
    mutter(X,MX), vater(X,VX), !,
    write(VX), write(" und "), write(MX).
printE(X) :-
    write("Keine Ahnung, wer die Eltern sind!").

mutter(ludwig_i,caroline_luise).
mutter(karl_ludwig,caroline_luise).
mutter(karl,amalie).
mutter(alexander,stephanie).
vater(ludwig_i,karl_friedrich).
vater(karl_ludwig,karl_friedrich).
vater(karl,karl_ludwig).
vater(alexander,karl).
```

Abbildung 4.2: Eltern von Mitgliedern der Zähringer, wissensbasiert geschlossen

Hier wird das Wissen über die Familienbeziehungen nicht in das Programm hinein codiert, sondern als gesonderte Wissensbasis repräsentiert. Das heißt, man kann diesen Teil unabhängig betrachten, verstehen und modifizieren. Man kann diese Wissensbasis auch unabhängig von dem Programm verwenden, um z.B. andere Familienzusammenhänge aufzuspüren.

⁴ Wissensbasierte Systeme in dem hier angegebenen Sinne schließen natürlich auch Expertensysteme ein. An ein Expertensystem wird man aber weitere Anforderungen stellen, wie etwa, dass es über Erklärungs-, Dialog- und Wissenserwerbskomponenten verfügt. Zu typischen KI-Methoden, die im Kontext von Expertensystemen Verwendung finden, verweisen wir auf [6].

Insgesamt sollte klar sein, dass es erhebliche Vorteile bringt, das Wissen auszufaktorisieren und separat zu behandeln. Aber natürlich muss man dafür auch einen Preis zahlen. Ein wesentlicher Preis ist, dass man dem explizit in einer Wissensbasis repräsentierten Wissen ja eine Bedeutung zuordnen möchte, die unabhängig von dem Programm ist, das auf der Wissensbasis operiert. Und hier stellt sich die Frage, welche Methode dafür am angemessensten ist. Hat man allerdings diesen Schritt getan, wird es für alle Beteiligten einfacher, über das repräsentierte Wissen zu kommunizieren, da man für die Bedeutung ja nicht mehr auf die Funktion referieren muss, die das Wissen in dem Programm spielt.

4.2.2 Die Rolle der Logik

Die formale Logik bietet Werkzeuge, um das oben genannte Problem, symbolischen Ausdrücken eine Bedeutung zuzuordnen, zu lösen. Aus diesem Grund spielt die formale Logik auch eine zentrale Rolle innerhalb der Forschung zur Wissensrepräsentation. Dabei gibt die formale Logik nicht nur Methoden vor, wie man symbolische Ausdrücke interpretieren kann, sondern parallel dazu stellt sie auch Methoden bereit, durch syntaktische Manipulationen implizite Aussagen zu erschließen, die sich zwangsläufig aus den gegebenen symbolischen Ausdrücken ergeben.

Dabei ist es eigentlich nicht richtig, von *der formalen Logik* zu sprechen. Es gibt Standardlogiken, wie die *Aussagenlogik* und die *Prädikatenlogik 1. Stufe* (siehe hierzu auch Kap. 5 *Inferenz*). Diese werden auch gerne eingesetzt, wenn es um die Formalisierung bestimmter Wissensbereiche geht. Daneben gibt es aber unzählige Varianten, wie intuitionistische Logiken, mehrwertige Logiken, Modallogiken, Relevanzlogiken, dynamische Logiken, Logiken höherer Stufe, usw. All diesen ist gemeinsam, dass sie den Begriff der *logischen Folgerbarkeit* definieren. Das heißt, es wird festgelegt, was aus einer Menge von Prämissen *notwendigerweise* folgt. Daneben gibt es aber auch Logiken, sogenannte nicht-monotone Logiken, die darauf abzielen, den Begriff der *plausiblen* Folgerbarkeit formal zu erfassen (siehe auch Kap. 6 *Nicht-monotonen Schließen*).

Man könnte jetzt meinen, dass diese Logiken direkt für die Repräsentation von Wissen eingesetzt werden. Dies ist allerdings nicht notwendigerweise der Fall. Stattdessen wird oft eine formale Sprache – ein *Wissensrepräsentationsformalismus* – für einen bestimmten Zweck entworfen, der nicht einer existierenden Logik entspricht. Um die Bedeutung der in einem Formalismus erlaubten symbolischen Ausdrücke – die *Semantik* eines Formalismus – zu spezifizieren und um Schlussfolgerungsprozeduren zu entwerfen, werden dann aber logische Methoden eingesetzt.

Eine Frage, die sich dabei ergibt, ist dann, warum man nicht so etwas wie einen *universellen Wissensrepräsentationsformalismus* entwirft, der in der Lage ist, alle Formen menschlichen Wissens zu erfassen. Der wesentliche Grund dafür, nicht in solch eine Richtung zu gehen, ist, dass mit der Ausdruckskraft einer Logik auch die erforderlichen Berechnungsressourcen wachsen. So ist es bereits für die Prädikatenlogik unentscheidbar, ob eine gegebene Aussage aus einer Menge von Prämissen logisch folgt. Allerdings kann man für jede prädikatenlogisch wahre Aussage einen Beweis finden. Für Logiken höherer Stufe gilt aber selbst dies nicht mehr, d.h. es gibt wahre Aussagen, die nicht bewiesen werden können.

Da es bei der Wissensrepräsentation auch immer um die Verarbeitung geht, beschränkt man sich in diesem Kontext deshalb zumeist auf Formalismen, deren Ausdruckskraft geringer als

die der Prädikatenlogik 1. Stufe ist, um eben logische Schlussfolgerungen effektiv berechnen zu können.

4.2.3 Schlussfolgerungstypen

Die formale Logik erlaubt uns, die logisch zwingenden Schlüsse, die sogenannten *deduktiven Schlüsse* präzise zu fassen und tatsächlich auch operational auf dem Rechner umzusetzen. Aus einer gegebenen Menge von Prämissen die logisch zwingenden Konsequenzen zu berechnen, ist aber nicht die einzige Art, Schlüsse zu ziehen.

Wie schon erwähnt, ist man oft auch an Schlüssen interessiert, die nur plausibel, aber nicht logisch zwingend sind. Man spricht dann oft auch von *anfechtbaren* Schlussweisen (engl. *defeasible reasoning*), d.h. wir schließen etwas aus einer Menge von Prämissen, sind aber bereit diesen Schluss zurückzunehmen, wenn neue, weitere Annahmen dies erfordern. Solche Schlussweisen sind offensichtlich *nicht monoton* in dem Sinne, dass hier die Menge der Konsequenzen nicht unbedingt mit der Menge der Prämissen wächst. Oft macht man dabei Annahmen, die zwar in der Regel, typischerweise stimmen, aber in Einzelfällen auch falsch sein können (engl. *default reasoning*). Wenn man beispielsweise erfährt, dass jemand einen Vogel besitzt, so wird man plausibler Weise schließen, dass dieser Vogel auch fliegen kann. Erfährt man dann zusätzlich, dass es sich um einen Strauß handelt, dann muss man diesen Schluss revidieren.

Neben diesen *annahmebasierten* Schlussweisen existieren *abduktive* Schlüsse, bei denen man von Beobachtungen auf wahrscheinliche Ursachen schließt. Dies ist immer dann der Fall, wenn man Diagnosen erstellt oder, wie Sherlock Holmes, versucht Verbrechen aufzuklären. Dabei beobachtet man Symptome, wie z.B. eine laufende Nase oder ein langes blondes Jahr auf der Jacke, und schließt dann, dass dafür ein Schnupfen bzw. die Begegnung mit einer blonden Person verantwortlich ist. Charakteristisch für diese Art von Schlüssen ist, dass man versucht, unter den vielen logisch möglichen Erklärungen die plausibelste zu finden.

Daneben gibt es dann noch *induktive* Schlüsse, bei denen man von vielen Beobachtungen auf Gesetzmäßigkeiten schließt. Man zieht zum Beispiel aus der Beobachtung, dass alle bisher erhaltenen Emails, in denen bestimmte Wörter enthalten sind, uninteressant waren, die Schlussfolgerung, dass zukünftige Emails, in denen solche Wörter enthalten sind, ebenfalls uninteressant sein werden. Diese Art des Schließens, die eng dazu verwandt ist, wie auch wir Menschen aus unseren Erfahrungen lernen, ist vom Blickwinkel der Künstlichen Intelligenz Gegenstand des Gebiets des maschinellen Lernens, und wir verweisen daher an dieser Stelle auf Kap. 12 *Maschinelles Lernen und Data Mining*.

Neben dem Erschließen neuer Aussagen hat man oft noch ganz andere Ziele im Blick. Wenn man eine Menge von als wahr angenommenen Prämissen gegeben hat, so beschränken diese die möglichen Situationen in der beschriebenen Welt. Oft ist man dann einfach an einer solchen möglichen Situation interessiert. Dies bezeichnet man dann als *Modellkonstruktion*. Solche Schlüsse spielen z.B. bei Konfigurations- und Planungsaufgaben eine große Rolle (siehe auch Kap. 10 *Planung*).

4.3 Ein Beispiel: Beschreibungslogiken

Als Beispiel eines Repräsentationsformalismus wollen wir im Folgenden eine Sprache vorstellen, in der sich terminologisches Wissen repräsentieren lässt. Dieser Formalismus gehört zur Familie der sogenannten *Beschreibungslogiken* [4]. Die grundlegende Idee dieser Logiken ist es, eine mit einer präzisen Semantik versehene Sprache zur Verfügung zu stellen, um Beziehungen zwischen Begriffen darzustellen.⁵ Mit Hilfe dieser Formalismen können wir so zum Beispiel Familienbeziehungen (wie sie etwa in Abb. 4.2 in der Beschreibung der Familie der Zähringer vorkamen) definieren und diese Beziehungen wiederum nutzen, um Verhältnisse zwischen den realen Familienmitgliedern der Zähringer (oder irgendeiner anderen Familie) zu beschreiben.

Beschreibungslogiken sind insbesondere daher von Interesse, weil sie die formale Basis für die sogenannten Ontologie-Sprachen bilden, wie sie für das *Semantic Web* (siehe Kap. 16 *Semantic Web*) genutzt werden. Beispiele für solche Ontologie-Sprachen sind die vom W3C-Konsortium spezifizierten Sprachfamilien *OWL* und *OWL 2*.

4.3.1 Der Formalismus

Unsere Repräsentationssprache enthält *Begriffssymbole*, die mit B bezeichnet werden und Objekte beschreiben, sowie *Rollensymbole*, die mit R bezeichnet werden und Beziehungen zwischen jeweils zwei Objekten beschreiben. Zudem gibt es verschiedene Operatoren, mit denen wir Rollen und Begriffe kombinieren können, um neue *Begriffsausdrücke* (im Folgenden mit C bezeichnet) zu gewinnen. Die Repräsentationssprache hat, formuliert in einer (abstrakten) BNF-Notation, folgende Gestalt:⁶

$C \rightarrow T$	universelles Konzept
B	Begriffssymbole
$C \sqcap C'$	Begriffskonjunktion
$C \sqcup C'$	Begriffsdisjunktion
$\neg C$	Begriffsnegation
$\forall R: C$	Werterestriktion
$\exists R: C$	Existentielle Restriktion

Die Begriffskonjunktion bezeichnet einen Begriff, der die Bedingungen beider Begriffe enthält. Beispielsweise bezeichnet die Begriffskonjunktion (*Mann* \sqcap *Elternteil*) den Begriff *Vater*. Ebenso verhält es sich mit der Begriffsdisjunktion und -negation. Die Werterestriktion bezeichnet Begriffe, bei denen die *Rollenfüller* der angegebenen Rolle R die Restriktionen des gegebenen Begriffs C erfüllen. So bezeichnet der Begriffsausdruck ($\forall \text{hatKind}: \text{Mann}$) den Begriff „jemand, der nur männliche Kinder hat“. Schließlich können mithilfe der Existenzrestriktion Begriffe beschrieben werden, bei denen es mindestens einen Rollenfüller gibt, der zu einem bestimmten Begriff gehört.

⁵ Beschreibungslogiken werden gelegentlich auch als *terminologische Logiken* bezeichnet und sind eng verwandt zu den Formalismen der *KL-ONE* Familie (siehe z.B. [23, 31]), die wiederum aus den sogenannten *Frame*-basierten Sprachen (siehe z.B. [25]) hervorgegangen sind.

⁶ Die folgende Notation entspricht der Standardnotation für *Beschreibungslogiken*, in der Literatur auch *Begriffssprachen*, *Termsubsumptionssprachen*, *terminologische Logiken* oder *KL-ONE-basierte Sprachen* genannt, wie sie auf dem „International Workshop on Terminological Logics“, 1991, diskutiert wurde [3] (siehe auch [5]). Die spezielle Sprache, die wir hier betrachten, trägt den Namen *ALC* (*Attributive Language with Complements*) [30].

Nehmen wir an, dass unser Vokabular die Begriffssymbole *Männlich*, *Weiblich*, *Mensch*, *Erwachsen* und das Rollensymbol *hatKind* enthält. Dann können wir z.B. folgende Begriffsausdrücke bilden:

$$\begin{aligned} \textit{Männlich} &\sqcap \textit{Mensch} \sqcap \textit{Erwachsen} \\ \textit{Mensch} &\sqcap \exists \textit{hatKind} : \top \end{aligned}$$

Die erste Zeile beschreibt den Begriff der menschlichen, männlichen Erwachsenen, oder einfach *Mann*. In der zweiten Zeile wird ein Mensch beschrieben, der mindestens ein Kind hat, also ein *Elternteil*. Um diese neu definierten Begriffe auch tatsächlich mit einem Symbol verknüpfen zu können, gibt es die Operatoren „ \sqsubseteq “ und „ \doteq “, die für eine „partielle Definition“ bzw. „vollständige Definition“ stehen und genauer besagen, dass ein Begriff *B* von einem anderen Begriff *C* subsumiert wird ($B \sqsubseteq C$) oder aber gleich zu *C* ist ($B \doteq C$). Zum Beispiel könnten wir folgende Begriffe einführen:

$$\begin{aligned} \textit{Frau} &\sqsubseteq \textit{Mensch} \sqcap \textit{Weiblich} \\ \textit{Mann} &\sqsubseteq \textit{Mensch} \sqcap \textit{Männlich} \\ \textit{Elternteil} &\doteq \textit{Mensch} \sqcap \exists \textit{hatKind} : \top \\ \textit{Vater} &\doteq \textit{Mann} \sqcap \textit{Elternteil} \\ \textit{Großelternteil} &\doteq \textit{Mensch} \sqcap \exists \textit{hatKind} : \textit{Elternteil} \\ \textit{Großmutter} &\doteq \textit{Frau} \sqcap \textit{Großelternteil} \end{aligned}$$

Diese nun eingeführten Begriffe, die die *Terminologie* unserer Anwendungsdomäne bilden, können wir jetzt aufgrund ihrer Definitionen in einer Spezialisierungshierarchie, auch *Subsumptionshierarchie* genannt, anordnen. Beispielsweise ist der Begriff *Großmutter* eine Spezialisierung von *Elternteil*.

Um über konkrete Objekte zu reden, erweitern wir unsere Repräsentationssprache um Mittel zur Beschreibung von Objekten. Wir schreiben $o \in C$, wenn das Objekt *o* zum Begriff *C* gehört. Also z.B. $s \in \textit{Mensch}$, falls *s* ein Mensch ist. $\langle x, y \rangle \in R$ schreiben wir, wenn das Objekt *y* ein Rollenfüller für die Rolle *R* des Objektes *x* ist. $\langle s, st \rangle \in \textit{hatKind}$ bedeutet zum Beispiel, dass das Objekt *st* das Kind von *s* ist. Dies erlaubt uns also, nicht nur terminologisches Wissen, sondern auch einfaches Faktenwissen zu repräsentieren. Im Kontext von Beschreibungslogien bezeichnet man den terminologischen Teil einer Wissensbasis meist als *TBox* und den Teil, in dem Faktenwissen (engl. *assertions*) spezifiziert wird, als *ABox*.

Betrachten wir an dieser Stelle nochmals das Beispiel der Zähringer-Familie aus Abschnitt 2.1, so können wir das im dort angegebenen Prolog-Programm (Abb. 2) repräsentierte Wissen in unserem Formalismus nun wie folgt wiedergeben:

$$\begin{aligned} \textit{caroline_luise} &\in \textit{Frau} \\ \textit{amalie} &\in \textit{Frau} \\ \langle \textit{caroline_luise}, \textit{ludwig_i} \rangle &\in \textit{hatKind} \\ \langle \textit{caroline_luise}, \textit{karl_ludwig} \rangle &\in \textit{hatKind} \\ \langle \textit{amalie}, \textit{karl} \rangle &\in \textit{hatKind} \\ &\dots \end{aligned}$$

Natürlich könnte man an dieser Stelle die Frage stellen, was wir durch die Einführung unseres Formalismus gegenüber einer Wissensrepräsentation mittels eines Prolog-Programms gewonnen haben. Im Grunde könnten wir mittels Prolog-Programmen bei weitem mehr Wissen repräsentieren: wir könnten in Prolog ja zum Beispiel Begriffe definieren wie „Mutter von mindestens drei Kindern“, die wir in unserem Formalismus nicht definieren können. Die wesentliche Idee hierbei ist nun, dass die Beschränkung der Ausdrucksstärke eines Formalismus es ermöglicht, Anfragen an eine Wissenbasis effizienter zu beantworten, als dies für eine in einem Prolog-Programm spezifizierte Wissenbasis möglich wäre. Auf diesen entscheidenden Gesichtspunkt kommen wir nochmals in Abschnitt 4.3.5 zu sprechen.

4.3.2 Semantik

Ein wesentlicher Schwachpunkt von frühen Repräsentationsformalismen ist es, dass die Bedeutung dieser Formalismen nur intuitiv und/oder durch das Systemverhalten gegeben ist. Beispielsweise kritisierte Woods [34], dass es keine Semantik der *semantischen Netze* gäbe. Dies führte dazu, dass relativ viele (meist fruchtlose) Debatten über die Ausdrucksfähigkeit von Formalismen geführt wurden. Schlimmer noch, es war auch nicht klar, *was* denn nun tatsächlich mithilfe eines Repräsentationsformalismus repräsentiert werden kann.

Eine Lösung dieses Problems ist es, eine *formale Semantik* anzugeben. Dabei gibt es natürlich erst einmal beliebige Möglichkeiten. Als Standardwerkzeug dafür haben sich jedoch logische Methoden bewährt. Wir haben damit also die Situation, dass zwar Logik selbst nicht als Repräsentationsformalismus benutzt wird, dass die verschiedenen Repräsentationsformalismen aber jeweils eine logische Semantik besitzen. Damit ist es möglich,

- die Bedeutung von Repräsentationsformalismen zu kommunizieren,
- die intendierte Semantik mit der formalen Spezifikation zu vergleichen [17],
- die Ausdrucksfähigkeit von Repräsentationsformalismen zu bestimmen [1, 26],
- die *Berechenbarkeitseigenschaften* von Repräsentationsformalismen zu bestimmen [8, 22, 24],
- *Inferenzalgorithmen* zu entwickeln und deren *Korrektheit* und *Vollständigkeit* zu beweisen [19, 30].

In unserem Fall könnten wir die Semantik unseres kleinen Repräsentationsformalismus durch eine direkte Übersetzung in die Prädikatenlogik erster Stufe angeben. Jedes Begriffssymbol B entspräche einem einstelligen Prädikat $B(x)$. Jedes Rollensymbol R entspräche einem zweistelligen Prädikat $R(x, y)$ und ein komplexer Begriffsausdruck C entspricht einer Formel $C(x)$ mit einer freien Variablen x .

Alternativ wollen wir die Semantik direkt mit Hilfe mengentheoretischer Ausdrücke angeben. Wesentlicher Teil einer solchen mengentheoretischen Semantik ist eine *Interpretation I* bestehend aus einer Interpretationsfunktion I^1 und einer nicht-leeren Grundmenge U . Die Interpretationsfunktion weist jedem Begriffssymbol B eine Teilmenge $B^I \subseteq U$, jedem Rollensymbol R eine zweistellige Relation $R^I \subseteq U \times U$ und jedem Symbol für Individuen a ein Element $a^I \in U$ zu. Komplexen Begriffsausdrücken können wir nun induktiv ebenfalls eine *Extension*

wie folgt zuordnen:

$$\begin{aligned}
 \top^I &:= U \\
 (C \sqcap C')^I &:= C^I \cap C'^I \\
 (C \sqcup C')^I &:= C^I \cup C'^I \\
 (\neg C)^I &:= U \setminus C^I \\
 (\forall R : C)^I &:= \{x \in U : \text{für jedes } y \in U \text{ mit } (x, y) \in R^I \text{ gilt: } y \in C^I\} \\
 (\exists R : C)^I &:= \{x \in U : \text{es gibt ein } y \in U \text{ mit } (x, y) \in R^I \text{ und } y \in C^I\}
 \end{aligned}$$

Eine Interpretation I erfüllt nun eine Formel der Gestalt $B \sqsubseteq C$, falls $B^I \subseteq C^I$, eine Formel der Gestalt $B \doteq C$, falls $B^I = C^I$, eine Formel der Gestalt $a \in C$, falls $a^I \in C^I$, und eine Formel der Gestalt $\langle a, b \rangle \in R$, falls $(a^I, b^I) \in R^M$. Man sagt, dass I ein Modell einer Menge Σ solcher Formeln ist, falls I jede einzelne der Formeln aus Σ erfüllt. Eine Menge von Formeln heißt erfüllbar, falls sie ein Modell hat. Schließlich definieren wir auch den Folgerungsbegriff: Eine Formel ϕ folgt aus einer Menge Σ von Formeln (symbol. $\Sigma \models \phi$), falls jedes Modell von Σ auch ϕ erfüllt.

Alternativ zu einer Semantik, die alle Terme der Sprache in mengentheoretische Ausdrücke übersetzt, sind auch andere Möglichkeiten denkbar. Beispielsweise könnten wir die Repräsentationssprache in (ein Fragment) der Prädikatenlogik übersetzen, wie oben skizziert. Daneben eignen sich aber auch andere Logiken. Insbesondere *Modallogiken* [12] sind dabei gute Kandidaten. Tatsächlich kann man viele terminologische Sprachen als notationelle Varianten von bestimmten Multimodallogiken auffassen [29].

4.3.3 Inferenzdienste

Mit der im letzten Abschnitt definierten Semantik können wir nun eine Reihe von Problemstellungen genauer spezifizieren, die sich ergeben, wenn eine Wissensbasis in unserem Beispieldomänum vorliegt. Hat man algorithmische Lösungen für diese Problemstellungen gefunden, so kann ein Tool, das diese Algorithmen implementiert, Inferenzdienste anbieten, die es dem Anwender erlauben, Anfragen an eine Wissensbasis zu beantworten.

Ein Anwender, der eine Wissensbasis in unserem Beschreibungsformalismus spezifiziert, könnte zum Beispiel daran interessiert sein zu prüfen, ob die von ihm entworfene Terminologie insofern sinnvoll ist, dass alle definierten Begriffe konsistent, d.h. erfüllbar, sind. Gerade bei großen Terminologien kann es passieren, dass man solche nicht erfüllbaren Konzepte spezifiziert. Schreiben wir T für den terminologischen Teil der Wissensbasis und \perp als eine Abkürzung für den Begriffsausdruck $\neg \top$, so kann man dieses Problem in symbolischer Form durch die Frage „ $T \models B \sqsubseteq \perp$ “ beschreiben. Denn offensichtlich gilt $T \not\models B \sqsubseteq \perp$ genau dann, wenn es ein Modell M von T (also eine Interpretation aller Begriffssymbole) gibt, derart dass B^M keine Teilmenge der leeren Menge, also B^M selbst nicht leer ist.

Weitere Inferenzdienste, die die Terminologie betreffen, ergeben sich daraus, dass man zum Beispiel überprüfen möchte, ob ein Begriff C (genauer: ein durch C ausgedrückter Begriff) von einem anderen Begriff C' subsumiert wird ($T \models C \sqsubseteq C'$) oder ob C und C' gar den gleichen Begriff ausdrücken ($T \models C \doteq C'$).

Man kann des Weiteren auch die Anfrage stellen, ob die gesamte Wissenbasis, also Terminologie T und das in der Wissenbasis spezifizierte Faktenwissen A konsistent ist, ob es also ein Modell gibt, das alle Formeln aus der Menge $T \cup A$ zugleich erfüllt (man sagt dann auch, dass A mit T konsistent ist). Diese Fragestellung ist natürlich nur dann interessant, wenn überhaupt Faktenwissen vorhanden ist, denn in unserem Formalismus hat eine TBox stets ein Modell (wenn vielleicht auch nur eines, in dem einzelne Begriffsextensionen leer sind). In Bezug auf das früher angegebene Beispiel der Zähringer-Familie lässt sich natürlich einfach feststellen, dass die faktische Beschreibung der Familienverhältnisse der Zähringer mit den allgemeinen Definitionen von Familienkonzepten konsistent ist.

Einer der wichtigsten Inferenzdienste ist es, zu prüfen, ob ein in der Wissenbasis spezifiziertes Objekt a Instanz eines Begriffs C ist. An dieser Stelle wollen wir darauf hinweisen, dass für die meisten Wissenbasen eine Antwort auf diese Frage nicht eindeutig ist, d.h., Wissenbasen können verschiedene Modelle besitzen, solche, in denen $a \in C$ gilt, und solche, in denen $a \in C$ nicht gilt. Meist stellt man die Frage daher in der Form, ob a Instanz des Konzepts C sein muss, d.h., dass aus Terminologie plus Faktenwissen logisch folgt, dass a Instanz von C ist ($(T \cup A \models C(a))$). Entsprechendes muss man berücksichtigen, wenn man zu einem gegebenen Begriff C alle in der Wissenbasis spezifizierten Objekte ausgeben will, die Instanz von C sein müssen.

4.3.4 Inferenzalgorithmen

Eine Repräsentationssprache gibt uns die Ausdrucksmittel, um das Wissen einer Anwendungsdomäne zu beschreiben. Die mit der Sprache assoziierte Semantik verrät uns, welche formale Bedeutung ein gegebener Repräsentationsausdruck hat und wie die Dienste eines Repräsentationssystems zu interpretieren sind. Was uns jetzt noch fehlt, ist die Mechanisierung, die *Algorithmisierung*, der Dienste eines Repräsentationssystems. Wie muss ein Algorithmus aussehen, der bei gegebener Wissenbasis entscheidet, ob ein bestimmter Begriff konsistent ist? Wie muss ein Algorithmus aussehen, der für einen Begriff alle in der Wissenbasis spezifizierten Instanzen des Begriffs ausgibt? Wie sieht ein Algorithmus aus, der eine Spezialisierungshierarchie einer Menge von Begriffen berechnet? Und angenommen, wir haben auf alle diese Fragen Antworten gefunden, wie können wir zeigen, dass die Algorithmen tatsächlich die in sie gesetzten Erwartungen erfüllen?

Wir wollen mit der letzten Frage beginnen. Angenommen, wir haben einen *Entscheidungsalgorithmus* (ein Algorithmus, der „ja“ oder „nein“ ausgibt) spezifiziert, von dem wir glauben, dass er entscheidet, ob ein beliebiges Objekt a zum Begriff C gehört. Dann müssen wir beweisen, dass

- immer, wenn der Algorithmus eine positive Antwort gibt, tatsächlich $a \in C$ gilt,
- immer, wenn $a \in C$ gilt, der Algorithmus eine positive Antwort gibt und
- der Algorithmus immer (bei jeder erlaubten Eingabe) terminiert.

Die erste Bedingung bezeichnet man als *Korrektheit*, die zweite als *Vollständigkeit* des Algorithmus – wie bei logischen Kalkülen (vgl. Abschnitt 4.3.2). Die dritte Bedingung ist eine notwendige Bedingung dafür, dass es sich überhaupt um einen Algorithmus handelt. Wird die dritte Bedingung nicht erfüllt, sprechen wir von einer *Prozedur*. Allerdings kann es vorkommen, dass

nicht alle drei Bedingungen zusammen erfüllt werden können. Schlussfolgerung in der Prädikatenlogik erster Stufe ist beispielsweise unentscheidbar, d.h., dass man keinen korrekten und vollständigen *Algorithmus* angeben kann, der entscheidet, ob eine Aussage aus einer anderen folgt. Man kann allerdings vollständige und korrekte *Prozeduren* für dieses Problem angeben, z.B. *Resolution* (siehe Kap. 5 *Inferenz*).

Offensichtlich können wir diese Eigenschaften nur dann untersuchen, wenn wir die Semantik des Repräsentationsformalismus formal angegeben haben. Außerdem ist es offensichtlich, dass wir einen Beweis nur dann führen können, wenn der Algorithmus kompakt genug ist, so dass ein Beweis der Korrektheit und Vollständigkeit auch tatsächlich möglich ist. Aus diesem Grund wählt man eine möglichst abstrakte Form bei der Angabe des Algorithmus.

Wir wollen beispielhaft für den oben angegebenen Repräsentationsformalismus einen abstrakten Algorithmus angeben, der *Subsumption zwischen zwei Begriffen entscheidet*, d.h. der bestimmt, ob innerhalb einer gegebenen Terminologie T ein Begriff C spezieller als ein anderer Begriff C' ist, symbolisch $T \models C \sqsubseteq C'$. Obwohl dies ja nur einer der Dienste ist, für die wir uns interessieren, ist die Aufgabenstellung doch so beschaffen, dass dieser Algorithmus verallgemeinert werden kann, um auch die anderen Dienste zu realisieren.

Als ersten Schritt zur Berechnung der Subsumptionsrelation in einer Terminologie T geben wir jetzt ein einfaches Verfahren an, um bei der Subsumptionsbestimmung von der Terminologie zu abstrahieren. Dazu formen wir die Terminologie T in eine andere Terminologie T^* um, in der der \sqsubseteq -Operator nicht mehr vorkommt. Zu diesem Zweck führen wir für jedes partiell definierte Begriffssymbol ein neues Begriffssymbol ein, das auf der rechten Seite der Definition konjunktiv hinzugenommen wird, also z.B.:

$$\begin{aligned} \text{Frau} &\doteq \text{Mensch} \sqcap \text{Weiblich} \sqcap \text{Frau}^* \\ \text{Mann} &\doteq \text{Mensch} \sqcap \text{Männlich} \sqcap \text{Mann}^* \end{aligned}$$

Die neuen Symbole Mann^* , Frau^* , ... bleiben innerhalb der Terminologie undefiniert. Wie man sich leicht klar macht, ändert man durch diesen Übergang von T nach T^* nichts an der Bedeutung der ursprünglichen Begriffssymbole. Insbesondere bleibt die Subsumptionsbeziehung zwischen Begriffsausdrücken, die nur Symbole der ursprünglichen Terminologie verwenden, von dieser Transformation unberührt.

Wenn wir jetzt einen Begriffsausdruck C gegeben haben, so wollen wir mit $E(C)$ seine *Expansion* bezüglich der Terminologie T^* bezeichnen, wobei mit *Expansion* die Ersetzung von Begriffssymbolen durch ihre Definition gemeint ist, die so lange durchgeführt wird, bis nur noch undefinierte Begriffssymbole auftauchen. Beispielsweise ist $E(\text{Großmutter})$:

$$\text{Mensch} \sqcap \text{Weiblich} \sqcap \text{Frau}^* \sqcap \text{Mensch} \sqcap \exists \text{hatKind} : (\text{Mensch} \sqcap \exists \text{hatKind} : \top).$$

Die Funktion $E(\cdot)$ ist natürlich nur dann wohl-definiert, wenn

1. jedes Begriffssymbol höchstens einmal auf der linken Seite einer Definition auftaucht und
2. die Terminologie keine *Zyklen* enthält (d.h., man kann die Formeln in der Terminologie so anordnen, dass auf der rechten Seite einer Definition nur Begriffssymbole vorkommen, die undefiniert sind oder aber in der Anordnung vorher definiert wurden).

Wollen wir jetzt berechnen, ob $T \models C \sqsubseteq C'$ gilt, so können wir stattdessen auch berechnen, ob $E(C)$ von $E(C')$ in der leeren Terminologie subsumiert wird, symbolisch $\emptyset \models E(C) \sqsubseteq E(C')$. Damit können wir uns jetzt darauf konzentrieren, einen Algorithmus für die Bestimmung der Relation \sqsubseteq in der leeren Terminologie zu entwerfen.

Eine erste Idee für solch einen Algorithmus könnte sein, die beiden Ausdrücke $E(C)$ und $E(C')$ strukturell miteinander zu vergleichen. Dies ist auch tatsächlich eine Möglichkeit, wie Levesque und Brachman [22] gezeigt haben. Sie haben für eine Teilmenge des oben angegebenen Formalismus einen vollständigen und korrekten *Subsumptionsalgorithmus* angeben, der auf strukturellem Vergleich basiert. Allerdings stellt sich heraus, dass die Verallgemeinerung dieses Vorgehens für mächtigere Formalismen, wie z.B. für den hier betrachteten Formalismus, aus mehreren Gründen nicht gangbar ist.

Wir wollen hier einen anderen Weg gehen, der an die *Constraint-Lösungsmethode* von Schmidt-Schauß und Smolka [30] angelehnt ist. Dazu machen wir uns zuerst klar, dass man Subsumption zwischen zwei Begriffsausdrücken auf *Inkonsistenz* eines Begriffsausdruck reduzieren kann, wobei ein Begriffsausdruck *inkonsistent* oder auch *leer* genannt wird, wenn man damit keine Objekte beschreiben kann, oder formal, wenn $C^I = \emptyset$ für jede Interpretation I . Nun gilt folgende Äquivalenz:

$$C \sqsubseteq C' \quad \text{genau dann wenn} \quad C \sqcap \neg C' \text{ inkonsistent.}$$

Wenn wir also einen Algorithmus für Subsumption gefunden haben, können wir auch Inkonsistenz von Begriffsausdrücken entscheiden und umgekehrt.⁷ Da Algorithmen für Inkonsistenz einfacher zu entwickeln und verifizieren sind, werden wir uns darauf konzentrieren, genau so einen Algorithmus zu entwickeln. Die Idee dahinter ist relativ simpel. Man versucht auf allen möglichen Wegen ein Modell zu konstruieren, in dem der testende Begriffsausdruck eine Instanz hat. Gelingt dies, ist der Begriffsausdruck konsistent. Gelingt dies nicht – und haben wir tatsächlich alle Möglichkeiten für die Konstruktion eines solchen Modells ausgeschöpft – dann ist der Begriffsausdruck inkonsistent.

Um den Algorithmus möglichst einfach zu halten, werden wir den Begriffsausdruck vorher in eine Normalform, die sogenannte *Negationsnormalform*, überführen, in der der Negationsoperator nur direkt vor Begriffssymbolen steht. Ein beliebiger Begriffsausdruck kann in seine Negationsnormalform überführt werden, indem man folgende Äquivalenzen ausnutzt:

$$\begin{aligned}\neg(C \sqcap C') &\equiv \neg C \sqcup \neg C' \\ \neg(C \sqcup C') &\equiv \neg C \sqcap \neg C' \\ \neg(\neg C) &\equiv C \\ \neg(\forall R: C) &\equiv \exists R: \neg C \\ \neg(\exists R: C) &\equiv \forall R: \neg C.\end{aligned}$$

Um zu bestimmen, ob ein Ausdruck in Negationsnormalform inkonsistent ist, führen wir jetzt den Begriff eines *Constraints* ein. Ein Constraint ist entweder ein Ausdruck der Art $(x: C)$ oder ein Ausdruck der Form $(x R y)$ mit der intuitiven Bedeutung, dass das Objekt x zum Begriff C

⁷ Voraussetzung dafür ist, dass wir Begriffskonjunktion und -negation in unserer Sprache zur Verfügung haben.

gehört bzw. dass die beiden Objekte x und y in der R -Beziehung zueinander stehen. Eine Menge von Constraints wird *Constraint-System* genannt und im Folgenden mit \mathcal{C} bezeichnet. Gegeben ein Begriffsausdruck C , dann nennen wir das System $\{(x: C)\}$ ein *jungfräuliches Constraint-System* und die Variable x wird *Wurzelvariable* genannt. Ein Constraint-System heißt *erfüllbar*, wenn der existentielle Abschluss über die Konjunktion aller Constraints eine erfüllbare Formel ist, und *unerfüllbar* sonst. Wir geben jetzt eine Menge von Transformationsregeln an, die die Erfüllbarkeitseigenschaft invariant lassen und entweder ein Objekt konstruieren, das alle Constraints erfüllt, oder zeigen, dass dies nicht möglich ist [18].

1. $\mathcal{C} \cup \{x: (C \sqcap C')\} \rightarrow_{\sqcap} \mathcal{C} \cup \{x: (C \sqcap C'), x: C, x: C'\},$ falls nicht schon $\{x: C, x: C'\} \subseteq \mathcal{C}.$
2. $\mathcal{C} \cup \{x: (\exists R: C)\} \rightarrow_{\exists} \mathcal{C} \cup \{x: (\exists R: C), x R y, y: C\},$ falls es nicht schon eine Variable z gibt, so dass $\{x R z, z: C\} \subseteq \mathcal{C}$ (hierbei sei y eine neue Variable, die nicht in \mathcal{C} vorkommt).
3. $\mathcal{C} \cup \{x: (\forall R: C), x R y\} \rightarrow_{\forall} \mathcal{C} \cup \{x: (\forall R: C), x R y, y: C\},$ falls nicht schon $(y: C) \in \mathcal{C}.$

Es sollte klar sein, dass ein Constraint-System \mathcal{C} nach der Anwendung einer der obigen Regeln erfüllbar ist, genau dann, wenn das System vorher erfüllbar war.

Was uns jetzt noch fehlt, ist eine Regel für die Begriffsdisjunktion. Im Gegensatz zu den anderen drei Regeln muss diese jedoch *nicht-deterministisch* sein. Um zu prüfen, ob $x: (C \sqcup C')$ erfüllbar ist, müssen wir entweder zeigen, dass $x: C$ oder $x: C'$ erfüllbar ist. Die entsprechende Regel lautet daher:

4. $\mathcal{C} \cup \{x: (C \sqcup C')\} \rightarrow_{\sqcup} \begin{cases} \mathcal{C} \cup \{x: (C \sqcup C'), x: C\} \text{ oder} \\ \mathcal{C} \cup \{x: (C \sqcup C'), x: C'\}, \end{cases}$
falls nicht schon $(x: C) \in \mathcal{C}$ oder $(x: C') \in \mathcal{C}.$

Falls ein Constraint-System erfüllbar ist, dann muss wenigstens eine der beiden Möglichkeiten der Regel \rightarrow_{\sqcup} zu einem erfüllbaren Constraintssystem führen. Umgekehrt, falls ein Constraint-System unerfüllbar ist, wird das System nach der Anwendung von \rightarrow_{\sqcup} unerfüllbar sein, egal welche Möglichkeit wir gewählt haben.

Wenden wir diese vier Regeln auf ein jungfräuliches Constraint-System solange an, bis keine Regel mehr anwendbar ist, erhalten wir ein *vollständiges Constraint-System*, und ob ein vollständiges Constraint-System erfüllbar ist, kann man sehr einfach feststellen. Ein Paar von Constraints der Art $(x: B), (x: \neg B)$, wobei B ein Begriffssymbol ist, wollen wir *elementaren Widerspruch* nennen. Offensichtlich ist ein Constraint-System nicht erfüllbar, falls es einen elementaren Widerspruch enthält. Gibt es keinen elementaren Widerspruch, dann können wir das vollständige Constraint-System benutzen, um ein Modell aufzubauen, in dem die Wurzelvariable ein Objekt bezeichnet, das zu dem ursprünglichen Begriffsausdruck gehört.

Mit den obigen Aussagen folgt daraus, dass der Begriffsausdruck C inkonsistent ist, genau dann, wenn jede mögliche Vervollständigung des jungfräulichen Constraint-Systems $\{x: C\}$ einen elementaren Widerspruch enthält.

Obwohl dieses Verfahren erst einmal nicht wie ein Algorithmus aussieht, kann man sich leicht klar machen, dass es tatsächlich immer terminiert und damit ein Entscheidungsverfahren ist. Die wesentliche Idee bei einem Terminierungsbeweis für das oben angegebene Verfahren ist,

dass die Transformationsregeln komplexe Ausdrücke in einfachere Ausdrücke zerlegen. Die nicht-deterministische Komponente kann man leicht beseitigen, indem man jeweils beide Möglichkeiten der Disjunktionsregel durchspielt. Natürlich wird man bei einer konkreten Implementierung nicht unbedingt ein Constraint-System und Regelanwendungen benutzen, um den Inkonsistenztest zu realisieren, da dieses nicht sehr effizient ist. Statt dessen wird man vermutlich andere Möglichkeiten, wie z.B. rekursiven Abstieg über die Struktur des Begriffsausdrucks, benutzen. Für die Verifikation des Verfahrens ist die obige abstrakte Form eines Algorithmus jedoch ideal.

Am Schluss dieses Abschnitts wollen wir noch einmal die Entwicklung unseres Inferenzalgorithmus zusammenzufassen. Wir waren ausgegangen von der Frage, wie wir $T \models C \sqsubseteq C'$ bestimmen können. Dieses Problem haben wir reduziert auf die Berechnung von $\emptyset \models E(C) \sqsubseteq E(C')$ – auf den Subsumptionstest von Begriffsausdrücken in der leeren Terminologie. Subsumption in der leeren Terminologie haben wir auf den Test der Inkonsistenz der Negationsnormalform von $E(C) \sqcap \neg E(C')$ reduziert, was wiederum auf die Unerfüllbarkeit von Constraint-Systemen reduziert wurde. Das letzte Problem wurde schließlich gelöst, indem wir eine Menge von vier Transformationsregeln angegeben haben, die angewendet auf ein jungfräuliches Constraint-System entweder immer zu einem elementaren Widerspruch führen und damit die Inkonsistenz beweisen, oder aber zu einem vollständigen Constraint-System ohne elementaren Widerspruch, das es uns erlaubt, ein Modell zu konstruieren, das den ursprünglichen Konzeptausdruck erfüllt.

4.3.5 Berechenbarkeitseigenschaften

Man kann sich an dieser Stelle fragen, was denn nun besonderes an unserem Repräsentationsformalismus ist. Offensichtlich ist es doch möglich, ihn vollständig in die Prädikatenlogik einzubetten. Da wir wissen, wie vollständige und korrekte Deduktionsverfahren für Prädikatenlogik aussehen, könnte man jetzt hergehen und solche Verfahren anwenden, statt sich den Kopf über spezielle Inferenzalgorithmen zu zerbrechen. Tatsächlich geht ein solches Argument jedoch am Kern der Sache vorbei. Um ein Beispiel aus der Bereich der Theoretischen Informatik zu geben, würde man ja auch die Untersuchung der Eigenschaften von kontextfreien Sprachen und die Entwicklung von Parsingalgorithmen für spezielle kontextfreie Sprachen nicht mit dem Argument abtun, dass es sich hierbei nur um Spezialfälle von kontextsensitiven Sprachen handelt.

Was uns an dieser Stelle interessiert, ist also die spezielle Struktur von Repräsentationsformalismen, die unter Umständen *effizientere* Verfahren zuläßt als allgemeine Beweisverfahren für die Prädikatenlogik erster Stufe. Unter Umständen sind diese Verfahren so beschaffen, dass wir gewisse *Garantien* für die Inferenzverfahren abgeben können, z.B., dass wir für jede Eingabe nach endlicher Zeit eine Antwort bekommen oder sogar schon nach einer Zeit, die durch eine Funktion in der Länge der Eingabe begrenzt ist.

Als Beispiel können wir hier unseren Repräsentationsformalismus betrachten, der offensichtlich die Eigenschaft hat, dass Subsumption *entscheidbar* ist, da wir ja ein Entscheidungsverfahren angeben können. Das heißt, wir haben gegenüber der Prädikatenlogik tatsächlich etwas gewonnen – auf dem Gebiet der Berechenbarkeitseigenschaften. Allerdings haben wir natürlich auch etwas von der Ausdrucksfähigkeit der Prädikatenlogik eingebüßt. Solche Betrachtungen gehören zu einer der wichtigsten Aufgaben der Wissensrepräsentation, nämlich den richtigen Kompromiss zwischen *Ausdrucksfähigkeit* und *Berechenbarkeitseigenschaften* zu finden [22].

Hierbei ist zu beachten, dass die spezielle syntaktische Struktur der Repräsentationssprache nur eine relativ untergeordnete Rolle spielt. Würden wir statt der linearen Form unseres Repräsentationsformalismus eine graphische Form wählen, die an semantischen Netzen orientiert ist, so mag man einen Vorteil bei der *Präsentation* des Wissens haben und einen heuristischen Vorteil beim Nachdenken über repräsentiertes Wissen und Inferenzverfahren zur Verarbeitung des Wissens haben. Solange aber die *Ausdrucksfähigkeit* von Formalismen gleich ist, das heißt salopp gesprochen, dass alles, was wir in dem einen Formalismus ausdrücken können, mit ungefähr dem gleichen Aufwand auch in dem anderen Formalismus ausdrückbar ist – ein Punkt, den man mithilfe der formalen Semantik überprüfen kann [1] – solange werden auch Inferenzverfahren ungefähr den gleichen Aufwand benötigen.

Nun ist die Unterscheidung in *entscheidbare* und *unentscheidbare Probleme* aber sehr grob und meist nicht sehr aussagekräftig. Viele *im Prinzip entscheidbare Probleme* können nichtsdestoweniger sehr kompliziert sein – so kompliziert, dass ein Algorithmus mehr Laufzeit benötigt als die zu erwartende Lebensdauer der Erde beträgt (auch wenn wir von sehr viel schnelleren Computern ausgehen). Wenn wir z.B. Brettspiele wie Schach oder Go betrachten, so ist das Problem, den besten Zug zu finden, im Prinzip entscheidbar, da es nur endlich viele Brettkonfigurationen gibt und somit auch nur endlich viele Spielverläufe. Die Anzahl der möglichen Brettkonfigurationen ist aber so groß, dass diese prinzipielle Entscheidbarkeit keine Relevanz für die Praxis besitzt.

Eine feinere Unterteilung zur Beurteilung der Berechenbarkeitseigenschaften bietet die *Komplexitätstheorie* [13, 27]. Man unterteilt hierbei die Menge aller Probleme in solche, für die Algorithmen existieren, deren Laufzeit in allen Fällen durch eine polynomiale Funktion in der Länge der Eingabe nach oben abgeschätzt werden können und solche, bei denen keine Algorithmen mit dieser Eigenschaft existieren. Die ersten werden als *handhabbare Probleme* bezeichnet. Der Grund für diese Art der Unterteilung liegt darin, dass man bei handhabbaren Problemen in der Regel davon ausgehen kann, dass auch relativ große Probleme noch in vernünftiger Zeit gelöst werden können, insbesondere, wenn der Grad des Polynoms niedrig ist (≤ 3). Bei nicht-handhabbaren Problemen steigt die notwendige Rechenzeit jedoch meist so schnell, dass nur relativ kleine Eingaben in vernünftiger Zeit bearbeitet werden können.

Natürlich wäre es wünschenswert, dass die Inferenzprobleme von Repräsentationsformalismen handhabbar sind, dass man also Inferenzalgorithmen angeben kann, die ein polynomiales Laufzeitverhalten haben. Levesque und Brachman [22] gehen sogar so weit, dass sie fordern, jeder Repräsentationsformalismus müsse diese Eigenschaft haben. Dies schränkt jedoch die Ausdrucksfähigkeit und/oder die Art der möglichen Dienste sehr radikal ein, so dass diese Forderung meist nicht eingehalten werden kann [9].

Wir wollen an dieser Stelle wieder unsere Repräsentationssprache und die Berechnung der Subsumptionsrelation als Beispiel heranziehen. Wenn wir den Constraint-Lösungsalgorithmus, der im letzten Abschnitt entwickelt wurde, analysieren, so stellen wir fest, dass er einige Stellen enthält, die dazu führen, dass im schlechtesten Fall die Laufzeit exponentiell mit der Größe der Eingabe wächst. Natürlich stellt sich sofort die Frage, ob man es nicht besser machen könnte – ob das Subsumptionsproblem handhabbar ist und wir lediglich einen schlechten Algorithmus entworfen haben.

Wenn man jetzt den Inferenzalgorithmus noch einmal betrachtet, stellt man fest, dass er selbst auf einer nicht-deterministischen Maschine⁸ unter Umständen mehr als polynomial viel Zeit verbraucht. Das Zusammenspiel der \rightarrow_{\exists} - und der \rightarrow_{\forall} -Regel kann dazu führen, dass exponentiell viele Variablen eingeführt werden. Man kann in der Tat zeigen, dass das Subsumptionsproblem in unserem Formalismus zu den schwierigsten Problemen einer Klasse von Problemen gehört, die dadurch beschrieben werden kann, dass der Speicherplatzbedarf höchstens polynomial mit der Eingabegröße wächst (die Laufzeit aber nicht beschränkt ist) [30]. Diese Klasse heißt PSPACE und umfasst die bekanntere Klasse NP, in der alle Probleme liegen, die in polynomialer Laufzeit auf nicht-deterministischen Maschinen entschieden werden können. Von dieser Klasse nimmt man an, dass ihre schwierigsten Probleme nicht in polynomialer Zeit entscheidbar sind.

Diese sehr präzise Charakterisierung der Komplexität des Subsumptionsproblems macht zweierlei deutlich. Erstens wissen wir jetzt, dass wir tatsächlich keinen im Prinzip besseren Algorithmus als den oben angegebenen finden können.⁹ Zweitens ist jetzt klar, dass wir nicht damit rechnen können, dass wir ein Repräsentationssystem bauen können, das unter allen Umständen eine korrekte Antwort in vernünftiger Zeit liefert.

4.3.6 Neuere Entwicklungen

Neben diesem im Detail beleuchteten Formalismus kann man nun eine Vielzahl von Varianten betrachten. So gibt es einerseits deutlich ausdrucksstärkere Formalismen, zum Beispiel solche, die noch weitere Konstruktoren für Begriffsausdrücke erlauben (z.B. Konstruktoren, die die Anzahl von Rollenfüllern abschätzen). Ferner kann man Beschreibungslogiken betrachten, in denen es möglich ist, die formalen Eigenschaften von Rollen genauer festzulegen, z.B. dass eine Rolle transitiv ist. Diese Konstruktoren und noch ein paar mehr sind beispielsweise in der Beschreibungslogik *SHOIN* enthalten, die die formale Grundlage der Web-Ontologiesprache OWL-DL darstellt [21]. Die Weiterentwicklung dieser Sprache hat zu OWL 2 geführt [16], die auf der noch mächtigeren Beschreibungslogik *SHROIQ* basiert [20]. *SHROIQ* lässt insbesondere Rolleninklusionsaxiome in begrenzter Weise zu, die im Allgemeinen zu Unentscheidbarkeit führen können.

Während die bisher erwähnten Erweiterungen immer die Eigenschaft hatten, dass man ein Fragment der Prädikatenlogik 1. Stufe erhielt, gibt es auch Konstruktoren (wie etwa der transitive Abschluss-Konstruktor), die bewirken, dass sich der resultierende Formalismus nicht mehr in die Prädikatenlogik 1. Stufe einbetten lässt.

Neben diesen ausdrucksstarken Formalismen sind auch Formalismen von praktischem Interesse, die deutlich ausdrucksschwächer als die oben erwähnten *ALC*-Varianten sind. Es zeigt sich zum Beispiel, dass (oftmals sehr große) medizinische Wissensbasen wie etwa SNOMED (*Systematized Nomenclature of Medicine*) mit sehr eingeschränkten Begriffskonstruktoren auskommen [32]. Insbesondere ist die *Werterestriktion* $\forall R : C$ nicht erforderlich. Damit kann man

⁸ Eine nicht-deterministische Maschine kann man sich vorstellen als einen normalen Computer, der an bestimmten Punkten nicht-deterministisch eine Entscheidung fällt. Man definiert dann, dass ein Problem von einer solchen Maschine gelöst wird, falls es eine Folge von solchen nicht-deterministischen Entscheidungen gibt, die zu einer Lösung führen.

⁹ Besser in dem Sinne, dass man mit polynomialer Laufzeit auskommt. Es kann durchaus sein, dass ein anderer Algorithmus in allen (oder fast allen) Fällen polynomial besser als der angegebene ist.

dann den größten Teil der SNOMED-Wissensbasis mit Hilfe der Beschreibungslogik \mathcal{EL}^{++} beschreiben, für die effiziente Schlussfolgerungsverfahren zur Verfügung stehen [2]. Dies heißt, dass man (relativ) ausdrucksschwache Formalismen verwenden kann, um solches Wissen zu repräsentieren, aber relativ effiziente Verfahren zur Verfügung hat, um Anfragen an solche Wissensbasen zu beantworten.

Neben der Entwicklung neuer Schlussverfahren für weitere Beschreibungslogiken spielen auch Kombinationen mit anderen Wissensrepräsentationsformalismen sowie die Unterstützung durch Knowledge-Engineering-Werkzeuge eine wichtige Rolle in der aktuellen Forschung auf dem Gebiet der Beschreibungslogiken.

4.4 Ausblick

In diesem Kapitel haben wir einige Kern-Fragestellungen auf dem Gebiet der Wissensrepräsentation und Wissensverarbeitung kennengelernt. Nach einem kleinen Streifzug durch verschiedene Spielarten des Wissens, konnten wir uns zunächst davon überzeugen, dass es sinnvoll ist, in wissensbasierten Systemen das zu repräsentierende Wissen von den wissensverarbeitenden Prozeduren separiert zu halten. Ferner wurde die grundlegende Rolle der Logik für die Wissensrepräsentation beleuchtet. So erlaubt die Verwendung von logischen Methoden es, Repräsentationsformalismen eine exakte Semantik zu geben: dies ist eine wesentliche Grundvoraussetzung dafür, verschiedene Repräsentationsformalismen hinsichtlich ihrer Ausdrucksstärke zu vergleichen und (sofern möglich) Übersetzungen zwischen diesen Formalismen anzugeben.

Exemplarisch haben wir dann einen Repräsentationsformalismus vorgestellt, in dem sich terminologisches wie auch Faktenwissen repräsentieren lässt. Diesen Formalismus, die Beschreibungslogik \mathcal{ALC} , kann man verstehen als ein Fragment der Prädikatenlogik 1. Stufe, das im Vergleich zu dieser Logik aber deutlich ausdrucksschwächer ist. Wir konnten uns davon überzeugen, dass die Beschränkung der Ausdrucksstärke jedoch in einem entscheidenden Vorteil mündet, wenn es darum geht, Inferenzanfragen an eine Wissensbasis zu beantworten, die nur die Ausdrucksmittel dieses Formalismus verwendet. Neben diesem im Detail beleuchteten Formalismus haben wir kurz alternative Formalismen vorgestellt, die deutlich ausdrucksstärker oder aber erheblich ausdrucksschwächer sind, dafür aber bessere Berechnungseigenschaften für wichtige Inferenzdienste aufweisen.

Beschreibungslogiken stellen ein prominentes, aber beileibe nicht das einzige Paradigma auf dem Gebiet der Wissensrepräsentation dar. Für die Repräsentation von regelbasiertem (also zum Beispiel konditionalem oder kausalem) Wissen gibt es eine ganze Reihe von weiteren Formalismustypen. Um zum Beispiel regelbasiertes Wissen als Default-Annahmen repräsentieren, bietet es sich an, diese im Rahmen der *Antwortmengen*-Programmierung (*answer set programming*) [14] darzustellen. Für die Repräsentation temporalen oder räumlichen Wissens bieten sich Constraint-basierte Formalismen an, um Inferenzen bezüglich den qualitativen Beziehungen zwischen zeitlichen oder räumlichen Entitäten zu ziehen [28].

Insgesamt konnte dieses Kapitel natürlich nur einen oberflächlichen Einstieg in das Gebiet der Wissensrepräsentation geben. Speziell wurden die Gebiete, die in anderen Teilen dieses Buches angesprochen werden, ausgelassen. Wer Interesse an dem Gebiet gefunden hat und sich weiter

informieren möchte, dem sei das Lehrbuch von Ron Brachman und Hector Levesque empfohlen [7]. Aktuelle Forschungsergebnisse findet man zum Beispiel in den Proceedings der zweijährlich stattfindenden *International Conference on Knowledge Representation and Reasoning (KR)*. Aber auch die bedeutenden internationalen KI-Konferenzen *IJCAI*, *AAAI* und *ECAI* haben jeweils große Sektionen zum Thema Wissensrepräsentation. Zudem findet man interessante Beiträge in dem Gebiet in den wissenschaftlichen Zeitschriften *Artificial Intelligence* und *Journal of Artificial Intelligence Research*.

Literaturverzeichnis

- [1] Baader, F. (1990). A formal definition for expressive power of knowledge representation languages. In [10].
- [2] Baader, F., Brandt, S., und Lutz, C. (2005). Pushing the envelope. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*, pages 364–369. Professional Book Center.
- [3] Baader, F., Bürkert, H.-J., Heinsohn, J., Hollunder, B., Müller, J., Nebel, B., Nutt, W., und Profitlich, H.-J. (1991). Terminological knowledge representation: A proposal for a terminological logic. In *International Workshop on Terminological Logics*, Dagstuhl, Germany. DFKI. DFKI Document D-91-13.
- [4] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., und Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.
- [5] Baader, F. und Nutt, W. (2003). Basic description logics. In [4], pages 43–95.
- [6] Beierle, C. und Kern-Isbner, G. (2000). *Methoden wissenbasierter Systeme*. Vieweg.
- [7] Brachman, R. J. und Levesque, H. J. (2004). *Knowledge Representation and Reasoning*. Elsevier.
- [8] Calì, A., Gottlob, G., und Lukasiewicz, T. (2012). A general datalog-based framework for tractable query answering over ontologies. *Journal of Web Semantics*, 14:57–83.
- [9] Doyle, J. und Patil, R. S. (1991). Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence*, 48(3):261–298.
- [10] ECAI-90 (1990). *Proceedings of the 9th European Conference on Artificial Intelligence (ECAI-90)*, Stockholm, Sweden. Pitman.
- [11] Fagin, R., Halpern, J. Y., Moses, Y., und Vardi, M. Y. (1995). *Reasoning About Knowledge*. MIT Press.
- [12] Fitting, M. C. (1993). Basic modal logic. In Gabbay, D. M., Hogger, C. J., und Robinson, J. A., editors, *Handbook of Logic in Artificial Intelligence and Logic Programming – Vol. 1: Logical Foundations*, pages 365–448. Oxford University Press, Oxford, UK.
- [13] Garey, M. R. und Johnson, D. S. (1979). *Computers and Intractability—A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA.
- [14] Gelfond, M. (2008). Answer sets. In van Harmelen, F., Lifschitz, V. und Porter, B., editors, *Handbook of Knowledge Representation*, pages 285–316. Elsevier.
- [15] Gettier, E. (1963). Is justified true belief knowledge? *Analysis*, 23:121–123.
- [16] Group, O. W. (2009). OWL 2 Web Ontology Language.
<http://www.w3.org/TR/owl2-overview/>.

- [17] Hanks, S. und McDermott, D. (1987). Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33(3):379–412.
- [18] Hollunder, B., Nutt, W., und Schmidt-Schauß, M. (1990). Subsumption algorithms for concept description languages. In [10], pages 348–353.
- [19] Horrocks, I. (1998). Using an expressive description logic: FaCT or fiction? In *Principles of Knowledge Representation and Reasoning: Proceedings of the 6th International Conference (KR-98)*, pages 636–647, Trento, Italy. Morgan Kaufmann.
- [20] Horrocks, I., Kutz, O., und Sattler, U. (2006). The even more irresistible SROIQ. In *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2–5, 2006*, pages 57–67. AAAI Press.
- [21] Horrocks, I., Patel-Schneider, P. F., und van Harmelen, F. (2003). From SHIQ and RDF to OWL: the making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26.
- [22] Levesque, H. J. und Brachman, R. J. (1987). Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93.
- [23] Nebel, B. (1990). *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, Heidelberg, New York.
- [24] Nebel, B. (1996). Artificial intelligence: A computational perspective. In Brewka, G., editor, *Essentials in Knowledge Representation*, Studies in Logic, Language and Information, pages 237–266. CSLI Publications, Stanford, CA.
- [25] Nebel, B. (1999). Frame-based systems. In Wilson, R. A. und Keil, F., editors, *MIT Encyclopedia of the Cognitive Sciences*, pages 324–325. MIT Press, Cambridge, MA.
- [26] Nebel, B. (2000). On the compilability and expressive power of propositional planning formalisms. *Journal of Artificial Intelligence Research*, 12:271–315.
- [27] Papadimitriou, C. H. (1993). *Computational Complexity*. Addison Wesley.
- [28] Renz, J. und Nebel, B. (2007). Qualitative spatial reasoning using constraint calculi. In Aiello, M., Pratt-Hartmann, I., und van Benthem, J., editors, *Handbook of Spatial Logics*, pages 161–215. Springer.
- [29] Schild, K. (1991). A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 466–471, Sydney, Australia. Morgan Kaufmann.
- [30] Schmidt-Schauß, M. und Smolka, G. (1991). Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26.
- [31] Schmolze, J. G. und Woods, W. A. (1992). The KL-ONE family. In Lehmann, F., editor, *Semantic Networks in Artificial Intelligence*. Pergamon Press.
- [32] Schulz, S., Suntisrivaraporn, B., Baader, F., und Boekera, M. (2007). SNOMED reaching its adolescence: Ontologists' and logicians' health check. *International Journal of Medical Informatics*, 78(1):S86–S94.
- [33] von Kutschera, F. (1982). *Grundfragen der Erkenntnistheorie*. Walter de Gruyter, Berlin.
- [34] Woods, W. A. (1975). What's in a link: Foundations for semantic networks. In Bobrow, D. G. und Collins, A. M., editors, *Representation and Understanding: Studies in Cognitive Science*, pages 35–82. Academic Press, New York, NY.
- [35] Zachte, E. (2012). Wikipedia Statistik. [Online; besucht 22. Mai 2012].

5 Automatische Inferenz

Peter Baumgartner

5.1 Einleitung

Das Gebiet des automatischen Schließens¹ beschäftigt sich mit dem Entwurf und der Implementierung von Computerprogrammen, die in der Lage sind mathematische Theoreme automatisch zu beweisen. So oder so ähnlich lautet eine gängige Kurzbeschreibung. Obwohl sicherlich nicht falsch, greift sie doch etwas zu kurz. Tatsächlich ist das Gebiet heutzutage hauptsächlich durch andere Anwendungsgebiete motiviert.

Ein solches Anwendungsgebiet ist „Programmverifikation“. Eine klassische Herangehensweise ist, durch logische Formeln Vor- und Nachbedingungen imperativer Programme anzugeben. Zum Beispiel so:

```
{ t ≥ 0 }
    t := t+1;
    a[t] := x;
{ t > 0 ∧ a[t] = x }
```

Eine typische Beweisaufgabe ist dann, partielle Korrektheit nachzuweisen: Falls die Voraussetzung $t \geq 0$ gilt und das Programm terminiert, dann gilt auch die Nachbedingung $t > 0 \wedge a[t] = x$.

Zur Programmverifikation dieser Art werden meist interaktive Systeme verwendet. Beispiele sind HOL [25], Isabelle [45], KIV [48], ACL2 [32] und KeY [1]. Interaktion ist nötig, um die „schwierigen“, weil Intuition erfordernenden Beweisschritte (z.B. das Finden von Schleifeninvarianten) dem Benutzer zu überlassen. „Kleine“ Routinebeweise überlässt man automatischen Theorembeweisern. Dadurch kann der gesamte Arbeitsaufwand beträchtlich verkleinert werden. Siehe [12] für eine umfassende Studie über die Kombination von Isabelle/HOL mit einigen der besten automatischen Beweissystemen.

Gerade in den letzten Jahren gewinnen aber auch vollautomatische Systeme zur Programmanalyse und Verifikation immer mehr an Bedeutung. Eines der ersten Systeme ist „Extended Static Checking“ [20], welches imperativer Programme auf mögliche Fehler wie Division durch Null, Arraygrenzenüberschreitung und Nullpointerzugriffe untersucht. Dies geschieht im Kern durch symbolische Auswertung eines bezüglich Datentypen und Kontrollfluss *abstrahierten* Programmes. Offensichtlich kann die Qualität solcher Analysen mit der Genauigkeit der Abstraktionen gesteigert werden, was aber gleichzeitig höhere Anforderungen an die verwendeten automatischen Beweissysteme stellt. Dafür werden heutzutage in erster Linie sogenannte *SMT-Beweiser*

¹ Die Begriffe „Inferenz“ und „Schließen“ werden in diesem Kapitel als Synonyme betrachtet.

eingesetzt (Satisfiability Modulo Theories) [44, 47], welche dedizierte Subsysteme für wichtige Datenstrukturen wie Arrays, Listen und Records, und Arithmetik integrieren. Ein Überblick über die weitreichende Forschung dazu würde den Rahmen dieses Kapitels sprengen (siehe Literaturhinweise weiter unten).

Ein anderes Anwendungsgebiet ist „Wissensrepräsentation“, siehe Kapitel 4. Die in diesem Bereich verwendeten Tableaux-Algorithmen können direkt dem Bereich des automatischen Schließen zugeordnet werden. Alternativ dazu können für gewisse Teilsprachen auch allgemeinere Resolutionsverfahren (siehe unten) eingesetzt werden [19, 31, z.B.]. Auch andere Bereiche der KI verwenden Methoden, die ihren Ursprung im Automatischen Schließen haben. Beispiele sind „Planen“ (Kapitel 10), „Nichtmonotones Schließen“ (Kapitel 6), Logikprogrammierung, und, gerade in letzter Zeit immer bedeutender, effiziente Theorembeweiser für die Aussagenlogik.

Aus diesen Betrachtungen sollte deutlich werden, dass „automatisches Schließen“ nicht auf den Beweis mathematischer Theoreme beschränkt ist.

Und mathematisches Theorembeweisen?

Der Wunsch, vollautomatisch (schwierige) mathematischer Theoreme beweisen zu können, hat die ursprüngliche Motivation des Gebietes geliefert. Der vielleicht spektakulärste Erfolg ist der vollautomatische Beweis eines 50 Jahre lang offenen Problems, der sogenannten Robbins-Vermutung [42]. Sie besagt das Folgende: Gegeben sei die folgende Basis der Booleschen Algebra (Huntington, 1933):

$$\begin{array}{ll} x \vee y = y \vee x & (\text{K – Kommutativität}) \\ (x \vee y) \vee z = x \vee (y \vee z) & (\text{A – Assoziativität}) \\ \neg(\neg x + y) + \neg(\neg x + \neg y) = x & (\text{H – Huntington Gleichung}) \end{array}$$

Die *Robbins-Vermutung* besagt nun, falls die Huntington Gleichung durch die Gleichung

$$\neg(\neg(x + y) + \neg(x + \neg y)) = x \quad (\text{R – Robbin Gleichung})$$

ersetzt wird, erhält man immer noch eine Boolesche Algebra. Der automatische Beweis gelang mit Bill McCunes System EQP [40]. Die Laufzeit betrug über 10 Tage.

Andere Systeme wurden erfolgreich eingesetzt, um bislang offene Probleme aus dem Bereich der finiten Algebra zu lösen [24, 54, 58]. All diese Probleme haben gemeinsam, dass ihre Lösungen keine große „schöpferische Phantasie“ benötigen und mehr auf kombinatorischem „Herumprobieren“ beruhen – „Schöpferisches Beweisen“ im Bereich der Mathematik ist nach wie vor in erster Linie Menschensache.

Freilich wurden auch Systeme entwickelt die menschliche Herangehensweisen beim Beweis mathematischer Theoreme simulieren. Sie fanden nützliche Anwendung als Assistenzsysteme in eingeschränkten Domänen, z.B. um Hilfestellung beim Erlernen mathematischer Theorien zu geben. Mit gutem Erfolg werden Techniken des maschinellen Lernens eingesetzt um aus einer (großen) Library von Axiomen und Lemmas relevante Prämissen zum Beweis einer konkreten Vermutung zu bestimmen [35]. Auf diese Entwicklungen genauer einzugehen würde jedoch den Rahmen dieses Kapitels sprengen.

Logisches Schließen.

Wir betrachten die folgenden beiden Aussagen:

- A_1 : Sokrates ist ein Mensch
- A_2 : Alle Menschen sind sterblich

Unter den Annahmen A_1 und A_2 wird man *intuitiv* schließen, dass Sokrates sterblich ist. Dieses Kapitel befasst sich mit auf mathematischer Logik basierenden Verfahren, um solche Schlussfolgerungen auf eine mathematisch fundierte Basis zu stellen. Eine naheliegende Formalisierung des Beispiels oben (mit Prädikatenlogik) ist wie folgt:

- A_1 : $\neg\text{mensch}(\neg\text{sokrates})$
- A_2 : $\forall x : (\text{mensch}(x) \Rightarrow \neg\text{sterblich}(x))$

Nun folgt aus rein formalen Gründen $\neg\text{sterblich}(\neg\text{sokrates})$ aus A_1 und A_2 , in Zeichen $\{A_1, A_2\} \models \neg\text{sterblich}(\neg\text{sokrates})$. Folgerung aus „rein formalen Gründen“ bedeutet, dass man von den gedachten Inhalten der Aussagen abstrahiert und sich rein auf die *Form* der Aussagen abstützt (statt „ $\neg\text{mensch}$ “ und „ $\neg\text{sterblich}$ “ könnten auch Symbole wie P und Q verwendet werden).

Die Frage, ob eine Folgerungsbeziehung oder „Nicht-Folgerungsbeziehung“ gilt, ist sogar in ganz einfachen Fällen nicht immer sofort zu sehen, wie die folgende Liste deutlich machen soll.

- | | |
|---|--|
| 1. $\{A_1, A_2\} \models \neg\text{sterblich}(\neg\text{sokrates})$ | (wahr) |
| 2. $\{A_1, A_2\} \models \neg\text{sterblich}(\neg\text{apollo})$ | (falsch) |
| 3. $\{A_1, A_2\} \not\models \neg\text{sterblich}(\neg\text{sokrates})$ | (falsch – schon alleine weil (1) wahr ist) |
| 4. $\{A_1, A_2\} \not\models \neg\text{sterblich}(\neg\text{apollo})$ | (wahr – schon alleine weil (2) falsch ist) |
| 5. $\{A_1, A_2\} \models \neg\neg\text{sterblich}(\neg\text{sokrates})$ | (falsch) |
| 6. $\{A_1, A_2\} \models \neg\neg\text{sterblich}(\neg\text{apollo})$ | (falsch) |

Die Folgerungsbeziehung (6) gilt zwar nicht, wenn man die Standardsemantik der Prädikatenlogik erster Stufe zu Grunde legt, wohl aber unter der *closed world assumption* und anderen nichtmonotonen Semantiken (vgl. Kapitel 6 über nichtmonotone Wissensrepräsentation).

Es ist deshalb wichtig, die Semantik der Prädikatenlogik, insbesondere die der Folgerungsbeziehung exakt zu definieren. Damit ist ein Referenzpunkt gegeben, an dem sich Inferenzverfahren bezüglich Korrektheit und Vollständigkeit messen lassen.

Inhalt

In diesem Kapitel konzentrieren wir uns auf Standardverfahren für die Aussagenlogik und die Prädikatenlogik erster Stufe, die sich gut zur Automatisierung eignen. Die Schwerpunkte liegen hierbei auf dem Davis-Putnam-Logemann-Loveland Verfahren für die Aussagenlogik und dem Resolutionsverfahren für die Prädikatenlogik. Letzteres wird recht detailliert besprochen, in der Variante der sogenannten geordneten Resolution, welche den derzeit besten Implementierungen zu Grunde liegt.

Wichtige andere Logiken und Verfahren dafür, auf die hier nicht näher eingegangen werden kann, sind modale Logiken (z.B. Temporallogiken), Logiken höherer Stufe, nichtmonotone Logiken und substrukturelle Logiken.

5.2 Entwurf automatischer Inferenzsysteme

Leistungsfähige automatische Inferenzsysteme sind komplexe Softwareprogramme, deren Entwurf entlang der folgenden Stufen beschrieben werden kann:

Logik/Dienst → Kalkül → Beweisprozedur → Implementierung

Logik und Dienst

Sicherlich ist die Wahl einer geeigneten Logik ein wichtiger Aspekt jeder formallogischen Modellierung und geht als ein zentraler Parameter in die Konstruktion eines automatischen Inferenzsystems ein. Die Wahl der Logik ist aber nicht der einzige zu bestimmende Parameter.

In der Frühzeit des automatischen Schließens, in den 60er Jahren, wurde alleine *Theorembelegen* als Anwendung angegangen, im Wortsinn also die Aufgabe, eine gegebene Formel als allgemeingültig zu beweisen. Entsprechende korrekte und vollständige Verfahren, die bis heute (in stark verbesserter Form) gebräuchlich sind, werden weiter unten beschrieben.

Mit der „Entdeckung“ der Logik für allgemeinere Wissensrepräsentationszwecke änderte sich das Bild, und die Liste der nützlichen *Dienste* eines Beweisers wurde länger:

Folgerung: Gegeben seien eine Formelmenge M und eine Formel F . Zu bestimmen ist ob $M \models F$ gilt.

Abduktion: Gegeben seien eine Formelmenge M und eine Formel F , sowie eine Formelmenge E („Mögliche Erklärungen“). Zu bestimmen ist eine Menge $A \subseteq E$, so dass (i) $M \cup A$ erfüllbar ist, und (ii) $M \cup A \models F$ gilt. Die Menge A heißt *abduktive Erklärung* (für F).

Es kann weiter gefordert werden, dass A eine minimale Menge ist die (i) und (ii) erfüllt.

Konsistenz: Gegeben sei eine Formelmenge M . Das Inferenzsystem sollte nachweisen können dass M erfüllbar ist, falls dies der Fall ist. Da dieser Dienst im Gegensatz zum Folgerungsdienst nicht mehr semi-entscheidbar ist, verwendet man oft entscheidbare Teilklassen der Pädikatenlogik, wie etwa „Beschreibungslogiken“, siehe Kapitel 4.

Modellberechnung: eine schärfere Formulierung des Konsistenzdienstes: Falls die gegebene Formelmenge M erfüllbar ist, sollte das Inferenzsystem ein Modell oder eine Repräsentation eines Modells für M liefern können.

Modellüberprüfung: Zu bestimmen ist ob eine gegebene Interpretation I ein Modell für eine gegebene Formelmenge M ist.

Ein gutes Beispiel dafür, dass bei der Wahl von Logik und Dienst Spielraum besteht ist die Aufgabenstellung „Planen“ (siehe Kapitel 10). In der Literatur wurden recht unterschiedliche logikbasierte Ansätze vorgeschlagen. Demnach kann ein Planungsproblem dargestellt werden über

- die Allgemeingültigkeit einer prädikatenlogischen Formel [27, 30, 60], oder
- die Erfüllbarkeit einer aussagenlogischen Formel [33] bezüglich klassischer Semantik, oder
- die Erfüllbarkeit einer aussagenlogischen Formel bezüglich nichtmonotonen Semantiken [21, 38, 43].

Kalkül

Unter einem *Kalkül* (*für eine gegebene Logik*) versteht man, grob gesprochen, ein effektives Verfahren welches schrittweise aus einer gegebenen Formel oder Formelmenge (den *Hypothesen*) neue Formeln erzeugt, bis ein gewisses Abbruchkriterium erfüllt ist.

Genauer gesagt, besteht ein *Kalkül* aus

- einer (entscheidbaren) Menge von Formeln, genannt *Axiome*,
- einer Kollektion von *Inferenzregeln*, die beschreiben wie aus gegebenen Formeln neue Formeln gewonnen werden können, und
- einer Vorschrift, genannt *Ableitungsbegriff*, die beschreibt wie die Axiome, Hypothesen und Inferenzregeln verwendet werden sollen.

Ein Kalkül ist demnach ein rein syntaktisches („mechanisches“) Verfahren.

Als Beispiel betrachten wir den sogenannten *Hilbert-Kalkül* (in seiner Aussagenlogischen Variante). Dazu benötigen wir zunächst die folgenden Formelschemata:

$$\begin{array}{ll} (\mathcal{F} \Rightarrow (\mathcal{G} \Rightarrow \mathcal{F})) & \text{(Prämissenbelastung)} \\ ((\mathcal{F} \Rightarrow (\mathcal{G} \Rightarrow \mathcal{H})) \Rightarrow ((\mathcal{F} \Rightarrow \mathcal{G}) \Rightarrow (\mathcal{F} \Rightarrow \mathcal{H}))) & \text{(Selbstdistributivität)} \\ ((\neg \mathcal{F} \Rightarrow \neg \mathcal{G}) \Rightarrow (\mathcal{G} \Rightarrow \mathcal{F})) & \text{(Kontraposition)} \end{array}$$

Die Axiome des Hilbert Kalküls bestehen aus allen Instanzen dieser Formelschemata, d.h. aus allen aussagenlogischen Formeln, die man durch uniforme Ersetzung von \mathcal{F}, \mathcal{G} und \mathcal{H} durch beliebige aussagenlogische Formeln erhält.

Der Hilbert Kalkül hat nur eine Inferenzregel, den *Modus Ponens*:

$$\text{MP} \quad \frac{\begin{array}{c} F \\ F \Rightarrow G \end{array}}{G}$$

wobei F und G beliebige aussagenlogische Formeln sind.

Eine Inferenzregel ist generell ein schematischer Ausdruck der Form $\frac{P_1 \dots P_n}{C}$, wobei $n \geq 0$. Eine Instanz einer Inferenzregel wird auch *Inferenz* genannt, die Formeln P_1, \dots, P_n heißen *Prämissen*, und die Formel C heißt *Konklusion*.

Die Modus Ponens Inferenzregel ist also nichts anderes als die intuitiv erwartete Schlussregel „Aus F und $F \Rightarrow G$ folgere G “.

Eine *Ableitung einer Formel F aus einer Menge von Formeln M* (den *Hypothesen*) ist eine endliche Folge von Formeln, die in F endet, und bei der jedes Element ein Axiom aus M ist oder die Konklusion einer MP Inferenz mit Prämissen aus vorhergehenden Elementen ist. Eine Ableitung von F aus den Hypothesen $\{H_1, \dots, H_m\}$ liefert einen Beweis der Tautologie $H_1 \wedge \dots \wedge H_n \Rightarrow F$.

Der Hilbert-Kalkül ist *deduktionsvollständig* und korrekt, d.h. für jede Tautologie (die mit den Junktoren \Rightarrow und \neg aufgebaut ist) gibt es eine Ableitung (ohne Hypothesen), aber eben nur für Tautologien. Kalküle dieser Art, verallgemeinert für die Prädikatenlogik, wurden bereits in den

1920-er Jahren entwickelt. Sie sind weder für die Beweissuche durch Menschen noch durch Maschinen geeignet. Besser geeignet sind Kalküle die von Anfang an in Ableitungen die zu beweisende Formel berücksichtigen. Bekannte Verfahren dazu sind der Kalkül des natürlichen Schließens, der Gentzen Kalkül, der Tableau Kalkül und insbesondere der Resolutionskalkül.

Bei *widerlegungsvollständigen* Kalkülen (z.B. Tableau, Resolution) verzichtet man auf die Möglichkeit, alle Tautologien ableiten zu können. Man fordert stattdessen nur, dass jede unerfüllbare Formel als solche nachgewiesen werden kann. Der Beweis, dass eine Formel F tautologisch ist, erfolgt dann durch Ableitung einer ausgezeichneten unerfüllbaren Formel aus der Hypothesenmenge $\{\neg F\}$. Falls dies gelingt, folgt mit der Korrektheit des Kalküls sofort dass F tautologisch ist.

So kann zum Beispiel im Resolutionskalkül, dem wichtigsten widerlegungsvollständigen Verfahren, die Tautologie $A \vee \neg A$ nicht abgeleitet werden. Diese vermeintliche Schwäche ist tatsächlich eine Stärke, schränkt sie doch die Menge der ableitbaren Formeln ein.

Beweisprozedur

Die Kalkülebene eignet sich sehr gut zur abstrakten Spezifikation von Inferenzsystemen und der Analyse theoretischer Eigenschaften, zum Beispiel Korrektheit und (Widerlegungs-)vollständigkeit, nicht aber zur direkten Implementierung.

So sind Kalküle meist nichtdeterministisch formuliert, d.h., zu einem Zeitpunkt können mehrere Inferenzregeln anwendbar sein. Nichtdeterminismen werden beim Übergang zu *Beweisprozeduren* aufgelöst. Dabei ist zwischen zwei Arten zu unterscheiden: „Don't-care“-Nichtdeterminismen ermöglichen eine beliebige Auswahl unter den möglichen Alternativen ohne die Vollständigkeit zu verletzen. Um eine möglichst günstige Steuerung der Beweissuche zu erreichen, kann diese Auswahl deshalb nach heuristischen Prinzipien erfolgen. Bei „Don't-know“-Nichtdeterminismen sind potenziell alle Alternativen zu untersuchen. Typischerweise wird dies durch eine Backtrack-Steuerung innerhalb der Beweisprozedur realisiert.

Für beide Fälle aber kann man zusammenfassen:

$$\text{Beweisprozedur} = \text{Kalkül} + \text{Steuerung}$$

Wir werden weiter unten zwei Beweisprozeduren sehen: Den DPLL Algorithmus für Aussagenlogik in Abschnitt 5.5, und die „Given Clause Loop“ Prozedur für Resolution.

Implementierung

Schließlich kommen beim Übergang zur *Implementierung* neben der eigentlichen Kodierung in einer Programmiersprache noch ausgesparte Details zu Datenstrukturen hinzu; auch effizienzsteigernde Maßnahmen wie Indexierungstechniken [26] werden erst hier eingebrochen. Siehe [57] für eine sehr gute Diskussion.

5.3 Prädikatenlogik erster Stufe

Die Syntax und die informelle Semantik der PL1 wurde bereits in Abschnitt 4.2.2 eingeführt. Die formale Semantik soll hier explizit definiert werden – schließlich liefert sie den Referenzpunkt

für die automatischen Verfahren. Ausführlichere Einführungen findet man *zum Beispiel* in den sehr guten Büchern [23, 29, 51].

Die Standardsemantik der PL1 geht auf eine Arbeit des polnischen Logikers Alfred Tarski 1936 zurück. Man spricht deshalb auch von der *Tarski-Semantik*. Um sie zu definieren, benötigt man zunächst eine nicht-leere Menge U , genannt *Universum* (oder *Grundmenge*, *Individuenbereich*), zum Beispiel die Menge der natürlichen Zahlen.

Dann werden den Prädikats- und Funktionssymbolen Relationen und Funktionen auf U zugeordnet. Solch eine Zuordnung wird formal durch eine Abbildung \cdot^I beschrieben, welche zu einer gegebenen Signatur Σ^2

- jedem n -steligen Prädikatssymbol P eine n -stellige Relation $P^I \subseteq U^n$ zuordnet, und
- jedem n -steligen Funktionssymbol f eine n -stellige Funktion $f^I : U^n \mapsto U$ zuordnet.³

Das Universum U zusammen mit der Abbildung \cdot^I wird oft als ein Paar $I = \langle U, \cdot^I \rangle$ zusammengefasst und dann als *Interpretation* bezeichnet. Jedem Term, in dem keine Variablen vorkommen, wird durch eine Interpretation eindeutig ein Wert aus dem Universum zugeordnet: sei $t = f(t_1, \dots, t_n)$ ein Term (mit $n \geq 0$). Dann ist der Wert von t unter I , geschrieben als t^I , rekursiv definiert über die Gleichung $f(t_1, \dots, t_n)^I = f^I(t_1^I, \dots, t_n^I)$. Man beachte, dass die Rekursion bei Konstanten endet, für die ja $n = 0$ gilt.

Falls t Variablen enthält, ist diese Abbildung nicht definiert. Um dennoch einen Term mit Variablen auswerten zu können, wird zusätzlich eine *Belegung* benötigt: Eine Belegung v ordnet jeder Variablen x einen Wert d aus dem Universum U zu, und man schreibt dann $v(x) = d$. Damit wird die Rekursionsgleichung oben auf nahe liegende Weise auf alle Terme verallgemeinert:

$$t^{I,v} = \begin{cases} f^I(t_1^{I,v}, \dots, t_n^{I,v}) & \text{falls } t \text{ ein Term } f(t_1, \dots, t_n) \text{ ist,} \\ v(x) & \text{falls } t \text{ eine Variable } x \text{ ist} \end{cases}$$

Darauf aufbauend wird jedem Atom $P(t_1, \dots, t_n)$ eindeutig ein Wahrheitswert \mathbf{w} oder \mathbf{f} zugeordnet:

$$P(t_1, \dots, t_n)^{I,v} = \begin{cases} \mathbf{w} & \text{falls } \langle t_1^{I,v}, \dots, t_n^{I,v} \rangle \in P^I, \\ \mathbf{f} & \text{sonst} \end{cases}$$

Im letzten Schritt müssen die bisherigen Definitionen auf Formeln erweitert werden. Dazu bezeichne $v[x \mapsto d]$ diejenige Belegung, die x auf d abbildet, aber sonst mit v identisch ist. Wir definieren

$$\begin{aligned} (F \wedge G)^{I,v} &= \begin{cases} \mathbf{w} & \text{falls } F^{I,v} = \mathbf{w} \text{ und } G^{I,v} = \mathbf{w} \\ \mathbf{f} & \text{sonst} \end{cases} \\ (F \vee G)^{I,v} &= \begin{cases} \mathbf{w} & \text{falls } F^{I,v} = \mathbf{w} \text{ oder } G^{I,v} = \mathbf{w} \\ \mathbf{f} & \text{sonst} \end{cases} \\ (\neg F)^{I,v} &= \begin{cases} \mathbf{w} & \text{falls } F^{I,v} = \mathbf{f} \\ \mathbf{f} & \text{sonst} \end{cases} \end{aligned}$$

² Eine Signatur besteht aus endlichen Mengen von Prädikats- und Funktionssymbolen.

³ Da $n = 0$ erlaubt sein soll, trifft diese Definition auch auf Konstantensymbole zu.

$$\begin{aligned}
 (F \Rightarrow G)^{I,v} &= (\neg F \vee G)^{I,v} \\
 (F \Leftrightarrow G)^{I,v} &= ((F \Rightarrow G) \wedge (G \Rightarrow F))^{I,v} \\
 (\forall x : F)^{I,v} &= \begin{cases} \mathbf{w} \text{ falls für alle } d \in U \text{ gilt: } F^{I,v[x \mapsto d]} = \mathbf{w} \\ \mathbf{f} \text{ sonst} \end{cases} \\
 (\exists x : F)^{I,v} &= \begin{cases} \mathbf{w} \text{ falls es ein } d \in U \text{ gibt mit } F^{I,v[x \mapsto d]} = \mathbf{w} \\ \mathbf{f} \text{ sonst} \end{cases}
 \end{aligned}$$

Die Semantik des Junktors \Rightarrow wird also auf die der Junktoren \neg und \vee zurückgeführt, und ähnlich für \Leftrightarrow . Bei dieser Definition wird stillschweigend angenommen dass die gegebene Interpretation I zu der auszuwertenden Formel F passt. Damit ist gemeint, dass die Signatur, die der Definition von \cdot^I zu Grunde liegt, mindestens die in F vorkommenden Prädikats- und Funktionssymbolen enthält. Im Folgenden treffen wir stets diese Annahme.

Beispiel 5.3.1. Sei $F = \forall x : P(x, f(x)) \wedge Q(g(a, z))$. Eine zu F passende Interpretation $I = \langle U, \cdot^I \rangle$ ist:

$$\begin{aligned}
 U &= \{0, 1, 2, \dots\} \\
 P^I &= \{(m, n) \mid m, n \in U \text{ und } m < n\} \\
 Q^I &= \{n \in U \mid n \text{ ist Primzahl}\} \\
 f^I &= \text{die Nachfolgerfunktion auf } U, \text{ also } f^I(n) = n + 1 \\
 g^I &= \text{die Additionsfunktion auf } U, \text{ also } g^I(m, n) = m + n \\
 a^I &= 2
 \end{aligned}$$

Falls nun zusätzlich eine Belegung v mit $v(z) = 3$ gegeben ist, so ergibt sich für den Wahrheitswert des zweiten Konjunktionsgliedes

$$\begin{aligned}
 Q(g(a, z))^{I,v} &= Q^I(g(a, z)^{I,v}) \\
 &= Q^I(g^I(a^{I,v}, z^{I,v})) \\
 &= Q^I(g^I(2, 3)) \\
 &= Q^I(2 + 3) = Q^I(5) \\
 &= \mathbf{w}.
 \end{aligned}$$

Das erste Konjunktionsglied, welches besagt „für alle $x \in \{0, 1, 2, \dots\}$ gilt: $x < x + 1$ “, wertet ebenfalls zu \mathbf{w} aus, und somit wertet auch die gesamte Formel F zu \mathbf{w} aus.

Im Folgenden werden wir gelegentlich den Begriff des Bindungsbereichs benötigen: Für eine Formel der Form $Qx : F$ mit $Q \in \{\forall, \exists\}$ heißt F Bindungsbereich der Quantifizierung Qx . Ein Vorkommen einer Variablen x in einer Formel heißt gebunden, falls x im Bindungsbereich einer Teilformel mit Quantifizierung $\forall x$ oder $\exists x$ vorkommt, ansonsten kommt x frei vor. Falls jedes Variabenvorkommen in einer Formel F gebunden ist, so heißt F auch Satz. Im Beispiel 5.3.1 kommt die Variable x (nur) gebunden in F vor, und die Variable z kommt frei vor. Die Formel F ist deshalb kein Satz.

Falls, allgemein, F ein Satz ist, so ist nicht schwer zu sehen, dass dessen Auswertung unabhängig von einer gegebenen Belegung ist. Für zwei beliebige Belegungen v_1 und v_2 gilt deshalb $F^{I,v_1} = F^{I,v_2}$ und der Ausdruck F^I bestimmt eindeutig das Ergebnis der Auswertung von F unter I und beliebiger Belegung. Falls F kein Satz ist, also freie Variablen x_1, \dots, x_m enthält, so kann man dennoch F^I schreiben, in dem man F implizit als Satz $\forall x_1 \dots \forall x_m : F$ auffasst. Diese Übereinkunft soll bei der Definition der folgenden, zentralen Begriffe getroffen werden.⁴

⁴ In der Literatur wird auch anders vorgegangen.

Definition 5.3.1 (Modell, erfüllbar, gültig, logische Folgerung). Seien F und G Formeln. Falls es eine Interpretation I gibt so dass $F^I = \mathbf{w}$, so heißt F erfüllbar und I ist ein Modell für F , i.Z. $I \models F$. Falls für jede Interpretation I $F^I = \mathbf{w}$ gilt, so heißt F (allgemein) gültig, i.Z. $\models F$.

Die Notation $I \not\models F$ bedeutet, dass I kein Modell für F ist. Falls F kein Modell hat, dann heißt F unerfüllbar. Ein gültiger Satz wird auch als Tautologie bezeichnet.

Im Beispiel 5.3.1 ist I kein Modell für F , da $I \not\models \forall z : F$. Zum Beispiel ist $F^{I,v[z \mapsto 2]} = \mathbf{f}$.

Im Folgenden werden wir eine nahe liegende Verallgemeinerung des Erfüllbarkeitsbegriffs aus Definition 5.3.1 benötigen. Sei $M = \{F_1, \dots, F_n, \dots\}$ eine (nicht notwendigerweise endliche) Menge von Formeln. Dann heißt M erfüllbar, falls es eine Interpretation I gibt, die (simultan) Modell für alle $F \in M$ ist, i.Z. $I \models M$, unerfüllbar sonst. Falls jedes Modell für M auch Modell für eine Formel G ist, so heißt G eine logische Konsequenz von M , oder auch G folgt aus M , i.Z. $M \models G$. Dass G aus M folgt, ist äquivalent zur Aussage „für alle Interpretationen I : falls $I \models F$ für alle $F \in M$ dann auch $I \models G$ “.

Proposition 5.3.1 (Deduktionstheorem). Sei M eine Menge von Formeln und $F \Rightarrow G$ eine Formel, in der F keine freien Variablen enthält. Dann gilt $M \models F \Rightarrow G$ gdw. $M \cup \{F\} \models G$.

Typischerweise liegt mit M eine Axiomatisierung der Domäne vor. Im Beispiel der Robbin's Conjecture oben könnte $M = \{(K), (A)\}$ sein. Die schwierige Richtung der Robbin's Conjecture besteht darin $M \models (R) \Rightarrow (H)$ nachzuweisen.⁵ Mit dem Deduktionstheorem kann dann die Robbin's Gleichung (R) als zusätzliche Hypothese zu den Axiomen in M aufgenommen werden, und es reicht zu zeigen $M \cup \{(R)\} \models (H)$.

Dieses Muster, Theoreme in Implikationsform zunächst wie angedeutet aufzuspalten, liegt vielen Kalkülen des Automatischen Schließens zu Grunde. Mehr noch:

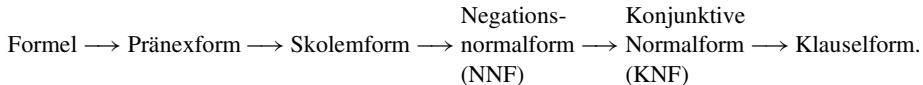
Anmerkung 5.3.1 (Refutationales Beweisen). Als weitere Konsequenz des Deduktionstheorems ist die Frage „Gilt $M \models G$?“ äquivalent zur Frage „Ist $M \cup \{\exists x_1, \dots, x_n : \neg G\}$ unerfüllbar?“, wobei x_1, \dots, x_n genau die in G vorkommenden freien Variablen sind.

Dieser Zusammenhang ist von fundamentaler Bedeutung. Um einen – im Wortsinn – „Theorem beweiser“ zu bauen, reicht es also aus, ein Verfahren zu implementieren, das Unerfüllbarkeit von Formelmengen nachweisen kann. In der Tat arbeiten alle Widerlegungsverfahren damit, zum Beispiel das Resolutionsverfahren (Abschnitte 5.6 und 5.7.2).

5.4 Normalformen

Die meisten modernen, automatischen Schlussverfahren arbeiten mit prädikatenlogischen Formeln, die in eine gewisse Normalform gebracht wurden. Im Hinblick auf die hier betrachteten Verfahren ist die sogenannte Klauselform am wichtigsten. Sie entsteht im letzten Schritt der folgenden Kette von Transformationen:

⁵ Die Formeln (K), (A), (R) und (H) werden hier als implizit allquantifiziert aufgefasst.



Wir besprechen diese Schritte der Reihe nach.

Schritt 1: Pränexform

Die erste Transformation verwendet die folgenden Formelschemata:

(1)	$(\mathcal{F} \Leftrightarrow \mathcal{G}) \Leftrightarrow ((\mathcal{F} \Rightarrow \mathcal{G}) \wedge (\mathcal{G} \Rightarrow \mathcal{F}))$	(\Leftrightarrow -Elimination)
(2)	$(\mathcal{F} \Rightarrow \mathcal{G}) \Leftrightarrow (\neg \mathcal{F} \vee \mathcal{G})$	(\Rightarrow -Elimination)
(3)	$\neg \neg \mathcal{F} \Leftrightarrow \mathcal{F}$	
(4)	$\neg(\mathcal{F} \vee \mathcal{G}) \Leftrightarrow (\neg \mathcal{F} \wedge \neg \mathcal{G})$	
(5)	$\neg(\mathcal{F} \wedge \mathcal{G}) \Leftrightarrow (\neg \mathcal{F} \vee \neg \mathcal{G})$	(deMorgansche Regeln)
(6)	$\neg \forall x : \mathcal{F} \Leftrightarrow \exists x : \neg \mathcal{F}$	
(7)	$\neg \exists x : \mathcal{F} \Leftrightarrow \forall x : \neg \mathcal{F}$	
(8)	$((\forall x : \mathcal{F}) \vee \mathcal{G}) \Leftrightarrow \forall x : (\mathcal{F} \vee \mathcal{G})$	falls x in \mathcal{G} nicht frei vorkommt
(9)	$(\mathcal{G} \vee (\forall x : \mathcal{F})) \Leftrightarrow \forall x : (\mathcal{G} \vee \mathcal{F})$	falls x in \mathcal{G} nicht frei vorkommt
(10)	$((\forall x : \mathcal{F}) \wedge \mathcal{G}) \Leftrightarrow \forall x : (\mathcal{F} \wedge \mathcal{G})$	falls x in \mathcal{G} nicht frei vorkommt
(11)	$(\mathcal{G} \wedge (\forall x : \mathcal{F})) \Leftrightarrow \forall x : (\mathcal{G} \wedge \mathcal{F})$	falls x in \mathcal{G} nicht frei vorkommt
(12)	$((\exists x : \mathcal{F}) \vee \mathcal{G}) \Leftrightarrow \exists x : (\mathcal{F} \vee \mathcal{G})$	falls x in \mathcal{G} nicht frei vorkommt
(13)	$(\mathcal{G} \vee (\exists x : \mathcal{F})) \Leftrightarrow \exists x : (\mathcal{G} \vee \mathcal{F})$	falls x in \mathcal{G} nicht frei vorkommt
(14)	$((\exists x : \mathcal{F}) \wedge \mathcal{G}) \Leftrightarrow \exists x : (\mathcal{F} \wedge \mathcal{G})$	falls x in \mathcal{G} nicht frei vorkommt
(15)	$(\mathcal{G} \wedge (\exists x : \mathcal{F})) \Leftrightarrow \exists x : (\mathcal{G} \wedge \mathcal{F})$	falls x in \mathcal{G} nicht frei vorkommt

Jede Instantiierung jedes dieser Schema mit beliebigen konkreten Formeln F und G für die Platzhalter \mathcal{F} und \mathcal{G} liefert eine Äquivalenz (eine Äquivalenz ist eine allgemeingültige Formel der Bauart $F \Leftrightarrow G$). Die Prädikatenlogik erster Stufe ist eine *extensionale Logik*, in der Teilformeln durch äquivalente Formeln ohne Änderung der semantischen Eigenschaften ersetzt werden dürfen. Dies rechtfertigt, die Formelschemata oben als Ersetzungsregeln aufzufassen. Durch wiederholte Anwendung (nur) von rechts nach links kann dann jede Formel F in eine äquivalente *Pränexform* überführt werden, welche die Form $Q_1 x_1 : \dots Q_n x_n : G$ hat, wobei $Q_i \in \{\forall, \exists\}$, für alle $i = 1, \dots, n$ und G eine quantorenfreie Formel ist, welche auch als die *Matrix* von F bezeichnet wird. Um die Äquivalenzen mit den Bedingungen „falls x in \mathcal{G} nicht frei vorkommt“ anzuwenden, kann es erforderlich sein, zunächst z.B. eine Teilformel $((\forall x : F) \vee G)$ durch die äquivalente Formel $((\forall x' : F') \vee G)$ zu ersetzen, wobei F' aus F durch Ersetzen jedes freien Vorkommens von x durch x' hervorgeht.

Dass diese Äquivalenzen – aufgefasst als Ersetzungsregeln – tatsächlich ausreichen um Pränexformen zu erhalten, kann man sich wie folgt plausibel machen: Mit den Schemata (1) und (2) können zunächst alle Vorkommen der Junktoren „ \Leftrightarrow “ und „ \Rightarrow “ eliminiert werden. Eine so erhaltene Formel enthält nur noch die Junktoren „ \wedge “, „ \vee “ und „ \neg “. Mit den Schemata (3)–(7) können dann die Negationszeichen nach innen gezogen werden. Man kann also insbesondere eine Formel erhalten, in der kein Quantor mehr in einer negierten Teilformel vorkommt. Schließlich können mit den Schemata (8)–(15) alle in konjunktiven oder disjunktiven Teilformeln vorkommenden Quantoren nach außen gezogen werden. Nach ausreichend vielen Regelanwendungen erhält man also eine Pränexform der gegebenen Formel. Es lässt sich sogar zeigen, dass das

Ergebnis dieses Prozesses unabhängig von der gewählten Regelanwendung ist und immer terminiert – eine für die Automatisierung wichtiger Aspekt.⁶

Schritt 2: Skolemform

O.b.d.A. sei nun ein *Satz* F in Pränexform gegebenen. (Zu jeder *Formel* kann man einen äquivalenten Satz erhalten, in dem die freien Variablen explizit durch äußere \forall -Quantifizierungen gebunden werden.) Ziel des zweiten Umformungsschritts ist es, alle Existenzquantoren in F zu eliminieren. Dies erfolgt durch wiederholte Anwendung des folgenden Umformungsschritts, so lange wie möglich. Das Gesamtergebnis wird dann als *Skolemform* von F bezeichnet.

Sei $F = \forall x_1, \dots, x_{j-1} : \exists x_j : Q_{j+1}x_{j+1} : \dots Q_n x_n : G$ in Pränexform mit $n \geq j \geq 0$, wobei $Q_{j+1}, \dots, Q_n \in \{\exists, \forall\}$. Falls $x_j \in \{x_{j+1}, \dots, x_n\}$ sei $G' = G$, andernfalls sei G' die Formel, die man aus G durch Ersetzung jedes Vorkommens der Variablen x_j durch den *Skolemterm* $f_{x_j}(x_1, \dots, x_{j-1})$ erhält. Das Ergebnis ist die Formel $F := \forall x_1, \dots, x_{j-1} : Q_{j+1}x_{j+1} : \dots Q_n x_n : G'$.

In jedem Schritt wird also genau ein Existenzquantor eliminiert, indem die existentiell quantifizierte Variable x_j in G durch einen Skolemterm $f_{x_j}(x_1, \dots, x_{j-1})$ ersetzt wird. Man beachte, dass dieser Vorgang nur stattfindet, falls $\exists x_j$ nicht durch eine weiter rechts stehende Quantifizierung derselben Variable verschattet wird.

Bei den einzelnen Schritten steht f_{x_j} für ein *Skolem-Funktionssymbol*, welches man sich aus dem Namen „ f “ und dem Namen der Variablen x_j zusammengesetzt denken kann. Die Rolle eines Skolemterms ist es, ein zu gegebenen allquantifizierten Variablen existierendes Objekt zu bezeichnen. So ist z.B. die Skolemisierung der Formel $\forall x : \exists y : y > x$ die Formel $\forall x : f_y(x) > x$. Wählt man in diesem Beispiel als Domäne die natürlichen Zahlen und interpretiert $>$ als die „größer als“ Relation, so sieht man leicht, dass auch die Skolemisierung z.B. in derjenigen Interpretation wahr ist, die f_y als die Nachfolgerfunktion auf natürlichen Zahlen interpretiert. In der Tat erhält Skolemisierung die Erfüllbarkeitseigenschaft der gegebenen Formel, Skolemisierung ist aber i.A. nicht Äquivalenz erhaltend. Um dies einzusehen, kann man im Beispiel eine Interpretation wählen mit z.B. $f_y(x) = 0$, für alle natürlichen Zahlen x , welche ein Modell für $\forall x : \exists y : y > x$ ist, nicht aber für $\forall x : f_y(x) > x$.

Unter der zu treffenden Annahme, dass die Matrix G der ursprünglichen Formel keine Skolem-Funktionssymbol enthält, wird auch im Laufe der Skolemisierung kein Skolemterm mit demselben Skolem-Funktionssymbol ein zweites Mal eingesetzt werden, und unter den gleich benannten existentiell quantifizierten Variablen wird immer die am weitesten innen stehende Quantifizierung eliminiert. So ist zum Beispiel die Skolemisierung von $\forall x \exists y \forall z \exists y : P(x, y, z)$ die Formel $\forall x \forall z : P(x, f_y(x, z), z)$, und nicht etwa $\forall x \forall z : P(x, f_y(x), z)$, was nicht korrekt wäre. (Letztendlich liefert die Semantik der Prädikatenlogik wie oben angegeben die Begründung dafür.)

Schritt 3: Negationsnormalform und Konjunktive Normalform

Die hier besprochene Transformation arbeitet lokal auf der Matrix G einer gegebenen Formel in Pränexform. Falls noch nicht geschehen, werden zunächst durch die im Schritt 1 angegebenen Äquivalenzschemata (1) und (2) die Junktoren \Leftrightarrow und \Rightarrow vollständig aus G eliminiert.

⁶ In der Sprache der Termersetzungssysteme ist das hier verwendete Regelsystem *konfluent* und *terminierend*.

Durch Anwendung der Äquivalenzschemata (3)–(5), so lange wie möglich, werden dann alle vorkommenden Negationszeichen soweit als möglich „nach innen“ geschoben und doppelte Negationen eliminiert. Im Ergebnis kommen Negationszeichen dann nur noch vor Atomen vor und man spricht von einer Formel in *Negationsnormalform (NNF)*. (Manche Beweisverfahren, siehe unten, arbeiten direkt auf NNF-Formeln.)

Wir betrachten nun die folgenden Äquivalenzschemata:

$$(16) \quad (\mathcal{F} \vee (\mathcal{G} \wedge \mathcal{H})) \Leftrightarrow ((\mathcal{F} \vee \mathcal{G}) \wedge (\mathcal{F} \vee \mathcal{H}))$$

$$(17) \quad ((\mathcal{G} \wedge \mathcal{H}) \vee \mathcal{F}) \Leftrightarrow ((\mathcal{F} \vee \mathcal{G}) \wedge (\mathcal{F} \vee \mathcal{H}))$$

Wiederum wird deren exhaustive Anwendung terminieren und alle Konjunktionen, die Teil einer Disjunktion sind, eliminieren. Im Ergebnis erhält man aus einer Matrix in NNF eine Matrix in *Konjunktiver Normalform (KNF)*, eine Konjunktion von Disjunktionen.

Die Assoziativitätsgesetze

$$(18) \quad ((\mathcal{F} \vee \mathcal{G}) \vee \mathcal{H}) \Leftrightarrow (\mathcal{F} \vee (\mathcal{G} \vee \mathcal{H}))$$

$$(19) \quad ((\mathcal{F} \wedge \mathcal{G}) \wedge \mathcal{H}) \Leftrightarrow (\mathcal{F} \wedge (\mathcal{G} \wedge \mathcal{H}))$$

rechtfertigen, als Schreiberleichterung Klammern wegzulassen. Jede KNF G kann damit geschrieben werden als

$$\begin{aligned} G &= F_1 \wedge \cdots \wedge F_n \quad \text{mit } n \geq 1, \text{ wobei für alle } i = 1, \dots, n: \\ F_i &= L_{i,1} \vee \cdots \vee L_{i,m_i} \quad \text{mit } m_i \geq 1, \text{ wobei jedes } L_{j,k} \text{ ein Atom} \\ &\quad \text{oder ein negiertes Atom ist.} \end{aligned}$$

Schritt 4: Klauselform

Der letzte Schritt beim Übergang zur Klauselform nutzt die Kommutativitäts- und Idempotenzgesetze

$$(20) \quad (\mathcal{F} \wedge \mathcal{G}) \Leftrightarrow (\mathcal{G} \wedge \mathcal{F}) \quad (21) \quad (\mathcal{F} \wedge \mathcal{F}) \Leftrightarrow \mathcal{F}$$

aus. Sie rechtfertigen, eine Matrix in KNF als Menge von Disjunktionen von Atomen oder negierten Atomen zu schreiben. Mit den für die KNF oben eingeführten Bezeichnungen erhält man dann die *Klauselmenge*

$$\{L_{1,1} \vee \cdots \vee L_{1,m_1}, \dots, L_{n,1} \vee \cdots \vee L_{n,m_n}\}.$$

Die Kommas stehen also für „ \wedge “. Weiterhin ist mit der Kommutativität von „ \vee “ leicht einzusehen, dass es auf die Reihenfolge der verbundenen Atome oder negierten Atome nicht ankommt. Man kann also $A \vee B \vee C$ schreiben, um z.B. die Formeln $((A \vee B) \vee C)$ oder $(B \vee (C \vee A))$ zu meinen.⁷

Allgemein wird definiert:

Definition 5.4.1 (Literal, Klausel, Klauselmenge). *Ein Literal ist ein Atom oder ein negiertes Atom. Eine Klausel ist eine (möglicherweise leere) Disjunktion von Literalen. Eine Klauselmenge ist eine (möglicherweise leere) Menge von Klauseln.*

⁷ In der Literatur werden Klauseln manchmal als *Mengen* aufgefasst. Das ist durch die Idempotenzegenschaft $(\mathcal{F} \vee \mathcal{F}) \Leftrightarrow \mathcal{F}$ gerechtfertigt.

Die leere Klausel wird als \square geschrieben. Mit \overline{L} wird das *Komplement* eines Literals L bezeichnet: für jedes Atom A ist $\neg\overline{A} = A$ und $\overline{\overline{A}} = \neg A$.

Die Definition 5.4.1 ist etwas allgemeiner als nötig, um eine Matrix in KNF zu repräsentieren, denn sowohl eine Klausel als auch eine Klauselmenge können leer sein. Semantisch steht die leere Klausel für eine unerfüllbare Formel (denn sie repräsentiert die leere Disjunktion, und um eine Disjunktion zu erfüllen, müsste mindestens ein Disjunktionsglied erfüllt werden, was aber nicht vorhanden ist) und die leere Klauselmenge steht für eine tautologische Formel (denn sie repräsentiert die leere Konjunktion, und, um eine Konjunktion zu erfüllen, müssten alle Konjunktionsglieder erfüllt werden, was trivialerweise der Fall ist).

Mit der Übereinkunft, dass alle Variablen der ursprünglichen Formel in Pränexform allquantifiziert sind, sind freilich auch alle Variablen der letztendlich erhaltenen Klauselmenge allquantifiziert.

Es sollte betont werden, dass alle besprochenen Umformungsschritte äquivalente Formeln liefern *bis auf Schritt 2 (Skolemform)*, welcher in der benötigten Richtung nur die Erfüllbarkeit erhält. Zusammenfassend:

Proposition 5.4.1. *Eine Formel F ist erfüllbar gdw. die Transformation von F in Klauselform erfüllbar ist.*

Man erinnere sich in diesem Zusammenhang an Bemerkung 5.3.1: Ein ursprüngliches Beweisproblem in Form der Frage „Gilt $\{A_1, \dots, A_n\} \models G$ “ kann damit auf die Frage „Ist die Formel $A_1 \wedge \dots \wedge A_n \wedge \exists x_1, \dots, x_n : \neg G$ unerfüllbar?“ reduziert werden, welche mit Satz 5.4.1 auf die Frage der Unerfüllbarkeit der Klauselform dieser Formel reduziert werden kann.

Probleme und Verbesserungen

Prinzipiell liefert Satz 5.4.1 zusammen mit Bemerkung 5.3.1 ein hinreichendes Werkzeug zum Einsatz refutationaler, auf Klauselogik basierender Verfahren als Theorembeweiser. Aus praktischer Sicht wirft die „naive“ Transformation wie oben beschrieben allerdings Probleme auf:

Exponentielles Aufblähen: Die Äquivalenzschemata (1), (16) und (17) duplizieren die Vorkommen von einer (oder zwei, im Fall von (1)) Teilformel(n). Damit geht bei wiederholter Anwendung im schlechtesten Fall eine exponentielle Vergrößerung der ursprünglichen Formel einher.

Das Problem kann vermieden werden durch Ersetzung jeder zur Aufblähung führenden Teilformeln F durch einen neuen „Namen“ $P_F(x_1, \dots, x_n)$, wobei x_1, \dots, x_n die in F frei vorkommenden Variablen sind und Hinzunahme einer „Definition“ $\forall x_1, \dots, x_n : P_F(x_1, \dots, x_n) \Leftrightarrow F$.

Skolemfunktionen: Statt der in Schritt 2 eingeführten Skolemterme können manchmal einfachere Skolemterme, mit weniger Variablen, eingeführt werden.

Zum Beispiel resultiert die Formel $F = \forall x : \exists y : P(y) \vee Q(x, y)$ in der Skolemform $\forall x : \exists y : P(f_y(x)) \vee Q(x, f_y(x))$. Andererseits distributiert der Existenzquantor über Disjunktionen, und F ist deshalb äquivalent zu $\forall x : (\exists y : P(y)) \vee (\exists y : Q(x, y))$. Weiterhin, da x nicht frei in $\exists y : P(y)$ vorkommt, ist diese Formel mit (8) äquivalent zu $(\exists y : P(y)) \vee (\forall x : \exists y : Q(x, y))$. Mit Umbenennung von y durch z in der zweiten Teilformel erhält man schließlich die Pränexform $\exists y : \forall x : \exists z : P(y) \vee Q(x, z)$ und deren Skolemi-

sierung $\forall x : P(f_y) \vee Q(x, f_z(x))$. Man beachte, dass hier an die Stelle des ersten Vorkommens der Skolemfunktion $f_y(x)$ die „einfachere“ Skolemkonstante f_y getreten ist.

Siehe [46] für Techniken die diese Probleme vermeiden.

5.5 Das DPLL Verfahren für die Aussagenlogik

In diesem Abschnitt betrachten wir ein Beweisverfahren, das sich besonders gut für die automatische Beweisfindung in der Aussagenlogik eignet. Das nach seinen Entwicklern benannte *Davis-Putnam-Logemann-Loveland* Verfahren, oder kurz *DPLL*, wurde bereits Anfang der 1960er Jahre entwickelt. Es ist im Wesentlichen ein backtracking-basierter Suchalgorithmus, der die Erfüllbarkeit einer aussagenlogischen Formel in konjunktiver Normalform (KNF, Abschnitt 5.4) entscheidet.⁸ Mehr noch, falls die Formel unerfüllbar ist, liefert DPLL (mit geeigneten Modifikationen) einen Beweis in Form einer speziellen Resolutionswiderlegung, und falls sie erfüllbar ist ein Modell.

Wir betrachten das Verfahren in seiner einfachsten Version. Der folgende Algorithmus DPLL arbeitet mit aussagenlogischen Klauseln als *Mengen* von Literalen. Ein Literal ist also eine aussagenlogische Variable A oder deren Negation $\neg A$, und eine Klausel steht für die Disjunktion ihrer Literale. Der Eingabeparameter M ist eine (endliche) Menge solcher Klauseln. Das Ergebnis von $\text{DPLL}(M)$ ist **true**, falls M erfüllbar ist, andernfalls **false**.

```

1: procedure DPLL( $M$ )
2: while  $M$  enthält eine Klausel der Form  $\{L\}$  do
3:    $M \leftarrow \text{simplify}(M, L)$ 
4:   if  $M = \{\}$  then
5:     return true
6:   if  $\{\} \in M$  then
7:     return false
8:    $L \leftarrow \text{choose-literal}(M)$ 
9:   if DPLL(simplify( $M, L$ )) then
10:    return true
11:   else
12:     return DPLL(simplify( $M, \overline{L}$ ))

```

Die Hilfsprozedur choose-literal in Zeile 8 liefert ein in Φ vorkommendes Literal, welches beliebig gewählt werden kann. Die Hilfsprozedur simplify ist wie folgt definiert:

```

1: procedure simplify( $M, L$ )
2:  $M' \leftarrow M \setminus \{C \in M \mid L \in C\}$  Lösche alle Klauseln die  $L$  enthalten
3:  $M'' \leftarrow \{C \setminus \{\overline{L}\} \mid C \in M'\}$  Lösche  $\overline{L}$  von den verbleibenden Klauseln
4: return  $M''$ 

```

⁸ Interessanterweise wurde DPLL ursprünglich für die Prädikatenlogik erster Stufe entwickelt [18]. Die Beweissuche wird dabei durch exhaustive Instantiierung auf aussagenlogische Erfüllbarkeitsprobleme reduziert, ähnlich der weiter unten besprochenen „British-Museum Prozedur“. Interessanterweise kommt in [18] ein – aus heutiger Sicht – *Resolutionsverfahren* für die Aussagenlogik zum Einsatz, und erst die späteren Arbeit [17] führt das hier besprochene Verfahren ein. Heutzutage ist mit „DPLL Verfahren“ üblicherweise nur letzteres gemeint.

Die DPLL Prozedur ist leicht zu verstehen. Eine Schlüsselrolle spielen dabei Klauseln der Form $\{L\}$, also *Einerklauseln*, welche nur ein einziges Literal enthalten. Trivialerweise nämlich ist jede Klauselmenge M , die eine Einerklausel $\{L\}$ enthält genau dann erfüllbar, wenn es ein Modell I für M gibt mit $I \models L$. Dieser Sachverhalt rechtfertigt es, den Wahrheitswert des Literals L mit „wahr“ zu fixieren und M entsprechend zu vereinfachen. Genauer gesagt kann jede Klausel C mit $L \in C$ gelöscht werden, und jede Klausel C mit $\overline{L} \in C$ wird zu $C \setminus \overline{L}$. (Dass diese Vereinfachungen die Erfüllbarkeit erhalten, folgt unmittelbar aus der Semantik der KNF als Konjunktionen von Disjunktionen von Literalen.) Genau dies leistet der Aufruf $\text{simplify}(M, L)$ in Zeile 3.

Man beachte, dass $\text{simplify}(M, L)$ möglicherweise neue Einerklauseln liefert. Diese, und bereits vorhandene Einerklauseln werden alle durch die Schleife in Zeile 2/3 eliminiert. Offensichtlich enthält $\text{simplify}(M, L)$ keine Literale L oder \overline{L} mehr. Da jede Klauselmenge nur endlich viele Literale enthält, und jeder Aufruf von simplify alle Vorkommen von (mindestens) einem Literal eliminiert, ist sichergestellt, dass die Schleife terminiert.

Der DPLL Algorithmus versucht also zunächst M so lange wie möglich zu Simplifizieren, in dem die durch Einerklauseln forcierten Wahrheitswerte propagiert werden. Dieser Vorgang erfordert keinerlei Suche und wird auch als „unit propagation“ bezeichnet. Die anschließenden Tests in den Zeilen 4–7 können bereits das Ergebnis des aktuellen Aufrufs liefern, nämlich dann, wenn M leer (und damit trivialerweise erfüllbar) ist, oder wenn M die leere Klausel enthält (und damit trivialerweise unerfüllbar ist).

Andernfalls ist die Erfüllbarkeit von M zu diesem Zeitpunkt noch unbestimmt und wird nun durch rekursive Suche gelöst. Dazu wird in den Zeilen 9–12 eine Fallunterscheidung bezüglich der möglichen Wahrheitswerte eines beliebigen in M vorkommenden Literalen L vorgenommen. Der Aufruf $\text{DPLL}(\text{simplify}(M, L))$ in Zeile 9 untersucht implizit die Klauselmenge $M \cup \{L\}$ auf Erfüllbarkeit, ähnlich in Zeile 12 für $M \cup \{\overline{L}\}$.

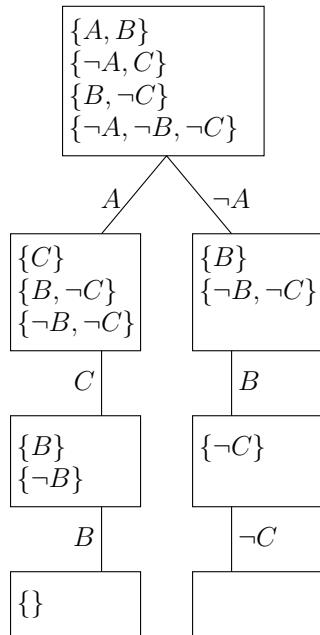
Man sieht leicht, dass M erfüllbar ist, genau dann wenn $M \cup \{L\}$ oder $M \cup \{\overline{L}\}$ erfüllbar ist, was sofort die Korrektheit des Algorithmus liefert. Die Termination wird mit dem gleichen Argument wie für die Schleife in Zeile 2/3 begründet.

Siehe Abbildung 5.1 für ein Beispiel.

Verbesserungen

Der DPLL Algorithmus benötigt im schlechtesten Fall exponentielle Laufzeit bezüglich der Größe der gegebenen Klauselmenge. Um dieses Problem zu mildern, wurden zahlreiche Verbesserung entwickelt, die zwar nicht die Komplexität von DPLL verbessern – das ist auch nicht zu erwarten – aber dennoch in der Praxis zu sehr schnellen Beweisen geführt haben:

Backjumping. Diese Verbesserung zielt darauf ab, unnötige Fallunterscheidungen zu vermeiden: angenommen, ein DPLL Lauf führt in Zeile 9 eine Fallunterscheidung bzgl. eines Literal L mit dem Ergebnis **false** durch. Nun ist es möglich, dass dieses Ergebnis nicht davon abhängt, dass L als „wahr“ angenommen wurde, d.h. mit „falsch“ wäre das Ergebnis ebenfalls **false**. Offensichtlich ist der entsprechende rekursive Aufruf in Zeile 12 dann unnötig. Um solche Fälle zu entdecken, werden die Abhängigkeiten der abgeleiteten Klauseln von den Fallunterscheidungsliteralen protokolliert. Der Fall tritt ein, falls eine leere Klausel abgeleitet wird, die in diesem Sinn unabhängig von L ist.



Die Abbildung links veranschaulicht den Lauf des DPLL Algorithmus in graphischer Form angewendet auf die in der Wurzel angegebene Klauselmenge M . Die mit A benannte Verzweigung repräsentiert den rekursiven Aufruf $\text{DPLL}(\text{simplify}(M, A))$, wobei $\text{simplify}(M, A)$ die im linken Knoten angegebene Klauselmenge ist. Die Nachfolgeknoten repräsentieren die anschließenden Simplifizierungsschritte mit C und mit B . Die abgeleitete leere Klauselmenge $\{\}$ führt zurück zum Fall $\neg A$ im rechten Zweig. Anschließende Simplifizierung ergibt die leere Klauselmenge und zeigt die Erfüllbarkeit von M an. Die durchgeföhrten Simplifizierungsschritte, mit den Literalen $\neg A$, B und C bestimmen das entsprechende Modell von M .

Abbildung 5.1: DPLL Algorithmus Beispiel.

Lemma Learning. Der DPLL Algorithmus arbeitet analytisch: alle betrachteten Klauseln sind Teilklauseln der gegebenen Klauselmenge. Die Idee der „Lemma Learning“ Verbesserung ist, durch Hinzufügen *neuer* Klauseln wiederkehrende Situationen im Suchraum zu erkennen und wiederholte identische Exploration zu vermeiden [53]. Eine neue Klausel entsteht dabei durch Regression einer leeren Klausel mit Hilfe der oben genannten Abhängigkeiten zu den relevanten Fallunterscheidungsliteralen. Falls zum Beispiel ein Pfad (im Sinne der baumartigen Ableitung in Abbildung 5.1) die Fallunterscheidungsliteralen A , $\neg D$, und E enthält und nur diese Literale relevant sind, um die leere Klausel abzuleiten, deutet das darauf hin, dass es kein Modell der gegebenen Klauselmenge M gibt, in dem dieses Literale wahr sind. Daraus folgt, dass die Klausel $\{\neg A, D, \neg E\}$ eine logische Konsequenz aus M ist und deshalb, eben als Lemma, zu M hinzugefügt werden kann. Falls nun, zum Beispiel, ein (anderer) Pfad die Literale E und $\neg D$ enthält, kann sofort die Einerklausel $\{\neg A\}$ ohne weitere Suche abgeleitet werden. Des weiteren wird durch Lemma Learning die oben genannte Backjumping-Verbesserung gleich miterledigt.

Restarts. Damit ist gemeint, dass ein DPLL Lauf abgebrochen und neu gestartet wird, falls innerhalb einer gesetzten Ressourcenschanke (zum Beispiel Zeit) kein Ergebnis kommt. Das ergibt natürlich nur dann Sinn, wenn derselbe Lauf nicht wiederholt wird. Dafür können Zufallselemente beim Auswahl des nächsten Literalen zum Tragen kommen (Zeile 8). Auch Lemmas erzeugen oft einen völlig unterschiedlichen Suchraum.

Implementierungstechniken. Seit etwa Anfang der 1990er Jahre wird intensiv untersucht, DPLL auch auf großen Klauselmengen „schnell“ zu machen. Eine wichtige Rolle kommt

dabei der Auswahl der Fallunterscheidungsliterale in Zeile 8 zu. Das Endergebnis ist zwar unabhängig davon, geschickt gewählte Literale können aber zu deutlich kürzeren Läufen führen. So kann zum Beispiel ein Literal gewählt werden, das die anschließende Simplifizierung maximiert [37]. Jedoch ist das Finden eines solchen Literals relativ kostspielig. Weniger aufwändige Heuristiken basieren zum Beispiel auf der relativen Häufigkeit von Literalvorkommen.

Andere Verbesserungen betreffen die Datenstrukturen zur Klauselrepräsentation. So wurden zum Beispiel Indizierungsmechanismen vorgeschlagen, um die benötigten Klauseln mit geringem Aufwand zu finden [59]. In [41] wurde (unter anderem!) die wichtige „two watched literal“ Technik eingeführt, um die Simplifizierungsoperation effizient zu implementieren. (Genauer gesagt setzt die Verbesserung beim Backtracking ein, welches – naiv implementiert – den Effekt der Simplifizierungsschritte auf die betroffenen Klauseln rückgängig machen muss.)

Mit diesen und weiteren Verbesserungen konnte die Leistungsfähigkeit implementierter aussagenlogischer Beweisverfahren drastisch verbessert werden: DPLL-basierte System konnten Anfang der 1990er Jahre Probleme mit ca. 100 Variablen und 200 Klauseln behandeln, moderne Systeme, sogenannte *Conflict Driven Clause Learning SAT-Solver*, schaffen Probleme bis zu 20 Millionen Klauseln mit 1 Million Variablen!

Schließlich sei noch erwähnt, dass andere, nicht auf DPLL basierende Verfahren für die Aussagenlogik entwickelt wurden. *Stochastische Systeme* sind oft gut zur Modellfindung geeignet, können Unerfüllbarkeit aber nur mit einer gewissen (sehr hohen) Wahrscheinlichkeit feststellen. *Binary Decision Diagrams* repräsentieren simultan die Modelle und Gegenmodelle einer Klauselmenge.⁹ Zusammen genommen stehen damit heute eine Reihe von Methoden zur Verfügung, um erfolgreich Anwendungsgebiete wie Hardware/Software Verifikation (Bounded Modell Checking), Planen, und Bioinformatik zu unterstützen.

Siehe [10] für einen umfassenden Überblick.

5.6 Aussagenlogische Resolution

Wir wenden uns nun dem wichtigsten Kalkül des automatischen Schließens zu, dem Resolutionskalkül [49]. Der Resolutionskalkül arbeitet auf Formeln besonders einfacher Struktur, Klauselmengen, verfügt (in der ursprünglichen Fassung) nur über eine einzige Inferenzregel und benötigt kein einziges Axiom. Darüberhinaus ist er *nicht* deduktionsvollständig, sondern nur widerlegungsvollständig, was den Suchraum von vorne herein gegenüber z.B. dem Hilbert Kalkül einschränkt. Die wichtigste Neuerung aber war die Verwendung von *Unifikation* in der Inferenzregel. Damit ist es möglich, Inferenzen auf allgemeinere Ebene zu führen, die z.B. unendlich vielen Inferenzen im Hilbert Kalkül entsprechen. Keine schlechten Voraussetzungen also für die „Mechanisierung der Logik“. Dennoch hat sich herausgestellt, dass das ursprüngliche Resolutionsverfahren oft hoffnungslos ineffizient ist. Im Laufe der Zeit wurden deshalb zahlreiche Effizienz steigernde Maßnahmen entwickelt. Eine davon ist die Variante der sogenannten „A-geordneten Resolution“, welche (als Spezialfall) von den meisten aktuellen Resolution-basierten Beweisern implementiert ist, siehe Abschnitt 5.6.2.

⁹ Ein Gegenmodell ist eine Interpretation für eine Formel die die Formel falsch macht.

Wir beschränken uns in diesem Abschnitt auf den Spezialfall der Aussagenlogik. Das reicht bereits aus, um sich ein Bild der Funktionsweise des Resolutionskalküls auch für Prädikatenlogik zu machen. Die Verallgemeinerung zur Prädikatenlogik erfolgt dann in Abschnitt 5.7.2.

5.6.1 Ein einfacher Resolutionskalkül

Zunächst ein paar Konventionen. Im Folgenden stehen die Bezeichner C und D für Klauseln, A und B für Atome oder positive Literale, und L steht für ein Literal. Deshalb bezeichnet zum Beispiel $C \vee A$ eine Klausel, die sich aus einer Disjunktion von Literalen C und einem positiven Literal A zusammensetzt. Mehr noch, unter Ausnutzung von Assoziativität und Kommutativität von \vee (vgl. „Schritt 4“ in Abschnitt 5.4) sollen mit der Notation wie z.B. $C \vee L \vee L$ alle Klauseln gemeint sein, die unter doppelter Verwendung eines Literalen L und der Literale in C gebildet werden können. Klauseln können also auch als Multimengen von Literalen aufgefasst werden.

Wie erwähnt benötigt der Resolutionskalkül keine Axiome. Unter Verwendung des in Abschnitt 5.2 eingeführten allgemeinen Kalkülbegriffs reicht es deshalb aus, die Inferenzregeln und den Ableitungsbegriff zu definieren.

Inferenzregeln

$$\text{Res} \quad \frac{C \vee A \quad \neg A \vee D}{C \vee D} \qquad \text{Fak} \quad \frac{C \vee A \vee A}{C \vee A}$$

Ableitungsbegriff

Wie im Hilbert Kalkül definiert man auch hier: Eine (*Resolutions*)ableitung einer Klausel C aus einer Menge M von Klauseln (den Hypothesen) ist eine Folge von Klauseln C_1, \dots, C_n wobei $n \geq 0$, $C_n = C$ und für alle $i = 1, \dots, n$ gilt:

1. C_i ist eine Hypothese aus M , oder
2. C_i ist die Konklusion einer Res- oder Fak-Inferenz mit Prämisse(n) aus $\{C_1, \dots, C_{i-1}\}$. Die Prämissen heißen auch *Elternklauseln*.

Falls $C = \square$ ist, so heißt die Ableitung auch (*Resolutions*)widerlegung.

Zum Beispiel gibt es für die Klauselmenge $M = \{\neg A \vee \neg A \vee B, A \vee B, \neg C \vee \neg B, C\}$ die folgende Resolutionswiderlegung:

1. $\neg A \vee \neg A \vee B$ (aus M)
2. $A \vee B$ (aus M)
3. $\neg C \vee \neg B$ (aus M)
4. C (aus M)
5. $\neg A \vee B \vee B$ (Res 2. und 1.)
6. $\neg A \vee B$ (Fak 5)
7. $B \vee B$ (Res 2 und 6)
8. B (Fak 7)
9. $\neg C$ (Res 8 und 3)
10. \perp (Res 4 und 9)

Zur Korrektheit des Resolutionskalküls

Ein Kalkül sollte zumindest *korrekt* sein. Auf Resolution bezogen heißt das: falls es eine Resolutionswiderlegung einer Klauselmenge M gibt, dann ist M unerfüllbar. Wir wollen uns vergewissern, dass dies der Fall ist.

Man benötigt im Wesentlichen die folgenden Eigenschaften. Sie besagen dass Resolventen und Faktoren logische Konsequenzen ihrer Elternklauseln sind:

1. $\{C \vee A, \neg A \vee D\} \models C \vee D$
2. $\{C \vee A \vee A\} \models C \vee A$

Der Beweis der zweiten Eigenschaft ist trivial, und der Beweis der ersten Eigenschaft geht so: Sei I eine beliebige Interpretation mit $I \models C \vee A$ and $I \models \neg A \vee D$. Wir müssen $I \models C \vee D$ zeigen. Dazu werden zwei Fälle unterschieden: Im ersten Fall ist $A^I = \mathbf{w}$. Also ist $\neg A^I = \mathbf{f}$. Mit $I \models \neg A \vee D$ folgt $D^I = \mathbf{w}$ und damit trivialerweise $I \models C \vee D$. Im zweiten, komplementären Fall gilt $A^I = \mathbf{f}$. Mit $I \models C \vee A$ folgt $C^I = \mathbf{w}$ und damit ebenfalls trivialerweise $I \models C \vee D$.

Unter Ausnutzung dieser Ergebnisse erhält man recht leicht, dass jede Klausel einer Ableitung aus M eine logische Konsequenz aus M ist. Falls nun die Ableitung die leere Klausel enthält (man erinnere sich dass die leere Klausel von keiner Interpretation erfüllt wird), dann muss bereits M unerfüllbar sein. Deshalb ist der Resolutionskalkül korrekt.

Schwache und starke Widerlegungsvollständigkeit

Diese Begriffe beschreiben eine prinzipielle Eigenschaft von Ableitungen widerlegungsvollständiger Kalküle. Gemeint ist das Folgende, am Beispiel des Resolutionskalküls. Gegeben sei eine unerfüllbare Klauselmenge M und eine beliebige Resolutionsableitung C_1, \dots, C_m . Die Frage ist, ob es immer eine Fortsetzung zu einer Widerlegung $C_1, \dots, C_m, C_{m+1}, \dots, (C_n = \square)$ gibt. Da dies der Fall ist, ist der Resolutionskalkül *stark* widerlegungsvollständig (oder *beweiskonfluent*). Andere widerlegungsvollständige Kalküle wie zum Beispiel Modellelimination oder der sogenannte „lineare Resolutionskalkül“ haben diese Eigenschaft nicht und werden deshalb *schwach* widerlegungsvollständig genannt.

Der Unterschied ist in der Praxis wichtig. Beweisprozeduren für stark widerlegungsvollständige Kalküle müssen niemals eine einmal begonnene Ableitung wieder zurückziehen; die Beweisfindung ist don't-care nichtdeterministisch. Offensichtlich ist dies eine vorteilhafte Eigenschaft. Man sollte allerdings bedenken, dass schwach vollständige Kalküle andere Vorteile bieten können. So arbeitet zum Beispiel der lineare Resolutionskalkül *ziel-orientiert*, d.h., jede Inferenz hängt mit dem zu beweisendem Theorem zusammen, eine Eigenschaft, die zum Beispiel die A-geordnete Resolution (siehe unten) nicht hat. Man kann deshalb nicht generell sagen, dass stark vollständige Kalküle immer vorzuziehen sind.

Der Resolutionskalkül ist stark widerlegungsvollständig. Wir erhalten dieses Ergebnis über den Begriff der „saturierten Menge“.

Definition 5.6.1 (Saturierte Klauselmenge). *Ein Klauselmengen M ist **saturiert** (unter Res und Fak), falls die Konklusion jeder Res oder Fak Inferenz mit Prämisse(n) aus M in M enthalten ist.*

Es stellt sich nun die Frage, wie man saturierte Klauselmengen erhält. Das geht so (etwas allgemeiner definiert):

Definition 5.6.2 (Deduktiver Abschluss). *Sei eine Kollektion von Inferenzregeln gegeben. Der deduktive Abschluss (unter Anwendung der Inferenzregeln) M^* einer Formelmenge M unter Anwendung der Inferenzregeln ist wie folgt definiert:*

$$\begin{aligned} M^0 &= M \\ M^{n+1} &= M^n \cup \{C \mid C \text{ ist Konkl. einer Inferenz mit Prämissen aus } M^n\} \\ M^* &= \bigcup_{n \geq 0} M^n \end{aligned}$$

Offensichtlich ist der deduktive Abschluss M^* von M unter Res und Fak saturiert. Weiter unten werden wir das folgende Theorem beweisen:

Theorem 5.6.1 (Korrektheit und Widerlegungsvollständigkeit). *Eine unter Res und Fak saturierte Klauselmenge M enthält die leere Klausel \square genau dann, wenn M unerfüllbar ist.*

Falls also eine ursprünglich gegebene Klauselmenge M unerfüllbar ist, folgt trivialerweise mit $M \subseteq M^*$, dass auch M^* unerfüllbar ist. Zusammen mit Theorem 5.6.1 folgt $\square \in M^*$. Mit diesen Ergebnissen folgt nun ohne größeren Aufwand die starke Widerlegungsvollständigkeit des Resolutionskalküls: gegeben sei eine Ableitung von M der Form C_1, \dots, C_m . Wir müssen zeigen, dass sie zu $C_1, \dots, C_m, C_{m+1}, \dots, (C_n = \square)$ fortgesetzt werden kann. Das ist aber einfach, man braucht nur aus der Berechnung des deduktiven Abschlusses, die fehlenden Klauseln $C_{m+1}, \dots, (C_n = \square)$ abzulesen, die ja alle aus M sind oder durch Res und Fak Inferenzen erhalten werden.

Beweisprozedur

Die Definition des deduktiven Abschlusses kann man auch so verstehen, dass keine mögliche Inferenz „unendlich lange“ hinausgezögert wird, außer die leere Klausel wird abgeleitet. Jede mögliche Inferenz muss also nach endlicher Zeit ausgeführt werden. In der Praxis verwendet man dazu die folgende Beweisprozedur.

```

1: procedure GivenClauseLoop( $M$ )
2: Usable  $\leftarrow M$ 
3: WorkedOff  $\leftarrow \emptyset$ 
4: while  $\square \notin \text{Usable}$  and  $\text{Usable} \neq \emptyset$  do
5:   Given  $\leftarrow \text{choose-clause}(\text{Usable})$ 
6:   Usable  $\leftarrow \text{Usable} \setminus \{\text{Given}\}$ 
7:   WorkedOff  $\leftarrow \text{WorkedOff} \cup \{\text{Given}\}$ 
8:   New  $\leftarrow \{D \mid \text{mit } C \in \text{WorkedOff} \text{ und Res } \frac{\text{Given} \quad C}{D}\} \cup \{D \mid \text{Fak } \frac{\text{Given}}{D}\}$ 
9:   Usable  $\leftarrow \text{Usable} \cup \text{New}$ 
10:  if  $\square \in \text{Usable}$  then
11:    return false
12:  else
13:    return true

```

Der Parameter M ist die gegebene Klauselmenge. Die GivenClauseLoop Prozedur arbeitet mit zwei Mengen von Klauseln: Usable, welches mit M initiiert wird, und WorkedOff. Die Prozedur nimmt nacheinander alle Klauseln aus Usable, kopiert sie nach WorkedOff, und führt alle

Inferenzen mit allen Klauseln aus WorkedOff durch. Die sich ergebenden Klauseln kommen nach Usable . Falls die Schleife terminiert und die leere Klausel abgeleitet wird, ist das Ergebnis **false** (für „*unerfüllbar*“). Andernfalls ist Usable leer, und da alle Inferenzen ausgeführt wurden, ist das Ergebnis **true** (für „*erfüllbar*“).

Der Hauptvorteil der `GivenClauseLoop` Prozedur gegenüber einer naiven Umsetzung von Definition 5.6.2 ergibt sich aus der folgenden (leicht einzusehenden) Invariante:

Die Menge WorkedOff ist bezüglich aller **Fak** Inferenzen und aller **Res** Inferenzen mit allen Klauseln aus WorkedOff abgeschlossen.

Die wiederholte Berechnung derselben Inferenzen mit Klauseln aus WorkedOff wird also vermieden.

Was für die Praxis noch fehlt, sind Mechanismen zum Löschen nicht benötigter Klauseln. Kriterien dafür werden weiter unten angegeben (siehe Abschnitt 5.6.3). Ein entsprechender Filter kann in Zeile 8 eingebaut werden, um solche Klauseln gar nicht erst in New aufzunehmen.

Die Hauptanwendung der `GivenClauseLoop` Prozedur für die prädikatenlogische Resolution. Sie kann unverändert dafür übernommen werden, freilich müssen die prädikatenlogischen Versionen der Inferenzregeln eingesetzt werden.

5.6.2 A-geordnete Resolution

Die Richtung von rechts nach links in Theorem 5.6.1 – „Falls M saturiert und unerfüllbar ist dann $\square \in M$ “ – ist nicht ganz so leicht zu beweisen wie die Korrektheit.¹⁰ Eine Beschäftigung mit diesem Beweis lohnt sich dennoch, liefert er doch die Grundlage zum Verständnis des Resolutionskalküls. Wir betrachten eine verfeinerte Variante, die *A-geordnete Resolution*, und führen den Beweis dafür. Wir beschränken uns hier auf die Aussagenlogik, das reicht, um die Grundidee zu vermitteln. Das volle Potential der zu Grunde liegenden Beweistechnik wird allerdings erst für die prädikatenlogische Variante mit eingebauten Inferenzregeln für Gleichheit benötigt. Siehe [2] für einen ausführlicheren Überblick.

Der Name *A-geordnete Resolution* ergibt sich daraus, dass Ordnungen auf Atomen zur Einschränkung von **Fak** und **Res** Inferenzen verwendet werden.

A-Ordnungen

Sei eine endliche Klauselmenge M gegeben und $\mathcal{A}_M = \{A_1, \dots, A_n\}$ die Menge aller in M vorkommenden aussagenlogischen Variablen. Sei \succ eine totale und strikte Ordnung auf \mathcal{A}_M .¹¹ Es gibt viele solcher Ordnungen, z.B. die lexikographische Ordnung auf den Namen der Atome.

Zu einer gegebenen Klauselmenge M kann die Ordnung \succ auf den darin vorkommenden Atomen \mathcal{A}_M beliebig gewählt werden. Diese Ordnung wird nun zu einer Ordnung auf allen Literalen und dann zu einer Ordnung auf allen Klauseln erweitert, die sich mit den Atomen aus \mathcal{A}_M bilden lassen.

¹⁰ Korrespondierend zur Einsicht, dass es schwerer ist, einen Beweis selbst zu finden als einen gegebenen Beweis nachzuvollziehen.

¹¹ Ein totale strikte Ordnung \succ auf einer Menge ist eine irreflexive und transitive Relation, so dass $x \succ y$ oder $y \succ x$ für alle Elemente der Menge gilt. Die übliche größer-als Relation $>$ auf den ganzen Zahlen ist solch eine Ordnung.

Literalordnung. Die Ordnung \succ wird auf eine (totale) Ordnung auf Literale \succ_L (eindeutig) durch die folgenden Regeln fortgesetzt:

1. Falls $A \succ B$ dann $A \succ_L B$.
2. Falls $A \succ B$ dann $\neg A \succ_L \neg B$.
3. $\neg A \succ_L A$.

Klauselordnung. Die Ordnung \succ_L wird auf eine (totale) Ordnung auf Klauseln \succ_C (eindeutig) auf Klauseln fortgesetzt, indem man eine Klausel als Multimenge ihrer Literale auffasst, und \succ_C als die Erweiterung von \succ_L auf Multimengen definiert.

Man kann sich die Erweiterung von \succ zu \succ_L so vorstellen, dass zu jedem Atom A das Literal $\neg A$ „direkt oberhalb“ der Ordnung \succ eingeschoben wird. Aus z.B. $A \succ B \succ C$ erhält man $\neg A \succ A \succ \neg B \succ B \succ \neg C \succ C$.

Um sich die Ordnung auf Klauseln plausibel zu machen, erinnere man sich, dass man aus einem Multiset ein kleineres Multiset erhält, indem man ein Element durch beliebig viele kleinere Elemente ersetzt. Mit der Ordnung von eben erhält man deshalb z.B.

$$\underline{A} \vee A \vee B \succ_C A \vee \underline{B} \succ_C A \vee \neg C \vee C .$$

Die unterstrichenen Literale geben hierbei die ersetzenen Literale an.

Im Folgenden schreiben wir einfach \succ statt \succ_L und auch statt \succ_C .

Man beachte, dass die Ordnung \succ auf Klauseln zwar wohlfundiert, aber in der Regel nicht isomorph zu den natürlichen Zahlen ist. So ist zum Beispiel $A \succ B_1 \vee \dots \vee B_n$ für alle $i \geq 1$.

Man beachte weiterhin, dass eine Klausel mehr als ein maximales Literal haben kann. Wegen der Totalität der Ordnung müssen diese Literale aber alle gleich sein. Wir können deshalb von „dem“ maximalen Literal sprechen. In der Klausel $A \vee A \vee B$ oben ist das der Fall, und das maximale Literal ist A .

Kalkül

Eine erste Anwendung finden Ordnungen bei der Definition der Inferenzregeln der A-geordneten Resolution:

$$\text{A-Res } \frac{C \vee A \quad \neg A \vee D}{C \vee D} \quad \text{falls } \begin{cases} A \text{ ist maximal in } C \vee A, \text{ und} \\ \neg A \text{ ist maximal in } \neg A \vee D \end{cases}$$

$$\text{A-Fak } \frac{C \vee A \vee A}{C \vee A} \quad \text{falls } A \text{ ist maximal in } C \vee A \vee A$$

Der Ableitungsbegriff ist derselbe wie in Abschnitt 5.6.1.

Man beachte, dass sowohl die A-Res als auch die A-Fak Inferenzregel im Vergleich zu ihren nicht-geordneten Versionen Res und Fak in ihrer Anwendbarkeit eingeschränkt sind. Das ist auch beabsichtigt, um weniger Klauseln ableiten zu müssen. Der deduktive Abschluss einer Klauselmenge M unter A-Res und A-Fak wird somit immer eine Teilmenge des deduktiven Abschlusses vom M unter Res und Fak sein.

Analog zu Theorem 5.6.1 erhalten wir für die A-geordnete Resolution:

Theorem 5.6.2. Eine unter A-Res und A-Fak saturierte Klauselmenge M enthält die leere Klausel \square genau dann, wenn M unerfüllbar ist.

Beweis. Unter Ausnutzung von Theorem 5.6.1 ist die Korrektheitsaussage trivial. Zum Beweis der Vollständigkeitsaussage nehmen wir eine saturierte Klauselmenge M an, die die leere Klausel \square nicht enthält. Man beachte, dass M unendlich groß sein kann, selbst wenn man sie als deduktiven Abschluss einer endlichen Ausgangsmenge bildet. Denn, aus z.B. den Klauseln $A \vee B$ und $\neg A \vee A \vee B$ kann man mit A-Res, falls $A > B$ ist, die Klausel $A \vee B \vee B$, und damit dann $A \vee B \vee B \vee B$ erhalten, und so weiter. Das ist aber kein Problem, und der Beweis funktioniert selbst mit einem unendlichen Vorrat von aussagenlogischen Variablen (diese Allgemeinheit wird für den prädikatenlogischen Fall auch gebraucht).

Wir müssen zeigen dass M erfüllbar ist. Dies erfolgt konstruktiv durch Angabe eines Modells I für M .

Zunächst zu einer Konvention. Es ist nützlich, aussagenlogische Interpretationen als Mengen von Atomen darzustellen. Man führt in der Menge einfach genau die Atome auf die „wahr“ sein sollen. Formal exakt, im Sinne der Eingangs in Abschnitt 5.3 geforderten Interpretation setzt man $A^I = \{\emptyset\}$, falls $A \in I$, und $A^I = \{\}$ falls $A \notin I$.

Damit nun zur Definition von I . Sie wird bestimmt durch wohl-fundierte Induktion über die Ordnung $>$ auf Klauseln, wie folgt. Sei D eine Klausel. Wir benötigen zwei zu D assoziierte Mengen ϵ_D und I_D . Als Induktionsannahme sei ϵ_E bereits definiert für alle Klauseln E mit $D > E$. Dann ist

$$\begin{aligned}\epsilon_D &= \begin{cases} \{A\} & \text{falls } D \text{ von der Form } C \vee A \text{ ist,} \\ & \text{wobei (i) } A > C, \text{ (ii) } I_D \not\models D, \text{ und (iii) } I_D \cup \{A\} \not\models C. \\ \emptyset & \text{sonst} \end{cases} \\ I_D &= \bigcup_{D > E} \epsilon_E\end{aligned}$$

Schließlich setze $I = \bigcup_{D \in M} \epsilon_D$.

Wir sagen, dass die Klausel $D = C \vee A$ das Atom A produziert, falls die angegebene Bedingung zutrifft; die Klausel D heißt dann produktiv.

Zunächst ein paar Erläuterungen zu diesen Definitionen. Die Interpretation I ist als Modell für M gedacht. (Und sie ist es auch, falls M saturiert ist und die leere Klausel nicht enthält, was unten gezeigt wird.) Intuitiv wird I schrittweise konstruiert, durch Inspektion aller Klauseln in M , von den kleineren zu den größeren hin. Die Menge ϵ_D beschreibt dabei eine minimale Änderung der aktuellen Interpretation, um eine aktuell inspiizierte Klausel D zu erfüllen.

Etwas genauer: Nehmen wir also an, dass eine Klausel D an der Reihe ist und dass die Interpretation I_D für alle Klauseln die kleiner sind als D bereits definiert ist. Man beachte, dass in der Definition von ϵ_D nur der Fall betrachtet wird, dass D von der Form $C \vee A$ ist, und die angegebenen Bedingungen (i), (ii) und (iii) erfüllt sind. Das heißt, D ist falsch in der momentanen Interpretation I_D (Bedingung (ii)) und enthält ein maximales positives Literal A (Bedingung (i)). Die Klausel D wird natürlich wahr in der um ϵ_D erweiterten Interpretation $I_D \cup \{A\}$. Bedingung (iii) gibt an, dass die Restklausel C dabei falsch bleibt.

Als Beispiel betrachten wir die folgenden drei Klauselmengen. Die gewählte A-Ordnung ist $A > B > C$. Damit ergibt sich für die Klauseln $(1) > (2) > (3) > (4)$. Die maximalen Literale in den Klauseln sind unterstrichen, die produktiven Literale sind mit * markiert.

$$\begin{array}{ll} (1) & \underline{\neg A} \vee C \\ (2) & \underline{A} \vee \underline{A} \vee B \end{array}$$

$$\begin{array}{ll} (1) & \underline{\neg A} \vee C \\ (2) & \underline{A} \vee \underline{A} \vee B \\ (3) & \underline{A^*} \vee B \end{array}$$

$$\begin{array}{ll} (1) & \underline{\neg A} \vee C \\ (2) & \underline{A} \vee \underline{A} \vee B \\ (3) & \underline{A} \vee B \\ (4) & \underline{B^*} \vee C \end{array}$$

Zunächst zur linken Menge. Weder (1) noch (2) ist produktiv. Wir haben $I_{(2)} = \emptyset$. Die Bedingungen (i) und (ii) sind für (2) erfüllt, nicht aber (iii), deshalb ist (2) nicht produktiv. Die Klausel (3) in der mittleren Menge erhält man durch eine A-Fak Inferenz von (2). Die Klausel (3) ist produktiv, und wir erhalten $I = \{A\}$. Damit sind (2) und (3) erfüllt, nicht aber (1). Die mittlere Klauselmenge ist auch nicht saturiert, und es besteht kein Anlass anzunehmen, dass I alle Klauseln erfüllt. Die Klausel (4) in der rechten Menge erhält man durch eine A-Res Inferenz von (1) und (3). Die rechte Klauselmenge ist saturiert, und die induzierte Interpretation $I = \{B\}$ ist ein Modell. Man beachte, dass (3) durch Hinzunahme von (4) nicht mehr produktiv ist; die Bedingung (ii) ist verletzt, da $I_{(3)} = \{B\}$ aber $I_{(3)} \models (3)$.

Sei $D = C \vee A$ produktiv. Da kein Atom aus $I \setminus (I_D \cup \epsilon_D)$ in D vorkommen kann, diese sind alle größer als A , welches selbst maximal in D ist, bleibt D wahr in I . Aus dem gleichen Grund bleibt die Restklausel C falsch in I . Dieser Sachverhalt wird weiter unten benötigt.

Um produktive Klauseln brauchen wir uns also im Beweis nicht weiter kümmern. Für nichtproduktive Klauseln erfolgt der Beweis durch Widerspruch. Angenommen es gibt eine nichtproduktive Klausel $D \in M$ mit $I \not\models D$. Wegen Wohlfundiertheit und Totalität der Klauselordnung \succ dürfen wir annehmen, dass D bereits als die kleinste Klausel mit dieser Eigenschaft gewählt worden ist, d.h., es gibt keine (nicht produktive) Klausel $C \in M$ mit $D \succ C$ und $I \not\models C$. Man könnte D als „kleinstes Gegenbeispiel“ bezeichnen.

Als nächstes treffen wir eine Fallunterscheidung bezüglich des maximalen Literals in D . Da nach Voraussetzung $\square \notin M$ gilt, muss D mindestens ein Literal enthalten. In beiden Fällen werden wir im Widerspruch zur Minimalität von D ein „kleineres Gegenbeispiel“ erhalten.

1. Fall: Das maximale Literal in D ist negativ

Die Klausel D hat die Form $\neg A \vee C$, wobei $\neg A$ das maximale Literal in $\neg A \vee C$ ist. (Man beachte, dass $\neg A$ durchaus mehrfach in D vorkommen kann, das spielt keine Rolle.) Aus $I \not\models \neg A \vee C$ folgt $I \not\models C$ und $I \models A$. Also muss es eine Klausel der Form $C' \vee A$ in M geben die A produziert (andernfalls wäre $I \models A$ nicht möglich). Man sieht leicht, dass alle Bedingungen zur Anwendung der A-geordneten Resolutionsregel erfüllt sind. Das heißt, es gibt eine Inferenz

$$\text{A-Res} \quad \frac{C' \vee A \quad \neg A \vee C}{C' \vee C}$$

Da $C' \vee A$ das Atom A produziert folgt $I \not\models C'$, wie oben allgemein für produktive Klauseln ausgeführt wurde. Zusammen mit $I \not\models C$ also $I \not\models C' \vee C$. Da M saturiert, also insbesondere unter Anwendung von A-Res abgeschlossen ist, folgt $C' \vee C \in M$.

Aus $\neg A > A$ (das folgt aus der Definition von $>$) und $A > C'$ (weil $C' \vee A$ das Atom A produziert) folgt mit Transitivität $\neg A > C'$. Wiederum aus den Eigenschaften der Ordnung folgt $\neg A \vee C > C' \vee C$.

Die Menge M enthält also eine Klausel, $C' \vee C$, die kleiner ist als D und nicht von I erfüllt ist. Das steht im direkten Widerspruch zur Minimalität von D .

2. Fall: Das maximale Literal in D ist positiv

Die Klausel D hat die Form $C \vee A$, wobei A das maximale Literal in $C \vee A$ ist. Da D nach Annahme nicht produktiv ist, folgt (i) $A \not> C$, (ii) $I_D \models D$ oder (iii) $I_D \cup \{A\} \models C$

Im Fall (i), $A \not> C$, folgt aus der Maximalität von A in $C \vee A$ und der Totalität von $>$, dass C das Literal A enthalten muss, D ist also von der Form $C' \vee A \vee A$. Dann aber gibt es eine Inferenz

$$\text{A-Fak } \frac{C' \vee A \vee A}{C' \vee A}$$

Da M saturiert, also insbesondere unter Anwendung von A-Fak abgeschlossen ist, muss $C' \vee A \in M$ sein. Aus $I \not\models C' \vee A \vee A$ folgt trivialerweise $I \not\models C' \vee A$. Andererseits gilt $C \vee A = C' \vee A \vee A > C' \vee A$. Wie im Fall 1 enthält die Menge M also eine Klausel, hier $C' \vee A$, die kleiner ist als D und nicht von I erfüllt ist. Wir erhalten denselben Widerspruch zur Minimalität von D .

Da Fall (i) bereits ausgeschlossen ist, dürfen wir von nun an $A > C$ annehmen. Daraus folgt, dass C das Literal A nicht enthalten kann. Und $\neg A$ kann auch nicht in C vorkommen, sonst wäre A nicht maximal in D . Deshalb kann kein Atom aus $I \setminus I_D$ in C vorkommen, diese sind alle größer oder gleich A . Daraus folgt $I \models C$ gdw. $I_D \models C$ gdw. $I_D \cup \{A\} \models C$ (da ja weder A noch $\neg A$ in C vorkommt).

Im Fall (iii) folgt $I \models C$ und damit $I \models D$, im direkten Widerspruch zur Annahme $I \models D$. Fall (iii) ist somit ausgeschlossen.

Im Fall (ii) haben wir $I_D \models C$ oder $I_D \models A$. Falls $I_D \models C$ erhalten wir wieder $I \models C$ und es geht weiter wie in Fall (iii). Falls $I_D \models A$ folgt trivialerweise $A \in I_D$ und $A \in I$, somit $I \models D$, wieder im Widerspruch zur Annahme $I \models D$.

Damit sind alle Fälle behandelt. Da jeder Fall zum Widerspruch führt, ist die Annahme $I \not\models D$ widerlegt und es folgt $I \models D$, wie gewünscht. \square

Der Vollständigkeitsbeweis liefert den Schlüssel zum Verständnis des Resolutionskalkül. Im Zentrum steht die im Beweis definierte Interpretation I . Diese Interpretation existiert immer, unabhängig davon, ob die gegebene Klauselmenge M erfüllbar ist oder nicht. Falls M erfüllbar ist, kann I als intendiertes Ergebnis angesehen werden. Falls M unerfüllbar ist, gibt es zu jedem Gegenbeispiel (d.h., zu jeder in I falsifizierten Klausel) außer der leeren Klausel immer ein kleineres Gegenbeispiel, das durch die Inferenzregeln A-Res und A-Fak bestimmt werden kann. Der „Zweck“ der Inferenzregeln in diesem Sinne ist es demnach, Gegenbeispiele zu reduzieren.

5.6.3 Verfeinerungen des Resolutionsverfahrens

Wie gesagt hat sich der Resolutionskalkül in der ursprünglichen Fassung von Robinson als nicht effizient genug für die Praxis herausgestellt. Deshalb wurden zahlreiche Verfeinerungen zur Suchraumreduktion entwickelt. Letztendlich läuft es immer darauf hinaus, den deduktiven Abschluss zu verkleinern. Generell kann man zwischen drei Arten von Verfeinerungen unterscheiden:

Restriktionsverfeinerungen: Nur ein Teil aller möglichen Inferenzen des Resolutionskalküls aus Abschnitt 5.6.1 ist gestattet.

Redundanzverfeinerungen: Gewisse Klauseln werden als „redundant“ gekennzeichnet und können gelöscht werden, d.h. sie brauchen für Inferenzen nicht herangezogen werden.

Strukturelle Verfeinerungen: An die Ableitungen werden gewisse Anforderungen gestellt.

Eine Verfeinerung der ersten Art haben wir bereits gesehen: die A-geordnete Resolution.

Die wichtigste Redundanzverfeinerung ist „Subsumption“: eine Klausel ist redundant, falls die momentane Klauselmenge eine Klausel mit echt weniger Vorkommen der ansonsten selben Literale gibt. Falls also eine Klausel C vorliegt, kann jede *subsumierte* Klausel der Form $C \vee C'$, wobei C' nicht leer ist, sofort gelöscht werden. (Erst damit wird die A-geordnete Resolution zu einem Entscheidungsverfahren für die Aussagenlogik.) Intuitiv ist klar, dass dies die Vollständigkeit der A-geordneten Resolution erhält: eine subsumierte Klausel kann niemals ein „minimales Gegenbeispiel“ sein, die kleinere subsumierende Klausel wäre es, und kann deshalb ignoriert werden.

Die vielleicht wichtigste Verfeinerung der dritten Art ist die *lineare Resolution*. Wir gehen kurz auf sie ein, weil sie die Grundlage der bekannten Programmiersprache Prolog ist. Ableitungen im linearen Resolutionskalkül müssen die folgende Struktur haben:

$$C_1, \dots, C_m, C_{m+1}, \dots, C_n$$

wobei $\{C_1, \dots, C_m\} = M$ die gegebenen Klausel sind, und für alle $i = m + 1, \dots, n$ die Klausel C_i aus C_{i-1} durch Fak oder aus C_{i-1} und C_j , für ein beliebiges $j < i$ durch Res hervorgeht.

Die Idee bei der linearen Resolution ist, eine *Zielklausel* C_m schrittweise durch Fak und Res Anwendungen zu verfeinern bis die leere Klausel abgeleitet ist. Inferenzen mit Klauseln die nichts mit der Zielklausel „zu tun haben“ können nicht gemacht werden. Lineare Resolution ist übrigens nur noch schwach widerlegungsvollständig, und auch die A-geordneten Versionen der Inferenzregeln können nicht verwendet werden.

5.7 Kalküle für die Prädikatenlogik

Die Gültigkeit einer ausagenlogischen Formel zu entscheiden, ist zumindest konzeptuell einfach: man kann dazu zum Beispiel eine Wahrheitstafel aufstellen, die über den Wahrheitswert der Formel für jede mögliche Belegung der Variablen Auskunft gibt. In der Prädikatenlogik

sieht dasselbe Problem zunächst hoffnungslos aus:¹² Muss man denn nicht alle Universen und alle Abbildungen \cdot^I der Funktions- und Prädikatssymbole (siehe Abschnitt 5.3) betrachten, und führen nicht auch noch die Quantoren, speziell der Allquantor \forall , zu „unendlichen Schleifen“?

Wie wir sehen werden, ist die Lage aber nicht ganz so hoffnungslos. Es reicht nämlich aus, *ein einziges* Universum und *eine einzige* Abbildung \cdot^I für die Funktionssymbole zu betrachten. Beide ergeben sich auf einfache Art und Weise in den sogenannten *Herbrand-Interpretationen* aus der gegebenen Signatur. Die Abbildung \cdot^I für die Prädikatssymbole wird aber nicht festgelegt und erzeugt, gewissermaßen, „Suchraum“.

Herbrand-Interpretationen haben im automatischen Beweisen eine immense Bedeutung. Praktisch alle Beweiser können als Verfahren zur Exploration von Herbrand-Interpretationen angesehen werden.

5.7.1 Herbrand-Theorie

Wie bereits im Abschnitt 5.3 gehen wir auch hier von einer gegebenen Signatur aus. Die Signatur muss mindestens eine Konstante enthalten.

Eine Vorbemerkung: Ein *Grundterm* ist ein Term der keine Variablen enthält.

Definition 5.7.1 (Herbrand-Interpretation). *Eine Herbrand-Interpretation ist eine Interpretation $I = \langle U, \cdot^I \rangle$ in der gilt:*

1. *Das Universum U ist die Menge der Grundterme, auch Herbrand-Universum genannt.*
2. *Falls $t_1, \dots, t_n \in U$ und f ein n -stelliges Funktionssymbol ist, so ist $f^I(t_1, \dots, t_n) = f(t_1, \dots, t_n)$.*

Zu einer Signatur mit einer Konstante a und einem 2-stelligen Funktionssymbol f ist deshalb

$$U = \{a, f(a, a), f(a, f(a, a)), f(f(a, a), a), f(f(a, a), f(a, a)), \dots\}$$

und es gilt zum Beispiel in jeder Herbrand-Interpretation $f(a, f(a, a))^I = f(a, f(a, a))$. Terme werden also „auf sich selbst“ abgebildet. Syntax und Semantik von Termen ist *dasselbe*. Das ist weniger eigenartig als es vielleicht klingt. Vergleichbares geschieht zum Beispiel bei Programmiersprachen die Listenkonstrukteure zur Verfügung stellen: In der Programmiersprache LISP zum Beispiel steht des Ausdruck (1 2 3) für „sich selbst“, die Liste (1 2 3).

Da das Herbrand Universum aus der Menge der Grundterme besteht, wird jedes n -stellige Prädikatssymbol per \cdot^I auf eine Menge von n -Tupeln von Grundterminen abgebildet. So könnte z.B. für ein zweistelliges Prädikatssymbol P in einer Interpretation $P^I = \{\langle a, a \rangle, \langle a, f(a, a) \rangle\}$ sein. In der Literatur hat sich eingebürgert, eine Herbrand-Interpretation über diese Information sogar zu *definieren* (das Universum und die Abbildung der Funktionssymbole ist ja festgelegt). Nur unwesentlich in der Notation verschieden, setzt man eine Herbrand-Interpretation genau mit der Menge der auf wahr gesetzten Grundatome gleich. Im Beispiel hätte man $I = \{P(a, a), P(a, f(a, a))\}$.

¹² Und das ist es ja auch. Zumindest in dem Sinn dass es kein Entscheidungsverfahren für das Gültigkeitsproblem gibt.

Interessant ist nun das folgende Theorem:

Theorem 5.7.1. *Sei F ein Satz in Skolemform. Dann ist F genau dann erfüllbar, wenn F ein Herbrand-Modell besitzt.*

Da jedes Herbrand-Modell per Definition ein Modell ist, ist die eine Richtung trivial. Um die andere Richtung zu beweisen, geht man von einem beliebigen (nicht-Herbrand) Modell J für F aus und bestimmt daraus eine Herbrand-Interpretation I . Man nimmt in I ein Grundatom $P(t_1, \dots, t_n)$ auf, genau dann wenn $\langle t_1^J, \dots, t_n^J \rangle \in P^J$. Schließlich muss man dann noch zeigen, dass I ein Modell für F ist (was wir hier nicht tun).¹³

Nun zurück zur klassischen Aufgabe des Theorembeweisens. Nehmen wir an, es ist $M \models F$ zu zeigen, für eine gegebene Menge von Sätzen M und einem Satz F . Mit Hilfe von Theorem 5.7.1 und weiterer Ergebnisse kann diese Aufgabe auf aussagenlogische Beweisverpflichtungen zurückgeführt werden:

- $M \models F$
- gdw. $M \cup \{\neg F\}$ unerfüllbar ist (Anmerkung 5.3.1)
- gdw. Die Skolemform G^{Sk} von $G = \bigwedge_{H \in M} H \wedge \neg F$ unerfüllbar ist (Abschnitt 5.4)
- gdw. G^{Sk} kein Herbrand-Modell besitzt (Theorem 5.7.1)
- gdw. die Herbrand-Expansion $E(G^{Sk})$ aussagenlogisch unerfüllbar ist (Theorem 5.7.2)
- gdw. eine endliche Teilmenge von $E(G^{Sk})$ aussagenlogisch unerfüllbar ist (Kompaktheit)

Diese Kette kann man auch als Bauplan für einen einfachen Theorembeweiser für die Prädikatenlogik lesen.

Die letzten beiden Äquivalenzen müssen erläutert werden. Zunächst ist der Begriff der *Herbrand-Expansion* zu klären:¹⁴

Definition 5.7.2 (Herbrand-Expansion). *Sei $F = \forall x_1 \dots \forall x_n G$ ein Satz in Skolemform und U das Herbranduniversum zu F . Die Herbrand-Expansion von F , ist die Menge der Formeln*

$$E(F) = \{G[x_1/t_1] \dots [x_n/t_n] \mid t_1, \dots, t_n \in U\}$$

$E(F)$ ist also die Menge aller Formeln die entsteht durch alle möglichen Ersetzungen aller Variablen in G durch Grundterm.

Sei zum Beispiel $F = \forall x P(a, x)$. Dann ist

$$E(F) = \{P(a, a), P(a, f(a, a)), P(a, f(f(a, a), a)), \dots\} .$$

Jede Formel(menge), die keine Variablen enthält, kann äquivalent als aussagenlogische Formel(menge) aufgefasst werden. Man muss dazu nur die Grundatome bijektiv auf aussagenlogische Variablen abbilden. (Zum Beispiel könnte $P(a, f(a, a))$ auf A_{17} abgebildet werden.)

¹³ Korollar zu Theorem 5.7.1: Jede erfüllbare Formel besitzt ein Modell mit abzählbarem Universum (nämlich dem Herbranduniversum). Eine Konsequenz davon ist, dass die reellen Zahlen nicht durch eine Formel der Prädikatenlogik eindeutig charakterisiert werden können.

¹⁴ Hier bezeichnet $G[x/t]$ die Formel, die man aus G erhält, in dem alle freien Vorkommen der Variablen x durch den Term t ersetzt werden.

Dies ist mit dem Begriff „aussagenlogisch unerfüllbar“ gemeint. Man beachte aber, dass dann im Allgemeinen *unendlich viele* (abzählbar viele) aussagenlogische Variablen benötigt werden.

Theorem 5.7.2 (Gödel-Herbrand-Skolem). *Für jeden Satz F in Skolemform gilt: F ist genau dann erfüllbar, wenn die Formelmenge $E(F)$ im aussagenlogischen Sinn erfüllbar ist.*

Das Theorem sagt im Wesentlichen aus, dass Prädikatenlogik durch Aussagenlogik approximiert werden kann.

Die letzte Äquivalenz oben benutzt die Kompaktheitseigenschaft der Aussagenlogik: eine (unendliche) Menge von Aussagenlogischen Formeln ist unerfüllbar genau dann, wenn es eine *endliche* Teilmenge gibt, die unerfüllbar ist. Beweise dafür sind in Logikbüchern zu finden, z.B. [51]. Interessanterweise liefert der Vollständigkeitsbeweis der A-geordneten Resolution, mit ein paar weiteren Überlegungen, die Kompaktheitseigenschaft gleich mit. (Man erinnere sich, dass der Beweis von Theorem 5.6.2 nicht verlangt, dass die gegebene Klauselmenge endlich ist).

Der Algorithmus von Gilmore

Die letzte Zeile in den Äquivalenzen oben fordert, eine endliche Menge von aussagenlogischen Formeln auf Unerfüllbarkeit zu testen, algorithmisch sicher eine lösbare Aufgabe. Zusammen genommen liefern diese Äquivalenzen ein Semi-Entscheidungsverfahren für das Folgerungsproblem der Prädikatenlogik, den Algorithmus von Gilmore (F ist eine Aussage in Skolemform und $F_1, F_2, \dots, F_n, \dots$ ist eine beliebige Aufzählung von $E(F)$):

```

1: procedure Gilmore( $F$ )  $n \leftarrow 0$ 
2: repeat
3:    $n \leftarrow n + 1$ 
4: until  $(F_1 \wedge F_2 \wedge \dots \wedge F_n)$  ist unerfüllbar
5: return „unerfüllbar“

```

Der Algorithmus von Gilmore ist partiell korrekt, d.h. falls er terminiert, ist $E(F)$ unerfüllbar. Wegen der Unentscheidbarkeit des Erfüllbarkeitsproblems der Prädikatenlogik kann man auch kein stärkeres Ergebnis erwarten.

Leider ist der Algorithmus von Gilmore hoffnungslos ineffizient. Ein Hauptproblem ist, dass die Aufzählung der Herbrand-Expansion völlig ungesteuert laufen kann. Das bedeutet, dass eine zum Beweis erforderliche endliche, unerfüllbare Teilmenge $\{F_1 \wedge F_2 \wedge \dots \wedge F_n\} \subseteq E(F)$ möglicherweise erst sehr spät gefunden wird, und auch möglicherweise viele unnötige Formeln enthält, Formeln die unerfüllbarkeitserhaltend gelöscht werden könnten.

5.7.2 Prädikatenlogische Resolution

Der prädikatenlogische Resolutionskalkül vermeidet das gerade erwähnte Problem der unkontrollierten Aufzählung der Herbrand-Expansion. Die Idee ist, in Inferenzen die Variablen der beteiligten Klauseln nur so weit wie nötig zu instantiiieren. Dies geschieht durch *Unifikation*. Eine einzige Inferenz kann damit unter Umständen unendlich viele Inferenzen auf der Grundebene repräsentieren. Ähnlich bei Löschoperationen: das Löschen einer subsumierten Klausel kann dem Löschen unendlich vieler Klauseln auf der Grundebene entsprechen.

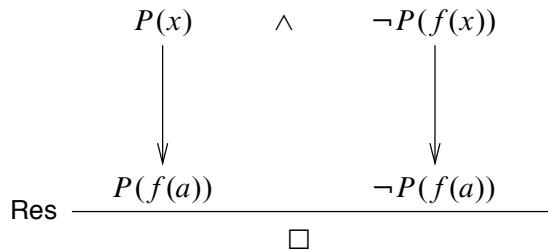
Das folgende Beispiel hilft, den Unterschied zum Algorithmus von Gilmore zu erklären. Sei $F = \forall x (P(x) \wedge \neg P(f(x)))$ ein Satz in Skolemform. Dann ist das Herbrand-Universum¹⁵ $U = \{a, f(a), f(f(a)), \dots\}$, und die ersten beiden Elemente der Herbrand-Expansion könnten sein

$$F_1 \wedge F_2 = (P(a) \wedge \neg P(f(a))) \wedge (P(f(a)) \wedge \neg P(f(f(a))))$$

Selbstverständlich kann der Test auf Unerfüllbarkeit im Schleifenrumpf des Algorithmus von Gilmore mit dem Resolutionsverfahren geführt werden. Die Klauseldarstellung von $F_1 \wedge F_2$ ist

$$\{P(a), \neg P(f(a)), P(f(a)), \neg P(f(f(a)))\} .$$

Von den vier Klauseln werden nur zwei Klauseln benötigt, um die leere Klausel \square herzuleiten (wir hatten auch „Glück“, dass sich die Unerfüllbarkeit bereits bei $n = 2$ ergibt). Grafisch:



Prädikatenlogische Resolution geht anders vor: Zunächst wird die Klauselform von F gebildet. Sie ist $\{P(x), \neg P(f(x))\}$. Nun kann die leere Klausel ohne „Raten“ inferriert werden:



Wie gesagt geschieht dies durch „Unifikation“, welche wir als Nächstes betrachten.

Unifikation

Abstrakt gesehen löst *Unifikation* das folgende Problem: Gegeben seien Beschreibungen von Objekten. Gesucht ist eine möglichst allgemeine Beschreibung derjenigen Objekte, die die gegebenen Beschreibungen gleichzeitig erfüllen.

Falls zum Beispiel $\text{Auto}(f, g, h)$ besagen soll, dass es ein Auto der Farbe f , der Geschwindigkeitsklasse g und dem Hersteller h gibt, dann liefert die Unifikation von $\text{Auto}(\text{rot}, y_1, z_1)$ („rote Autos“) und $\text{Auto}(x_2, y_2, \text{BMW})$ („Autos von BMW“) das Ergebnis $\text{Auto}(\text{rot}, y_2, \text{BMW})$ („rote Autos von BMW“).

¹⁵ Jedes Herbrand-Universum muss mindestens eine Konstante enthalten.

Technisch gesehen fasst man einen Term als Spezifikation aller seiner Grundinstanzen auf. Die Unifikation zweier Terme t_1 und t_2 liefert dann einen Term t , der genau die durch t_1 und t_2 spezifizierten Grundinstanzen spezifiziert, falls es solche Grundinstanzen gibt.

Die folgenden Definitionen präzisieren das eben Gesagte.

Definition 5.7.3 (Substitution, Instanz). *Eine Substitution ist eine endliche Menge von Variable/Term Paaren*

$$\sigma = \{x_1/t_1, \dots, x_n/t_n\} ,$$

wobei $x_i \neq t_i$ und $x_i \neq x_j$ für alle $i, j = 1, \dots, n$ mit $i \neq j$.

Sei F ein Term, ein Atom, ein Literal oder eine Klausel. Die Anwendung von σ auf F , geschrieben $F\sigma$, erhält man aus F , indem simultan jedes freie Vorkommen der Variablen x_i in F durch t_i ersetzt wird, für alle $i = 1, \dots, n$.

Eine Substitution σ ist eine Grundsubstitution für F , falls $F\sigma$ keine Variablen enthält („grund ist“). Dann heißt $F\sigma$ auch Grundinstanz von F .

Beispiel: Sei $t = g(x, y, z)$ und $\sigma = \{x/f(y), y/a\}$. Dann ist $t\sigma = g(f(y), a, z)$. Es ist also nicht $t\sigma = g(f(a), a, z)$, der Term den man aus $g(x, y, z)$ erhält, indem man, nicht-simultan, erst x durch $f(y)$ und dann y durch a ersetzt.

Substitutionen können komponiert werden: Für zwei Substitutionen σ und δ bezeichnet $\sigma\delta$ die Substitution, so dass $t\sigma\delta = (t\sigma)\delta$, für alle Terme t . Die Substitution $\sigma\delta$ auf t anwenden bedeutet also, erst σ und dann δ auf t anwenden.

Definition 5.7.4 (Unifikator). *Gegeben zwei Terme oder Atome s und t . Eine Substitution σ ist ein Unifikator (für s und t) gdw. $s\sigma = t\sigma$. Ein Unifikator σ ist allgemeinster Unifikator (most general unifier, MGU) gdw. es für jeden Unifikator σ' eine Substitution δ gibt, so dass $\sigma' = \sigma\delta$.*

Beispiel: Sei $s = \text{Auto(rot, } y_1, z_1)$ und $t = \text{Auto}(x_2, y_2, \text{BMW})$ gegeben. Dann sind alle die Substitutionen

$$\begin{aligned}\sigma_1 &= \{x_2/\text{rot}, y_2/y_1, z_1/\text{BMW}\} \\ \sigma_2 &= \{x_2/\text{rot}, y_1/y_2, z_1/\text{BMW}\} \\ \sigma_3 &= \{x_2/\text{rot}, y_1/\text{schnell}, y_2/\text{schnell}, z_1/\text{BMW}\}\end{aligned}$$

Unifikatoren für s und t , aber nur σ_1 und σ_2 sind allgemeinste Unifikatoren. Man kann zeigen, dass allgemeinste Unifikatoren eindeutig bestimmt sind, bis auf Umbenennung der beteiligten Variablen. Das rechtfertigt es, mit $\text{mgu}(s, t)$ „den“ allgemeinsten Unifikator zu bezeichnen.

Natürlich kann es auch sein, dass für zwei Terme überhaupt kein Unifikator existiert. Zum Beispiel für $f(a)$ und $f(b)$. Was man benötigt, ist ein Algorithmus der zu gegebenen Termen entscheidet, ob sie unifizierbar sind, und, falls ja, einen allgemeinsten Unifikator berechnet. Der folgende Algorithmus leistet dies. Er arbeitet auf einer Menge U von Gleichungen, Paare von Termen die simultan unifiziert werden sollen. Das Ziel ist, durch Dekomposition der Termstruktur eine Menge von Gleichungen bestimmter Struktur zu erhalten, so dass der Unifikator unmittelbar abgelesen werden kann.

Unifikationsalgorithmus

Eingabe: Zwei Terme s und t .

1. Setze $U = \{s = t\}$

2. Solange wie möglich, wende die folgenden Umformungsregeln auf U an:

$\{x = x\} \cup N \longrightarrow N$	(Trivial)
$\{x = t\} \cup N \longrightarrow \{x = t\} \cup N[x/t]$	(Bindung)
falls x in N vorkommt und x nicht in t vorkommt	
$\{x = t\} \cup N \longrightarrow \text{FAIL}$	(Occur Check)
falls t keine Variable ist und x in t vorkommt	
$\{f(s_1, \dots, s_m) = f(t_1, \dots, t_m)\} \cup N \longrightarrow \{s_1 = t_1, \dots, s_m = t_m\} \cup N$	(Dekomposition)
$\{f(s_1, \dots, s_m) = g(t_1, \dots, t_m)\} \cup N \longrightarrow \text{FAIL}$	(Konflikt)
falls $f \neq g$	
$\{t = x\} \cup N \longrightarrow \{x = t\} \cup N$	(Orientierung)
falls t keine Variable ist	

Ausgabe: U .

Die Präsentation von Unifikationsalgorithmen als Regelsystem geht auf [39] zurück. In der ursprünglichen Fassung von Robinson [49] wurde eine Pseudo-Code Darstellung verwendet. Regelsysteme haben den Vorteil leichter analysiert werden zu können.

Proposition 5.7.1 (Korrektheit und Vollständigkeit des Unifikationsalgorithmus). *Der Unifikationsalgorithmus terminiert und es tritt genau einer der beiden folgenden Fälle ein.*

1. Erfolgsfall:

1. U ist von der Form $U = \{x_1 = t_1, \dots, x_n = t_n\}$, und
2. $x_i \neq x_j$, für $1 \leq i, j \leq n$ mit $i \neq j$, und
3. x_i kommt nicht in t_j vor, für $1 \leq i, j \leq n$.

In diesem Fall ist

$$\sigma = \{x_1/t_1, \dots, x_n/t_n\}$$

allgemeinster Unifikator von s und t .

2. Misserfolgsfall: $U = \text{FAIL}$.

In diesem Fall gibt es keinen Unifikator für s und t .

Siehe [34] für ein nicht mehr ganz neues, aber immer noch lohnendes Übersichtspapier zur Unifikation.

Theorie-Unifikation

„Unifikation“ kann allgemeiner als hier betrachtet werden. Besonders interessant ist, Funktionssymbole mit gewissen Eigenschaften auszustatten und spezielle Unifikationsalgorithmen dafür zu entwerfen. Falls zum Beispiel das Funktionssymbol f den Status der Assoziativität und der Kommutativität erhält, dann sind die Terme $f(x, f(a, b))$ und $f(f(b, a), c)$ „Theorie-unifizierbar“, und $\sigma = \{x/c\}$ ist ein Unifikator. In der Tat war der Einsatz eines Beweisers mit eingebauter Theorie-Unifikation für assoziative und kommutative Funktionssymbole entscheidend, um den Beweis der in der Einleitung erwähnten Robbins-Vermutung zu finden.

Kalkül

Der Resolutionskalkül besteht aus den folgenden Inferenzregeln (vgl. Abschnitt 5.6.1 für die aussagenlogischen Varianten).

$$\text{Res} \frac{C \vee A \quad \neg B \vee D}{(C \vee D)\sigma} \quad \text{falls } \sigma = \text{mgu}(A, B)$$

$$\text{Fak} \frac{C \vee A \vee B}{(C \vee A)\sigma} \quad \text{falls } \sigma = \text{mgu}(A, B)$$

Die aussagenlogische Resolutionsregel **Res** basiert auf komplementären Literalen A und $\neg A$. In der prädikatenlogischen Version werden stattdessen die Literale A und $\neg B$ nach Anwendung des MGUs σ komplementär. Statt des *allgemeinsten* Unifikators σ könnte die Regel auch über *beliebige* Unifikatoren definiert werden. Das würde aber der Idee widersprechen, dass die Inferenzen auf so allgemeiner Ebene wie möglich durchgeführt werden sollen. Analoges gilt für die Fak Inferenzregel.

Beispiel:

$$\text{Res} \frac{Q(z) \vee P(z, z) \quad \neg P(x, y)}{Q(x)} \quad \sigma = \text{mgu}(P(z, z), P(x, y)) = \{z/x, y/x\}$$

$$\text{Fak} \frac{Q(z) \vee P(z, a) \vee P(a, y)}{Q(a) \vee P(a, a)} \quad \sigma = \text{mgu}(P(z, a), P(a, y)) = \{z/a, y/a\}$$

Ableitungsbeispiel:

wie in Abschnitt 5.6.1. Als einzige Änderung muss sichergestellt werden, dass die beiden Elternklauseln bei Res Inferenzen keine Variablen gemeinsam haben. Es reicht aus, die Variablen in einer der beiden Elternklauseln ggf. vorher umzubenennen. So können zum Beispiel die beiden Klauseln $P(x)$ und $\neg P(f(x))$ nicht resolvieren, $P(x)$ und $\neg P(f(x'))$ aber schon.

Die Begriffe der saturierten Klauselmenge (Def. 5.6.1), des deduktiven Abschlusses (Def. 5.6.2), sowie die GivenClauseLoop Prozedur aus Abschnitt 5.6.1 können ohne weitere Änderungen auch für die prädikatenlogischen Inferenzregeln verwendet werden.

A-Geordnete Resolution

Auch A-Ordnungen können eingebracht werden. Die Inferenzregeln sehen dann so aus:

$$\text{A-Res } \frac{C \vee A \quad \neg B \vee D}{(C \vee D)\sigma} \quad \text{falls } \begin{cases} \sigma = \text{mgu}(A, B), \\ A\sigma \text{ ist maximal in } (C \vee A)\sigma, \text{ und} \\ \neg A\sigma \text{ ist maximal in } (\neg B \vee D)\sigma \end{cases}$$

$$\text{Fak } \frac{C \vee A \vee B}{(C \vee A)\sigma} \quad \text{falls } \begin{cases} \sigma = \text{mgu}(A, B), \text{ und} \\ A\sigma \text{ ist maximal in } (C \vee A \vee B)\sigma \end{cases}$$

Man erinnere sich, dass im aussagenlogischen Fall die Ordnung \succ als *totale* Ordnung vorausgesetzt war. Im prädikatenlogischen Fall fordert man, dass \succ total auf *Grundterminen* ist. Zum Beispiel muss für die beiden Atome $R(a, b)$ und $R(b, a)$ deshalb $R(a, b) \succ R(b, a)$ oder $R(b, a) \succ R(a, b)$ gelten. Andererseits, falls Variablen beteiligt sind, kann beispielsweise weder $R(x, y) \succ R(y, x)$ noch $R(y, x) \succ R(x, y)$ gelten.¹⁶ Das bedeutet, dass eine Klausel nun *mehrere verschiedene* maximale Literale enthalten kann. In der Klausel $R(x, y) \vee R(y, x)$, zum Beispiel, sind beide Literale maximal. Die Ordnungseinschränkungen sind deshalb nicht mehr so effektiv wie im aussagenlogischen Fall.

Korrektheit und Vollständigkeit

Die Korrektheits- und Vollständigkeitsergebnisse, Theoreme 5.6.1 und 5.6.2 gelten auch für die prädikatenlogische Resolution. Ebenso können die Verfeinerungen aus Abschnitt 5.6.3 angepasst werden (siehe dazu [2]).

Die Korrektheitseigenschaft zeigt man am besten direkt, ähnlich wie im aussagenlogischen Fall. Die Vollständigkeitseigenschaft ergibt sich aus einigen wenigen Überlegungen, die im Folgenden kurz skizziert werden.

Unter Ausnutzung des Gödel-Herbrand-Skolem Theorems, Theorem 5.7.2, kann eine Klauselmenge M als die Menge aller Grundinstanzen aller Klauseln in M aufgefasst werden. Sei M^{gr} diese Menge. Um die Vollständigkeit von z.B. A-geordneter Resolution zu beweisen, nimmt man an, dass M unerfüllbar ist. Aus dem eben Gesagten folgt, dass auch M^{gr} unerfüllbar ist. Nun bildet man den deduktiven Abschluss $(M^{\text{gr}})^*$ von M^{gr} . Mit Theorem 5.6.2 folgt, dass $(M^{\text{gr}})^*$ die leere Klausel enthält. Was man aber eigentlich zeigen muss, ist, dass M^* , der deduktive Abschluss von M unter den Inferenzregeln der A-geordneten Resolution, die leere Klausel enthält. Im Wesentlichen muss man dazu die folgende Lifting-Eigenschaft beweisen (was wir hier nicht tun):

Lifting-Eigenschaft: Seien C_1 und C_2 zwei Klauseln, die keine Variablen gemeinsam haben, und seien γ_1 und γ_2 zwei Grundsubstitutionen für C_1 und C_2 .

Für jede Grundklausel D gilt: falls es eine Inferenz

$$\text{A-Res } \frac{C_1\gamma_1 \quad C_2\gamma_2}{D}$$

¹⁶ Man braucht aus Vollständigkeitsgründen nämlich die Eigenschaft, dass eine Ordnungsbeziehung unter Instantiierung erhalten bleibt. Wäre $R(x, y) \succ R(y, x)$, dann müsste sowohl $R(a, b) \succ R(b, a)$ als auch $R(b, a) \succ R(a, b)$ gelten, und per Transitivität $R(a, b) \succ R(a, b)$. Dann aber wäre \succ keine strikte Ordnung mehr.

gibt, dann gibt es auch eine Klausel C , eine Substitution δ und eine Inferenz

$$\text{A-Res} \frac{C_1 \quad C_2}{C}$$

so dass $C\delta = D$.

Mit anderen Worten: Jede A-Res Inferenz auf Grundebene, insbesondere falls $C_1\gamma_1, C_2\gamma_2 \in (M^{\text{gr}})^\star$ ist, kann auf der prädikatenlogischen Ebene „simuliert“ werden, in dem Sinn, dass die Resolvente D der Grundebene eine Instanz der Resolvente C der prädikatenlogischen Ebene ist. Eine analoge Lifting-Eigenschaft muss für die A-Fak Inferenzregel gezeigt werden. Daraus folgt dann leicht, dass die Menge aller Grundinstanzen von M^\star genau die Menge $(M^{\text{gr}})^\star$ ist. Falls also insbesondere $\square \in (M^{\text{gr}})^\star$ ist (was wir wissen), dann ist auch $\square \in M^\star$, was zu zeigen ist.

5.8 Weitere Betrachtungen

Das Hauptziel dieses Kapitels war, einige wichtige Techniken des automatischen Schließens vorzustellen. Der Fokus lag auf dem DPLL-Verfahren für die Aussagenlogik und dem Resolutionsverfahren für die Prädikatenlogik. Für mehr als die Grundideen zu erläutern war jedoch kein Platz. Tiefer gehende Übersichtsarbeiten findet man in den Handbüchern [10, 28, 50]. Deutsch sprachige einschlägige Lehrbücher sind [7, 8, 11, 29, 51]. Aktuelle Entwicklungen verfolgt man in den Konferenzen CADE, IJCAR, LPAR, Tableaux, CAV, SAT, CP, und JELIA. Einschlägige Zeitschriften sind z.B. Journal of Automated Reasoning, Journal of Symbolic Computation, Journal of Artificial Intelligence Research, und Journal of Applied Logic. Allgemeine KI-Konferenzen und Zeitschriften sind ebenfalls relevant. Mit der TPTP Library (Thousands of Problems for Theorem Provers) [56] steht eine umfangreiche und stetig wachsende Sammlung von Benchmark Problemen zur Verfügung, die auch für jährlich stattfindende Wettbewerbe verwendet wird.

Abschließend noch einige Hinweise zu Themen, die hier nicht behandelt wurden. Siehe die oben erwähnten Handbücher oder die genannte Literatur zum Nachlesen.

Geschichte. Wer sich für die Geschichte des automatischen Schließens interessiert, kann in [16] nachlesen. Originalarbeiten findet man in der Kollektion [52]. In den 1990er Jahren förderte die DFG die Entwicklung des automatischen Schließens in einem „Schwerpunktprogramm Deduktion“. Die Ergebnisse sind in dem dreibändigen Werk [9] zusammengefasst.

Andere Kalküle. Andere Kalküle, die sich für automatisches Schließen eignen, sind Konnektionenkalküle [6, 36] und Tableaux [15]. Die Beweisfindung in diesen Kalkülen erfolgt über die Zerlegung der Eingabeformel gemäß ihrer Junktorenstruktur. Im Gegensatz zur Resolution arbeiten sie also „analytisch“, ohne „neue“ Formeln abzuleiten. Tableau Kalküle werden heute in erster Linie für Beschreibungslogiken und Modallogiken eingesetzt.

Ganz anders funktionieren „instanzbasierte Verfahren“. (Siehe [5] für einen Überblick). Ihnen ist gemeinsam, die zu untersuchende Klauselmenge durch Instanzen dieser Klauseln anzureichern, ähnlich zum Gilmore-Verfahren in Abschnitt 5.7.1. Als ein wesentli-

cher Unterschied zum Gilmore Verfahren erfolgt die Instanziierung aber besser kontrolliert, gesteuert durch Unifikation. Dies ermöglicht zum Beispiel eine prädikatenlogische Version des DPLL-Verfahrens aus Abschnitt 5.5 [4].

Entscheidungsverfahren und Modellgenerierung. Ein Entscheidungsverfahren für das Erfüllbarkeitsproblem der Prädikatenlogik kann es aus theoretischen Gründen nicht geben. Man kann aber versuchen, für Teilsprachen der Prädikatenlogik Verfeinerungen des Resolutionskalküls (oder anderer Kalküle) zu finden, um Entscheidungsverfahren dafür zu erhalten (siehe [22]).

Im allgemeineren Ansatz der „Modellgenerierung“ versucht man, ein Modell einer gegebenen Formel zu finden (statt Unerfüllbarkeit zu beweisen). Dazu durchsucht man den Raum der passenden Interpretationen mit *endlichem* Individuenbereich, als Kandidaten für solch ein Modell. Durch systematische Suche kann sichergestellt werden, dass ein solches Modell (theoretisch wenigstens) immer gefunden wird, falls eines existiert. In der Praxis verwendet man Verfahren, die das Problem auf Aussagenlogik reduzieren, oder dedizierte Kalküle.

Theorieschließen. Damit ist gemeint, ein allgemeines Beweisverfahren – DPLL, Resolution oder ein anderer Kalkül – mit einem Beweisverfahren für eine spezielle Theorie zu kombinieren. Dadurch entfällt die Axiomatisierung dieser Theorie in der Eingabeformel, das bringt Effizienzvorteile. Für wichtige entscheidbare Theorien (zum Beispiel lineare Arithmetik) ist auch keine Axiomatisierung bekannt, die zu Entscheidungsverfahren in den gebräuchlichen Kalkülen führen. Die Koppelung mit einem Spezialverfahren ist dann aus praktischer Sicht unumgänglich. Theorieschließen wurde erstmals für den Resolutionskalkül entwickelt [55]. Eine aktuellere Version ist [3].

Auch die heute weit verbreiteten SMT Verfahren (siehe Abschnitt 5.1) können hier eingeordnet werden. Im sogenannten DPLL(T) Ansatz wird das DPLL-Verfahren mit einem beliebigen Entscheidungsverfahren für das Grundfragment einer beliebigen Theorie oder Kombination von Theorien kombiniert. Ein gutes Lehrbuch (auch) zu diesem Thema ist [13].

Literaturverzeichnis

- [1] Ahrendt, W., Baar, T., Beckert, B., Bubel, R., Giese, M., Hähnle, R., Menzel, W., Mołkowski, W., Roth, A., Schlager, S., und Schmitt, P. H. (2005). The KeY tool. *Software and System Modeling*, 4:32–54.
- [2] Bachmair, L. und Ganzinger, H. (2001). Resolution Theorem Proving. In Robinson, A. und Voronkov, A., editors, *Handbook of Automated Reasoning*. North Holland.
- [3] Bachmair, L., Ganzinger, H., und Waldmann, U. (1994). Refutational theorem proving for hierachic first-order theories. *Appl. Algebra Eng. Commun. Comput.*, 5:193–212.
- [4] Baumgartner, P. (2000). FDPLL – A First-Order Davis-Putnam-Logemann-Loveland Procedure. In McAllester, D., editor, *CADE-17 – The 17th International Conference on Automated Deduction*, volume 1831 of *Lecture Notes in Artificial Intelligence*, pages 200–219. Springer.

- [5] Baumgartner, P. und Thorstensen, E. (2010). Instance based methods – a brief overview. *KI – Künstliche Intelligenz*, 24:35–42.
- [6] Bibel, W. (1981). On Matrices with Connections. *Journal of the Association for Computing Machinery*, 28:633–645.
- [7] Bibel, W. (1992). *Deduktion*, volume 6.2 of *Handbuch der Informatik*. Oldenburg.
- [8] Bibel, W. (1993). *Wissensrepräsentation und Inferenz*. Vieweg.
- [9] Bibel, W. und Schmitt, P. H., editors (1998). *Automated Deduction. A basis for applications*. Kluwer Academic Publishers.
- [10] Biere, A., Heule, M., van Maaren, H., und Walsh, T., editors (2009). *Handbook of Satisfiability*. IOS Press.
- [11] Bläsius, K. und Bürkert, H.-J. (1987). *Deduktionssysteme*. Oldenburg.
- [12] Böhme, S. und Nipkow, T. (2010). Sledgehammer: Judgement day. In Giesl, J. und Hähnle, R., editors, *IJCAR*, volume 6173 of *Lecture Notes in Computer Science*, pages 107–121. Springer.
- [13] Bradley, A. und Manna, Z. (2007). *The Calculus of Computation*. Springer.
- [14] Bundy, A., editor (1994). *Automated Deduction – CADE 12*, LNAI 814, Nancy, France. Springer-Verlag.
- [15] D'Agostino, M., Gabbay, D. M., Hähnle, R., und Posegga, J., editors (1999). *Handbook of Tableau Methods*. Kluwer.
- [16] Davis, M. (2001). Chapter 1. The Early History of Automated Deduction. In Robinson, A. und Voronkov, A., editors, *Handbook of Automated Reasoning*, pages 3–15. North Holland.
- [17] Davis, M., Logemann, G., und Loveland, D. (1962). A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397.
- [18] Davis, M. und Putnam, H. (1960). A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215.
- [19] de Nivelle, H., Hustadt, U., und Schmidt, R. A. (2000). Resolution-based methods for modal logics. *Logic Journal of the IGPL*, 8(3):265–292.
- [20] Detlefs, D. L., Leino, K. R. M., Nelson, G., und Saxe, J. B. (1998). Extended static checking. SRC Research Report 159, 130 Lytton Ave., Palo Alto.
- [21] Dimopoulos, Y., Nebel, B., und Köhler, J. (1997). Encoding Planning Problems in Non-monotonic Logic Programs. In *European Conference on Planning (ECP-97)*. Springer.
- [22] Fermüller, C. G., Leitsch, A., Hustadt, U., und Tammet, T. (2001). Resolution decision procedures. In Robinson, A. und Voronkov, A., editors, *Handbook of Automated Reasoning*, volume II, chapter 25, pages 1791–1849. Elsevier Science B.V.
- [23] Fitting, M. (1996). *First-order logic and automated reasoning* (2. ed.). Graduate texts in computer science. Springer.
- [24] Fujita, M., Slaney, J., und Bennett, F. (1995). Automatic generation of some results in finite algebra. In *IJCAI-95 – Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal*, pages 52–57.
- [25] Gordon, M. J. C. und Melham, T. F., editors (1993). *Introduction to HOL: A theorem proving environment for higher order logic*. Cambridge University Press.
- [26] Graf, P. (1994). Extended Path-Indexing. In [14].
- [27] Green, C. C. (1969). Theorem-Proving by Resolution as a basis for Question-Answering Systems. In Meltzer, B. und Michie, D., editors, *Machine Intelligence 4*, pages 183–205. American Elsevier Publishing Company, Inc.

- [28] Harrison, J. (2009). *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press.
- [29] Hölldobler, S. (2003). *Logik und Logikprogrammierung*. Synchron.
- [30] Hölldobler, S. und Schneeberger, J. (1990). A New Deductive Approach to Planning. *New Generation Computing*, 8:225–244.
- [31] Hustadt, U., Motik, B., und Sattler, U. (2004). Reducing shiq-description logic to disjunctive datalog programs. In Dubois, D., Welty, C. A., und Williams, M.-A., editors, *KR*, pages 152–162. AAAI Press.
- [32] Kaufmann, M. und Moore, J. (1996). Acl2: An industrial strength version of nqthm. In *Proceedings of Eleventh Annual Conference on Computer Assurance (COMPASS-96)*, pages 23–34. IEEE Computer Society Press.
- [33] Kautz, H. und Selman, B. (1996). Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search. In *Proceedings of the 13th National Conference on Artificial Intelligence*, Portland, OR, USA.
- [34] Knight, K. (1989). Unification: A multidisciplinary survey. *ACM Computing Surveys*, 21(1):93–124.
- [35] Kühlwein, D., van Laarhoven, T., Tsivtsivadze, E., Urban, J., und Heskes, T. (2012). Overview and evaluation of premise selection techniques for large theory mathematics. In Gramlich, B., Miller, D., und Sattler, U., editors, *IJCAR*, volume 7364 of *Lecture Notes in Computer Science*, pages 378–392. Springer.
- [36] Letz, R. (1998). Clausal Tableaux. In [9].
- [37] Li, C. M. und Anbulagan (1997). Heuristics based on unit propagation for satisfiability problems. In *Proceedings of 15th IJCAI*, pages 366–371, Nagoya, Aichi, Japan.
- [38] Lifschitz, V. (2002). Answer set programming and plan generation. *Artifical Intelligence*, 138:39–54.
- [39] Martelli, A. und Montanari, H. (1982). An efficient unification algorithmus. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282.
- [40] McCune, W. (1996). Robbins Algebras Are Boolean.
<http://www.mcs.anl.gov/home/mccune/ar/robbins/index.html>.
- [41] Moskewicz, M. W., Madigan, C. F., Zhao, Y., Zhang, L., und Malik, S. (2001). Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th Design Automation Conference (DAC'01)*.
- [42] NewYorkTimes (1996). Computer Math Proof Shows Reasoning Power. New York Times.
- [43] Niemelä, I. (1999). Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3–4):241–273.
- [44] Nieuwenhuis, R., Oliveras, A., und Tinelli, C. (2006). Solving SAT and SAT Modulo Theories: from an Abstract Davis-Putnam-Logemann-Loveland Procedure to DPLL(T). *Journal of the ACM*, 53(6):937–977.
- [45] Nipkow, T. und Paulson, L. C. (1992). Isabelle-91. In Kapur, D., editor, *Proceedings of the 11th International Conference on Automated Deduction*, pages 673–676, Saratoga Springs, NY. Springer-Verlag LNAI 607. System abstract.
- [46] Nonnengart, A. und Weidenbach, C. (2001). Computing small clause normal forms. In [50], pages 335–367.
- [47] Ranise, S. und Tinelli, C. (2006). Satisfiability modulo theories. *Trends and Controversies – IEEE Intelligent Systems Magazine*, 21(6):71–81.

- [48] Reif, W., Schellhorn, G., Stenzel, K., und Balser, M. (1998). Structured specifications and interactive proofs with KIV. In [9].
- [49] Robinson, J. (1965). A machine-oriented logic based on the resolution principle. *JACM*, 12(1):23–41.
- [50] Robinson, J. A. und Voronkov, A., editors (2001). *Handbook of Automated Reasoning (in 2 volumes)*. Elsevier and MIT Press.
- [51] Schöning, U. (2000). *Logik für Informatiker*. Spektrum Akademischer Verlag.
- [52] Siekmann, J. und Wrightson, G., editors (1983). *Automation of Reasoning – Classical Papers on Computational Logic*. Symbolic Computation. Springer.
- [53] Silva, J. P. M. und Sakallah, K. A. (1999). Grasp: A search algorithm for propositional satisfiability. *IEEE Trans. Computers*, 48(5):506–521.
- [54] Slaney, J., Lusk, E., und McCune, W. (1994). SCOTT: Semantically constrained Otter (system description). In [14], pages 764–768.
- [55] Stickel, M. (1985). Automated Deduction by Theory Resolution. *Journal of Automated Reasoning*, 1:333–355.
- [56] Sutcliffe, G. und Suttner, C. (1998). The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203.
- [57] Voronkov, A. (2001). Algorithms, datastructures, and other issues in efficient automated deduction. In R. Gore, A. L. und Nipkow, T., editors, *IJCAR 2001*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 13–28. Springer Verlag, Berlin, Heidelberg, New-York.
- [58] Zhang, H. und Hsiang, J. (1994). Solving open quasigroup problems by propositional reasoning. In *Proc. of International Computer Symposium*, Hsinchu, Taiwan.
- [59] Zhang, H. und Stickel, M. E. (1994). Implementing the davis-putnam algorithm by tries. Technical report, Department of Computer Science, The University of Iowa.
- [60] Zhu, Y. und Plaisted, D. (1997). Folplan: A semantically guided first-order planner. In *Proceedings of the 10th International FLAIRS Conference*.

Teil II

Theorie und Methoden

6 Nichtmonotones Schließen

Gerhard Brewka

In diesem Kapitel wird zunächst die Rolle des nichtmonotonen Schließens für die Künstliche Intelligenz erörtert. Nach der Darstellung einiger einfacher Formen der Nichtmonotonie, wie sie in verschiedenen verbreiteten Systemen vorkommt, stellt Abschnitt 6.2 einige der wichtigsten existierenden nichtmonotonen Logiken dar: Reiters Default Logik, Moores Autoepistemische Logik und McCarthys Circumscription. Abschnitt 6.3 beschreibt dann ausführlich einen Ansatz, der Default-Schließen auf inkonsistenztolerantes Schließen zurückführt. Dabei werden bevorzugte maximal konsistente Teilmengen der Prämissen berücksichtigt. Es wird gezeigt, wie durch eine geeignete Definition der bevorzugten Teilmengen Prioritäten zwischen Default-Regeln auf einfache Weise ausgedrückt werden können. Abschnitt 6.4 geht auf den Zusammenhang zwischen nichtmonotonen Logiken und Logikprogrammierung ein und diskutiert die Antwortmengenprogrammierung, einen neueren Ansatz zum Problemlösen. Der folgende Abschnitt 6.5 bietet eine kurze Einführung in das Gebiet der formalen Argumentation. Das Kapitel schließt mit einem kurzen Ausblick in Abschnitt 6.6.

6.1 Einführung

Unser empirisches Wissen über die Welt ist stets unvollständig, und sehr häufig lassen sich aus ihm nicht alle Fakten ableiten, die wir benötigen, um darauf Entscheidungen, Planungen und Aktionen zu gründen. Trotz solcher Lücken in unserem Wissen sind wir natürlich gezwungen zu handeln. Vielfach verwenden wir deshalb Regeln mit Ausnahmen, um das fehlende Wissen zu ergänzen. Solche Regeln drücken aus, was typischerweise der Fall ist, und können dazu benutzt werden, plausible Konklusionen abzuleiten, sofern keine der Konklusion widersprechende Information vorliegt. Jeder intelligente Agent muss in der Lage sein, solche Regeln – im KI-Deutsch werden sie auch Default-Regeln genannt – sinnvoll zu handhaben.

Eine zweite für uns wichtige Fähigkeit ist der Umgang mit inkonsistenter Information. Es kommt ständig vor, dass sich die uns vorliegende Information widerspricht, z.B. weil sie aus unterschiedlichen Quellen stammt. Natürlich dürfen wir uns dadurch nicht lähmeln lassen. Wir müssen auf sinnvolle Weise mit der inkonsistenten Information umgehen.

Für beide Probleme bietet die klassische Prädikatenlogik für sich genommen keine Lösung: im Falle inkonsistenter Prämissen liefert die Prädikatenlogik die Menge aller Formeln als Theoreme. Das ist sicher nicht das, was wir wollen, denn wer auf widersprüchliche Information stößt, wird sicher nicht anfangen, auf einmal alle beliebigen Sätze für wahr zu halten. Auch die Behandlung von Regeln mit Ausnahmen lässt sich in der klassischen Logik nicht adäquat modellieren. Natürlich können wir eine Regel wie *Vögel fliegen typischerweise* folgendermaßen

darstellen:

$$\begin{aligned} \forall x. \text{Vogel}(x) \wedge \neg \text{Ausnahme}(x) &\Rightarrow \text{Fliegt}(x) \\ \forall x. \text{Ausnahme}(x) &\Leftrightarrow \text{Pinguin}(x) \vee \text{Strauss}(x) \vee \neg \text{HatFlügel}(x) \dots \end{aligned}$$

Diese Darstellung erfordert aber eine vollständige Auflistung aller möglichen Ausnahmen, eine für sich genommen unmögliche Aufgabe, denn auch das Wissen um Ausnahmen ist natürlich so gut wie immer unvollständig. Aber selbst wenn eine solche vollständige Liste verfügbar wäre, wäre die Darstellung immer noch inadäquat: um für einen bestimmten Vogel, sagen wir Tweety, abzuleiten, dass er fliegt, ist es notwendig zu beweisen, dass kein Ausnahmefall vorliegt, d.h. dass Tweety kein Pinguin ist, kein Strauß, dass er Flügel hat, usw. Das aber wollen wir gerade nicht: wir wollen *Tweety fliegt* auch ableiten können, wenn nicht gezeigt werden kann, dass Tweety eine Ausnahme ist, nicht nur dann, wenn gezeigt werden kann, dass er keine Ausnahme ist.

Worin ist diese Schwierigkeit begründet? Die klassische Logik besitzt folgende Monotonieeigenschaft: Für alle Mengen von Prämissen A und Formeln p, q gilt:

$$\text{Wenn } A \vdash q, \text{ dann } A \cup \{p\} \vdash q.$$

Zusätzliche Information kann also nie alte Konklusionen ungültig machen (\vdash bezeichnet die klassische Ableitbarkeitsrelation). Wie wir im obigen Beispiel gesehen haben, wollen wir eine Default-Regel dazu benutzen, um eine plausible Konklusion abzuleiten, sofern nichts auf einen Ausnahmefall hindeutet. Wenn wir später zusätzliche Information erhalten, die besagt, dass entgegen unseren Erwartungen doch eine Ausnahme vorliegt, dann muss diese Konklusion zurückgenommen werden. Jedes Schließen, das auf Regeln mit Ausnahmen basiert, muss deshalb in diesem Sinne nichtmonoton sein. Dasselbe gilt konsequenterweise auch für eine Logik, die solches Schließen formalisiert.

Uns geht es hier vor allem um Default-Schließen. Es gibt aber andere Formen nichtmonotonen Schließens, die wenigstens erwähnt werden sollen. Wie gesagt erlauben Default-Regeln die Ableitung plausibler Konklusionen aus unvollständigem Wissen, sofern keine widersprüchliche Information vorliegt. Eine ganz andere Form nichtmonotonen Schließens ist das autoepistemische Schließen. Dabei wird Wissen über das eigene Wissen (autoepistemisches Wissen) zur Ableitung korrekter Schlüsse verwendet. Das Standardbeispiel lautet:

Ich kenne alle meine Brüder.

(= Wenn x mein Bruder ist, so weiß ich das.)

Ich habe keine Information, dass Peter mein Bruder ist.

Also ist Peter nicht mein Bruder.

Die Nichtmonotonie entsteht in diesem Fall deshalb, weil die Bedeutung von Aussagen über das eigene Wissen kontextabhängig ist: wenn wir in unserem Beispiel zusätzliche Information erhalten, die besagt, dass Peter doch mein Bruder ist, so wissen wir, dass die erste Prämisse falsch war, als sie zur Ableitung von *Peter ist nicht mein Bruder* verwendet wurde. Aber das heißt nicht, dass diese Prämisse nun aufgegeben werden muss. Die Prämisse bezieht sich auf das jeweils aktuelle Wissen, und es ist durchaus sinnvoll anzunehmen, dass sie bezogen auf das jetzt erweiterte Wissen zutrifft. Eine genauere Analyse des autoepistemischen Schließens findet sich in [49].

Auch das Schließen auf der Basis inkonsistenter Information (inkonsistenztolerantes Schließen) ist nichtmonoton. Betrachten wir die inkonsistente Prämissemenge

$$\{p, \neg p, q, r, s\}$$

Wie würde ein intelligenter Agent mit solcher Information umgehen? Er würde sicherlich nicht alle verfügbare Information als gleichermaßen wertlos ansehen. Eine Möglichkeit wäre es, die maximal-konsistenten Teilmengen der Prämissen zu betrachten und als ableitbar anzusehen, was aus all diesen Teilmengen abgeleitet werden kann. In unserem Beispiel hätte das den Effekt, dass p und $\neg p$, eliminiert¹ würden und nur die anderen Prämissen für Ableitungen verwendet werden könnten. Natürlich kann dabei das Zufügen von neuer Information, etwa $\neg q$, weitere Prämissen unwirksam machen. Damit ist auch diese Form von Schließen nichtmonoton. Wir werden in Abschnitt 6.3 noch einmal ausführlicher auf inkonsistenztolerantes Schließen zurückkommen.

Bevor wir uns einigen wichtigen Formalisierungen zuwenden, wollen wir noch einige Beispiele diskutieren, die die Relevanz nichtmonotonen Schließens für die Künstliche Intelligenz zeigen. Historisch gesehen wurden die Arbeiten auf diesem Gebiet vor allem motiviert durch das berühmte Frame-Problem: Wie lässt sich adäquat repräsentieren, dass die meisten Objekte sich nicht ändern, wenn ein Ereignis eintritt? Das Problem trat vor allem im Zusammenhang mit dem Situationskalkül deutlich zutage. Der Situationskalkül wurde von McCarthy und Hayes entwickelt [45, 48], um in einem logischen Rahmen das Schließen über Aktionen und Ereignisse in der Zeit zu modellieren. Damit die Effekte von Ereignissen dargestellt werden können, werden Fakten mit den Situationen indiziert, in denen sie gelten, etwa:¹

$$\begin{aligned} &Holds(In(Fred, Kitchen), Sit105) \\ &Holds(Color(Kitchen, Red), Sit105) \end{aligned}$$

Ereignisse produzieren neue Situationen, etwa:

$$Sit106 = Result(Go(Fred, Bathroom), Sit105)$$

Weitere Axiome beschreiben, wie Ereignisse die Welt verändern:

$$\forall x, y, s. Holds(In(x, y), Result(Go(x, y), s))$$

Das Problem ist nun, z.B. zu zeigen, welche Farbe die Küche in $Sit106$ hat. Um ableiten zu können, dass sie immer noch rot ist, brauchen wir folgendes Axiom:

$$\forall x, y, v, w, s. Holds(Color(x, y), s) \Rightarrow Holds(Color(x, y), Result(Go(v, w), s))$$

Solche *Frame-Axiome* sind für jedes Paar bestehend aus einem Ereignis und einem durch dieses Ereignis nicht veränderten Faktum nötig. Das macht den Ansatz für die Praxis völlig unbrauchbar. Eine mögliche Lösung dieses Problems wäre die Einführung eines *Persistenz-Defaults*, das informell etwa so lauten könnte:

Ereignisse verändern Eigenschaften normalerweise nicht.

¹ In diesen und den folgenden Beispielen beginnen Konstanten mit großen Buchstaben, Variablen mit kleinen.

Mit einem solchen Default, so die Grundidee, ist nur noch die Beschreibung der tatsächlichen Effekte von Ereignissen erforderlich. Explizite Frame-Axiome erübrigen sich. Ihre Konklusionen liefert das Persistenz-Default.

Mittlerweile hat sich allerdings herausgestellt, dass eine einfache Repräsentation dieses Defaults in einer der inzwischen entwickelten nichtmonotonen Logiken oft nicht zu den erwarteten Resultaten führt. Eine Analyse der dabei auftretenden Probleme und eine Darstellung einiger Lösungsversuche finden sich in [27] und [36].

Ein verwandtes Problem ist das Qualifikationsproblem: wie lässt sich adäquat repräsentieren, dass Aktionen fehlschlagen können? Wenn man z.B. den Autoschlüssel im Zündschloß dreht, so ist der erwartete Effekt, dass der Motor anspringt. Aber, jeder kennt das, es gibt Ausnahmesituationen, in denen dieser Effekt nicht eintritt: der Tank kann ausgelaufen sein, die Batterie leer, die Zündkerzen defekt, der Motor ausgebaut, usw. Wieder ist die Liste der möglichen Ausnahmen beliebig verlängerbar. Keiner von uns wird wohl jedesmal nachsehen, ob der Motor noch da ist, bevor er zu starten versucht. Unsere Handlungen basieren auf der Erwartung, dass sich die Dinge wie üblich verhalten. Wieder ist es eine natürliche Idee, Defaults zu verwenden, um die erwarteten Effekte von Ereignissen formal zu beschreiben.

Es gibt zahlreiche weitere Gebiete, in denen Nichtmonotonie eine wichtige Rolle spielt. Bei der Diagnose etwa können Defaults verwendet werden, um das normale Verhalten von Teilen eines Gerätes zu beschreiben. Fügt man zu dieser Beschreibung des normalen Verhaltens die Beschreibung des tatsächlich beobachteten Verhaltens hinzu, so geben die nicht angewendeten Defaults Auskunft darüber, welche Bauteile defekt sein können. Beim Sprachverständigen verwenden wir ständig Defaults, etwa um Mehrdeutigkeiten zu beheben. Beim Bilderkennen werden Defaults benutzt, um aus Teilszenen vollständige Bilder zu erzeugen. Selbst das Gesetz ist nichtmonoton, wie folgende Regeln zeigen, die in dieser oder ähnlicher Form tatsächlich in Gesetzesstexten zu finden sind [25]:

Verträge sind gültig.

Verträge mit Minderjährigen sind ungültig.

Verträge mit Minderjährigen, die im Beisein eines Vormundes geschlossen werden, sind gültig.

Es ist leicht zu sehen, dass in all diesen Fällen zusätzliche Information zur Rücknahme von Schlüssen führen kann.

Nichtmonotone Systeme haben eine recht lange Tradition in der Künstlichen Intelligenz. Schon sehr lange werden etwa Frame-Systeme verwendet, wie sie in fast allen Expertensystemwerkzeugen zu finden sind. Diese Systeme ermöglichen die Beschreibung von Klassenhierarchien und typischen Eigenschaften von Instanzen dieser Klassen. Frames beschreiben Klassen, Slots repräsentieren Attribute der Instanzen dieser Klassen und ihre typischen Werte. Die grundlegende Idee ist die, dass im Falle von Widersprüchen die spezifischste Information vorgezogen wird. Betrachten wir folgendes Beispiel (die hier für Frame- und Instanzdefinitionen verwendete Sprache dürfte selbsterklärend sein):

```
(defframe auto
  (slots  (sitze 5)
          (zylinder 4)
          (raeder 4)))

(defframe sportwagen
  (supers auto)
  (slots  (sitze 2)
          (preis hoch)))

(definstance speedy of sportwagen)
```

Aus diesen Definitionen lässt sich ableiten, dass `speedy` 2 Sitze und 4 Zylinder hat. Der aus der Definition von `auto` stammende Wert 5 für `sitze` wird nicht berücksichtigt, da `auto` eine Oberklasse von `sportwagen` ist, wie es durch die entsprechende `supers`-Spezifikation festgelegt wurde. Wir erweitern nun die zweite Definition folgendermaßen:

```
(defframe sportwagen
  (supers auto)
  (slots  (sitze 2)
          (preis hoch)
          (zylinder 6)))
```

Jetzt wird für `zylinder` der Wert 6 abgeleitet, d.h. zusätzliche Information hat zur Revision einer früheren Ableitung geführt.

Ein anderes bekanntes nichtmonotonen System ist Prolog mit seiner Behandlung der Negation (negation as failure). Dabei wird *not A* als gültig betrachtet, wenn *A* nicht abgeleitet werden kann. Betrachten wir wieder unser altes Tweety-Beispiel:²

```
Fliegt(X) :- Vogel(X), not Abnormal(X).
Abnormal(X) :- Pinguin(X).
Vogel(Tweety).
```

Die Prolog-Anfrage `Fliegt (Tweety) ?` ist erfolgreich, da `not Abnormal (Tweety)` abgeleitet werden kann. Fügen wir jedoch die Prämisse

```
Pinguin(Tweety).
```

hinzu, so kann `Fliegt (Tweety)` nicht mehr abgeleitet werden. Auch hier also ein Fall von Nichtmonotonie.

Die vor allem aus dem Bereich der deduktiven Datenbanken bekannte Annahme der Weltabschlossenheit (Closed World Assumption, CWA) ist in engem Zusammenhang hierzu zu sehen.

² Abweichend von unserer üblichen Notation verwendet Prolog Großbuchstaben für Variablen.

Ahnlich wie in Prolog geht man davon aus, dass nur positive Information explizit gespeichert werden muss. Fehlen einer Information wird als Falschheit dieser Information gedeutet. Wenn etwa in einer Datenbank über Flugverbindungen keine Direktverbindung zwischen Bonn und London enthalten ist, so wird das so interpretiert, dass es eben keine solche Verbindung gibt. Formal lässt sich die CWA folgendermaßen präzisieren:

Definition 6.1.1. Sei T eine Menge von Formeln. p ist ableitbar aus T unter der CWA genau dann wenn

$$T \cup ASS(T) \vdash p$$

wobei $ASS(T) := \{\neg q \mid q \text{ ist atomar und es gilt } T \not\vdash q\}$.

Wie diese drei Beispiele zeigen, spielen nichtmonotone Systeme schon lange eine wichtige Rolle in der Künstlichen Intelligenz und, allgemeiner, in der Informatik. Allerdings sind die Formen der Nichtmonotonie, die wir hier vorgestellt haben, äußerst beschränkt und lassen sich nicht einfach verallgemeinern. So führt etwa die CWA zu Inkonsistenz, wenn Disjunktionen in den Prämissen zugelassen werden. Seit Ende der achtziger Jahre hat sich das nichtmonotone Schließen als eigenständiges Forschungsgebiet der KI etabliert. Vorrangiges Ziel war zunächst, geeignete Formalisierungen zu finden, die mindestens so ausdrucksstark sind wie die klassische Logik. Einige wichtige Ergebnisse dieser Bemühungen werden im folgenden Abschnitt dargestellt.

6.2 Formalisierungen nichtmonotonen Schließens

Bevor wir in diesem Abschnitt einige der wichtigsten nichtmonotonen Logiken darstellen, wollen wir kurz ein grundlegendes Problem diskutieren, das jede Formalisierung technisch schwierig macht: das Problem sich widersprechender Defaults. Hier das Standardbeispiel:

Quäker sind Pazifisten.

Republikaner sind keine Pazifisten.

Nixon ist Quäker und Republikaner.

Ist Nixon Pazifist oder nicht? Es gibt zwei Defaults, die jeweils benutzt werden können, um sich widersprechende Konklusionen abzuleiten. Die Defaults sind für sich genommen völlig korrekt, sie können aber natürlich nicht beide gleichzeitig verwendet werden. Es scheint in diesem Fall unterschiedliche Mengen von Überzeugungen zu geben, die durch die Defaults gleichermaßen gestützt werden. Wie wir sehen werden, tragen einige Ansätze dem dadurch Rechnung, dass sie verschiedene Formelmengen, sogenannte Extensionen, generieren, in denen jeweils eine maximale Menge von Defaults angewendet wurde. Damit stellt sich natürlich die Frage, was eigentlich die ableitbaren Formeln sind. Es gibt zwei Möglichkeiten: einer skeptischen Sichtweise entspricht es, nur das zu glauben, was in allen Extensionen gilt. Diese Sichtweise ist, wie wir sehen werden, implizit in McCarthys Circumscription. Reiter dagegen vertritt die Sichtweise, dass jede der erzeugten Extensionen für sich genommen als akzeptable Menge von Überzeugungen eines Agenten aufgefaßt werden kann.

Zu den wichtigsten Ansätzen der Formalisierung nichtmonotonen Schließens gehören:

- Die Default-Logik (Reiter). Nichtklassische Inferenzregeln werden verwendet, um Defaults darzustellen. Die Extensionen werden als Fixpunkte eines Operators definiert. Der Operator garantiert, dass in einer Extension möglichst viele Defaults angewendet wurden.
- Der modale Ansatz (McDermott und Doyle, Moore). Hier drückt ein Modaloperator explizit aus, ob etwas geglaubt wird oder konsistent ist. Wiederum wird ein Fixpunktoperator verwendet, um die Extensionen zu definieren.
- Zirkumskription (McCarthy, Lifschitz). Hier wird Folgerbarkeit nicht, wie üblich, als Gültigkeit in allen, sondern als Gültigkeit in bestimmten, bevorzugten Modellen der Prämisse definiert. Syntaktisch werden die ‚uninteressanten‘ Modelle durch Hinzufügen eines Formelschemas bzw. einer Formel 2. Stufe eliminiert.
- Konditionale Ansätze (Delgrande, Kraus/Lehmann/Magidor, Boutilier). Sie verwenden ein Konditional \rightarrow , also einen nicht-wahrheitsfunktionalen Junktor, um Defaults in der logischen Sprache zu repräsentieren. Eine Formel q folgt nichtmonoton aus einer Formel p , wenn das Konditional $p \rightarrow q$ in dem entsprechenden Ansatz ableitbar ist.

Wir werden einige dieser Ansätze im folgenden näher erläutern. Dabei werden wir die Default Logik ausführlich beschreiben. Die Darstellung der autoepistemischen Logik sowie der Zirkumskription beschränkt sich auf die wichtigsten Grundideen. Eine Darstellung der konditionalen Ansätze würde den Rahmen dieses Kapitels sprengen. Als Einstieg hierzu empfehlen wir dem Leser [5, 13, 29].

6.2.1 Default Logik

In der Default Logik (DL) [60] werden Default-Regeln als eine Art von nichtklassischen Inferenzregeln repräsentiert. Eine Default-Theorie ist ein Paar (D, W) , wobei W , eine Menge von klassischen prädikatenlogischen Formeln, das sichere Wissen repräsentiert. D ist eine Menge von Defaults der Form

$$\frac{A : B_1, \dots, B_n}{C}$$

wobei A , B_i und C klassische Formeln sind (als alternative Notation wird häufig auch $A : B_1, \dots, B_n / C$ verwendet). Diese Default-Regel ist intuitiv in folgendem Sinne zu lesen: wenn A ableitbar ist und, für alle i ($1 \leq i \leq n$), $\neg B_i$ nicht abgeleitet werden kann, dann leite C ab. A heißt Vorbedingung, B_i Konsistenzannahme und C Konsequenz des Defaults. Wir werden im folgenden offene Defaults, d.h. Defaults mit freien Variablen, als Schemata interpretieren, die alle ihre Grundinstanzen repräsentieren.

Die Frage ist nun: gegeben eine Default-Theorie (D, W) , was sind die durch sie induzierten akzeptablen Überzeugungsmengen, die Extensionen genannt werden? Es gibt einige Eigenschaften, die eine solche Menge S erfüllen sollte:

1. sie sollte das sichere Wissen W enthalten,
2. sie sollte im Sinne der klassischen Logik abgeschlossen sein,
3. alle ‚anwendbaren‘ Defaults sollten angewendet worden sein, wobei Anwendbarkeit in Bezug auf S selbst definiert werden muss,
4. sie sollte keine Formel enthalten, die sich nicht aus W zusammen mit den Konsequenzen ‚anwendbarer‘ Defaults in D herleiten lässt.

Die ersten drei Eigenschaften lassen sich direkt formal aufschreiben, etwa in Form der folgenden Definition:

Definition 6.2.1. Sei (D, W) eine Default-Theorie. S ist bezüglich (D, W) abgeschlossen gdw.

1. $W \subseteq S$,
2. $\text{Th}(S) = S$
3. falls $A:B_1, \dots, B_n/C \in D$, $A \in S$, $\neg B_i \notin S$ ($1 \leq i \leq n$), dann $C \in S$.

Allerdings trägt die Abgeschlossenheit einer Menge von Formeln der vierten gewünschten Eigenschaft noch nicht Rechnung. Eine in der Mathematik und Informatik übliche Art, unerwünschte Elemente aus Mengen auszuschließen, ist die Forderung nach Minimalität von Mengen. Es liegt also nahe, Extensionen als kleinste Mengen zu definieren, die die obige Definition erfüllen.

Leider hilft das in unserem Fall nicht weiter. Betrachten wir die simple Default-Theorie $(\{\text{true}:b/a\}, \emptyset)$. Zwar erfüllt, wie gewünscht, die Menge $S_1 = \text{Th}(\{\{a\}\})$ unsere obige Definition, aber leider auch die Menge $S_2 = \text{Th}(\{\neg b\})$. Es ist leicht zu sehen, dass die drei Eigenschaften von Definition 6.2.1 gelten, außerdem gibt es keine echte Teilmenge von S_2 , die diese Bedingungen erfüllt, d.h. S_2 ist auch minimal. Das zeigt, dass wir unsere vierte gewünschte Bedingung auf diesem Weg nicht erreichen können.

Reiter verwendet deshalb einen Trick. Er testet einen Kandidaten, also eine Menge von Formeln S , auf folgende Weise: S wird überführt in eine Menge $\Gamma(S)$. $\Gamma(S)$ ist die kleinste Menge, die die drei ersten Eigenschaften aus unserer obigen Aufzählung erfüllt, wobei jedoch die Konsistenzbedingungen einer Default-Regel bezüglich S (und nicht $\Gamma(S)$) geprüft werden. S hat den Test bestanden, ist also Extension, wenn es sich bei diesem Test reproduziert, wenn also $\Gamma(S) = S$. Es ist sofort zu sehen, dass nur solche Mengen diesen Test bestehen, die bezüglich (D, W) abgeschlossen sind. Aber auch Eigenschaft 4 muss erfüllt sein: wenn ein Kandidat Formeln enthält, die nicht aus W oder durch Anwendung von Defaults hergeleitet werden können, so werden diese bei dem Test nicht reproduziert, und der falsche Kandidat scheidet aus. In unserem Beispiel etwa ist $\Gamma(S_2) = \text{Th}(\emptyset)$, d.h. wir erhalten die Menge aller Tautologien. Damit ist S_2 keine Extension, da z.B. $\neg b \in S_2$ aber $\neg b \notin \Gamma(S_2)$.

Hier nun die endgültige Fassung von Reiters Definition der Extensionen:

Definition 6.2.2. Sei (D, W) eine Default-Theorie, S eine Menge von Formeln. Wir definieren einen Operator Γ , so dass $\Gamma(S)$ die kleinste Menge ist, für die gilt:

1. $W \subseteq \Gamma(S)$,
2. $\text{Th}(\Gamma(S)) = \Gamma(S)$
3. falls $A:B_1, \dots, B_n/C \in D$, $A \in \Gamma(S)$, $\neg B_i \notin S$ ($1 \leq i \leq n$), dann $C \in S$.

E ist eine Extension von (D, W) genau dann wenn E Fixpunkt von Γ ist, d.h. wenn gilt $\Gamma(E) = E$.

Reiter hat eine äquivalente, quasi-induktive Charakterisierung der Extensionen gegeben. Diese Version wird häufig in Beweisen benutzt und macht sehr deutlich, in welchem Sinne Formeln in den Prämissen begründet sein müssen. Sei E eine Menge von Formeln. Wir definieren für

eine gegebene Default-Theorie (D, W) eine Folge von Formelmengen wie folgt:

$$E_0 = W, \text{ und für } i \geq 0$$

$$E_{i+1} = Th(E_i) \cup \{C \mid A:B_1, \dots, B_n/C \in D, A \in E_i, \neg B_i \notin E\}$$

Reiter hat gezeigt, dass E eine Extension von (D, W) ist, gdw. $E = \bigcup_{i=0}^{\infty} E_i$. Das Vorkommen von E in der Definition von E_{i+1} macht diese Definition nicht-konstruktiv.

Die Tabelle 6.1 stellt einige einfache Beispiele dar. Das letzte Beispiel in der Tabelle zeigt, wie der Konsistenztest in einem Default verwendet werden kann, um Prioritäten zwischen Defaults auszudrücken.

Tabelle 6.1: Einige einfache Beispiele für Defaults.

D	W	Fixpunkte(e)
$\frac{Bird(x) : Flies(x)}{Flies(x)}$	$Bird(Tw)$	$Th(W \cup \{Flies(Tw)\})$
$\frac{Bird(x) : Flies(x)}{Flies(x)}$	$Bird(Tw)$ $Peng(Tw)$ $\forall x. Peng(x) \Rightarrow \neg Flies(x)$	$Th(W)$
$\frac{\begin{array}{c} Bird(x) : Flies(x) \\ Flies(x) \\ Peng(x) : \neg Flies(x) \\ \hline \neg Flies(x) \end{array}}{\neg Flies(x)}$	$Bird(Tw)$ $Peng(Tw)$	$Th(W \cup \{Flies(Tw)\})$ $Th(W \cup \{\neg Flies(Tw)\})$
$\frac{\begin{array}{c} Bird(x) : Flies(x) \wedge \neg Peng(x) \\ Flies(x) \\ Peng(x) : \neg Flies(x) \\ \hline \neg Flies(x) \end{array}}{\neg Flies(x)}$	$Bird(Tw)$ $Peng(Tw)$	$Th(W \cup \{\neg Flies(Tw)\})$

Reiters Default-Logik ist heute sicher eine der prominentesten nichtmonotonen Logiken. Dafür gibt es zwei Gründe. Zum einen ist – trotz der etwas trickreichen technischen Definition der Extensionen – die der Default-Logik zugrundeliegende Idee einfach und intuitiv: verwende Inferenzregeln mit einem zusätzlichen Konsistenztest für die Darstellung von Defaults. Zweitens hat sich herausgestellt, dass die Default-Logik in bestimmter Hinsicht expressiver ist als manch konkurrierender Ansatz, etwa Zirkumskription. Diese zusätzliche Ausdrucksfähigkeit ist nötig, um z.B. die Semantik für logisches Programmieren erfassen zu können.

Durch die Verwendung von Inferenzregeln lassen sich vor allem Probleme mit der Kontraposition von Defaults vermeiden. In der klassischen Logik ist eine Implikation $A \Rightarrow B$ äquivalent zu ihrer Kontraposition $\neg B \Rightarrow \neg A$. Im Zusammenhang mit Defaults ist Kontraposition manchmal unerwünscht. Z.B. ist es sicher richtig, dass Informatiker normalerweise wenig über Nichtmonotonie wissen. Daraus folgt aber nicht, dass jemand, der viel über Nichtmonotonie weiß, normalerweise kein Informatiker ist. Durch die Verwendung von Inferenzregeln zur Darstellung von Defaults lassen sich solche unerwünschten Effekte vermeiden.

Allerdings hat die Verwendung von Inferenzregeln im Sinne von Reiter auch ihre Nachteile. Zunächst kann es vorkommen, dass es gar keine Extension gibt. Man betrachte etwa die Theorie

$$(\{true : \neg A / A\}, \{\}).$$

Keine Menge von Formeln, die A nicht enthält, ist eine Extension, da die Default-Regel nicht angewendet wurde. Aber auch keine Menge S , die A enthält, kann eine Extension sein: wenn S die Formel A enthält, dann wird dadurch bei der Konstruktion von $\Gamma(S)$ die Default-Regel unanwendbar. Damit kann die Formel A nicht mehr in $\Gamma(S)$ enthalten sein und S ist kein Fixpunkt.

Es gibt auch Situationen, in denen intuitiv erwartete Resultate nicht erzielt werden, wie folgendes Beispiel zeigt (da Defaults und Fakten syntaktisch unterschieden werden können, lassen wir in diesem und den folgenden Beispielen D und W implizit):

- 1) *Italiener: Trinkt_Wein/Trinkt_Wein*
- 2) *Franzose: Trinkt_Wein/Trinkt_Wein*
- 3) *Italiener \vee Franzose*

Man würde erwarten, aus dieser Default-Theorie *Trinkt_Wein* ableiten zu können, denn, unabhängig davon, ob die betreffende Person Italiener oder Franzose ist, eine der Default-Regeln sollte anwendbar sein. In der Default-Logik jedoch kann ein Default nur dann angewendet werden, wenn seine Vorbedingung bereits abgeleitet wurde. Default-Logik ist deshalb nicht in der Lage, Fallunterscheidungen adäquat zu behandeln.

Angeregt durch Arbeiten von Gabbay [20], Makinson [40] und anderen [29] ist es in jüngerer Zeit üblich geworden, dieses und ähnliche Probleme in Form von metatheoretischen Eigenschaften nichtmonotoner Inferenzrelationen zu formulieren. Die hier relevante Eigenschaft wird üblicherweise OR genannt. \sim bezeichne eine beliebige (nichtmonotone) Inferenzrelation, d.h. eine Relation mit einer Menge von Formeln auf der linken und einer Formel auf der rechten Seite. OR lässt sich folgendermaßen formulieren:

$$OR : \text{Wenn } X \cup \{y\} \sim a \text{ und } X \cup \{z\} \sim a, \text{ dann } X \cup \{y \vee z\} \sim a.$$

Es stellt sich hier natürlich die Frage, was die der Default-Logik entsprechende Inferenzrelation ist. Um solch eine Relation geeignet zu definieren, wird üblicherweise die Menge von Defaults D festgehalten. Mit anderen Worten, jede Menge D erzeugt ihre eigene Inferenzrelation \vdash_D , die auf folgende Weise definiert werden kann:

Definition 6.2.3. Sei D eine Menge von Reiterschen Defaults, W eine Menge von Formeln erster Ordnung, p eine Formel. Wir definieren $W \vdash_D p$ gdw. p in allen Extensionen von (D, W) enthalten ist.

Unser Weinbeispiel zeigt, dass \vdash_D die Eigenschaft OR verletzt.

Es ist zu bemerken, dass auf Fallunterscheidung basierende Schlüsse möglich sind, wenn wir unsere Defaults in folgender Form repräsentieren:

$$true: \text{Italiener} \Rightarrow \text{Trinkt_Wein}/\text{Italiener} \Rightarrow \text{Trinkt_Wein}$$

Dadurch entstehen aber wiederum Probleme mit der Kontraposition, denn nun kann $\neg \text{Italiener}$ von $\neg \text{Trinkt_Wein}$ abgeleitet werden. Eine bessere Darstellung ist deshalb:

$$\text{true: } \text{Trinkt_Wein/Italiener} \Rightarrow \text{Trinkt_Wein}$$

Jetzt können wir die Default-Regel nicht mehr benutzen, um $\neg \text{Italiener}$ abzuleiten, falls $\neg \text{Trinkt_Wein}$ gegeben ist. Einige mehr implizite Konsequenzen der Kontraposition lassen sich dadurch aber immer noch nicht vermeiden. Wenn z.B.

$$\neg \text{Trinkt_Wein} \vee \neg \text{Trinkt_Bier}$$

gegeben ist, so können wir das Default anwenden und seine Konsequenz benutzen, um $\neg \text{Italiener} \vee \neg \text{Trinkt_Bier}$ abzuleiten. Das ist nicht möglich, wenn die ursprüngliche Default-Regel mit Vorbedingung *Italiener* verwendet wird.

Im Weinbeispiel waren die Konklusionen, die man erhält, zu schwach. Es gibt auch Fälle, in denen Default-Logik zu starke Ableitungen liefert. Folgendes Beispiel stammt von Poole [55]

- 1) $\text{true: Usable}(x) \wedge \neg \text{Broken}(x)/\text{Usable}(x)$
- 2) $\text{Broken}(\text{Left_Arm}) \vee \text{Broken}(\text{Right_Arm})$

Die einzige Extension enthält

$$\text{Usable}(\text{Left_Arm}) \wedge \text{Usable}(\text{Right_Arm})$$

obwohl wir wissen, dass einer der Arme gebrochen ist. Reiters Definition der Extensionen garantiert nur, dass jede einzelne Konsistenzbedingung eines angewendeten Defaults mit der generierten Extension konsistent ist. Die Gesamtmenge aller Konsistenzbedingungen aller angewendeten Defaults muss nicht notwendigerweise mit der Extension konsistent sein. Deshalb erhält man in unserem Beispiel die unerwünschte Ableitung.

Makinson hat gezeigt [40], dass \vdash_D eine weitere wichtige Eigenschaft nichtmonotoner Inferenzrelationen nicht erfüllt, die Kumulativität. Kumulativität besagt, informell, dass die Hinzunahme eines Theorems zu einer Prämisse die Menge der ableitbaren Formeln nicht verändern soll. Formal:

$$\text{Wenn } X \vdash a, \text{ dann } X \cup \{a\} \vdash b \text{ gdw. } X \vdash b.$$

Diese Eigenschaft scheint wesentlich für jede Inferenzrelation zu sein, wenn man Inferenz als Explizitmachen dessen versteht, was implizit in den Prämissen steckt. Denn warum sollte, wenn a implizit in X ist, das Hinzufügen von a zu X irgendeine Auswirkung haben (außer natürlich, dass man beim nächsten Mal nicht mehr so lange nach dem Beweis suchen muss).

Leider erfüllt Default-Logik, genauer \vdash_D , die Kumulativitätseigenschaft nicht, wie Makinsons Gegenbeispiel zeigt. Betrachten wir folgende Menge D :

- 1) $\text{true: } p/p$
- 2) $p \vee q: \neg p/\neg p$

Für $W = \emptyset$ gibt es genau eine Extension, die p und damit $p \vee q$ enthält. Es gilt also $\emptyset \vdash_D p \vee q$ und $\emptyset \vdash_D p$. Wenn jedoch $p \vee q$ als Prämisse in W verwendet wird, dann entsteht eine zweite Extension, die $\neg p$ enthält. Damit ist p nicht mehr in allen Extensionen enthalten, d.h. $\{p \vee q\} \not\vdash_D p$.

Diese Schwierigkeiten haben zu einer Reihe von Modifikationen der Default Logik geführt, von denen wir hier nur einige kurz erwähnen können. Lukaszewicz [39] hat eine Version definiert, die auf einem zweistelligen Fixpunktoperator basiert. Das zweite Argument wird benutzt, um Konsistenzbedingungen der angewendeten Defaults mitzuführen. Ein Default wird nur angewendet, wenn seine Konsistenzbedingung nicht diejenige eines anderen angewendeten Defaults verletzt. Dadurch wird die Existenz von Extensionen garantiert, und Default-Logik wird semimonoton, d.h., durch die Hinzunahme weiterer Defaults können neue Extensionen entstehen, existierende Extensionen werden jedoch nicht zerstört (sie können allerdings größer werden). Das Poolesche Beispiel (broken arms) lässt sich aber immer noch nicht adäquat behandeln.

In [8] wird CDL, eine kumulative Version der Default Logik, vorgestellt. Die Basiselemente dieser Logik sind Assertionen der Form (p, Q) , wobei p eine Formel ist und Q die Menge der Konsistenzbedingungen, die man benötigt, um p abzuleiten. In CDL muss die Gesamtmenge der Konsistenzbedingungen aller angewendeten Defaults einer Extension konsistent sein. Damit lässt sich das Poolesche Beispiel wie von ihm intendiert behandeln.

Schaub und Delgrande haben eine Variante der Default Logik eingeführt, bei der die Konsistenzbedingungen der angewendeten Defaults als Kontext mitgeführt werden. Defaults werden nur dann angewendet, wenn ihre Konsistenzbedingungen in Bezug auf den gesamten Kontext (und nicht nur die ableitbaren Formeln) erfüllt sind. Auch hier führt das Poolesche Beispiel zu den erwarteten Resultaten. Diese und weitere Varianten der Default Logik sind in [14] beschrieben.

Anstatt Reiters Logik zu modifizieren, kann man natürlich auch untersuchen, ob es Spezialfälle gibt, in denen bestimmte Probleme gar nicht erst auftreten. Dabei wird meist die Form der zulässigen Defaults eingeschränkt. In uneingeschränkten Default-Theorien kann ein Default sogar dann anwendbar sein, wenn seine Konsequenz falsch ist. Um das zu vermeiden, können wir fordern, dass die Konsequenz eines jeden Defaults von seiner Konsistenzbedingung impliziert wird (wir nehmen hier an, dass es nur eine Konsistenzbedingung gibt). Solche Defaults heißen *semi-normal* und können in folgender Form notiert werden:

$$A: B \wedge C/C$$

Die Beschränkung auf semi-normale Defaults hat keine großen Auswirkungen auf die Ausdrucksfähigkeit, der Nutzen von nicht semi-normalen Defaults ist sowieso eher fraglich. Andererseits gewinnt man durch diese Einschränkung auch nicht allzuviel: Existenz von Extensionen, OR und Kumulativität sind immer noch nicht erfüllt [16], selbst wenn alle Defaults keine Vorbedingung haben, also von folgender Form sind:

$$\text{true}: B \wedge C/C$$

Eine interessante Klasse von Defaults, die einige wünschenswerte Eigenschaften besitzt, ist die Klasse der *normalen* Defaults. Sie haben folgende Form:

$$A: B/B$$

Reiter hat gezeigt [60], dass normale Default-Theorien immer Extensionen besitzen, außerdem sind sie semi-monoton. Allerdings ist die Ausdrucksmächtigkeit solcher Theorien nicht immer ausreichend. In Tabelle 1 haben wir bereits ein Beispiel dafür gesehen, wie man durch die geeignete Wahl von Konsistenzbedingungen Prioritäten in die Defaults hineinkodieren kann. Prioritäten spielen eine wichtige Rolle in praktischen Anwendungen, denn durch sie lässt sich die Zahl der Extensionen reduzieren.

Der Vorteil der Verwendung eines semi-normalen Defaults $d = A: B \wedge C / C$ für die Repräsentation von Prioritäten besteht darin, dass d blockiert wird, wenn $\neg B$ wahr ist, ohne dass man implizit annehmen muss, dass normalerweise B gilt. Deshalb wird d auch dann blockiert, wenn $\neg B$ selbst eine Default-Konklusion ist. Siehe [8] für eine ausführlichere Diskussion und weitere Beispiele.

Normale Default-Theorien erfüllen immer noch nicht OR und Kumulativität. Diese beiden Eigenschaften gelten aber bei einer weiteren Einschränkung, nämlich dann, wenn nur normale Defaults ohne Vorbedingung verwendet werden [16]. Diese Einschränkung ist allerdings erheblich, und das Resultat unterstreicht sicherlich die Bedeutung der oben erwähnten Modifikationen der Default-Logik.

6.2.2 Autoepistemische Logik

Moores Autoepistemische Logik (AEL) [49] ist das derzeit sicher prominenteste Beispiel für den modalen Ansatz. Moore geht es um die Modellierung eines rationalen Agenten, der in der Lage ist, über seine eigenen Überzeugungen zu reflektieren, und der dabei über vollständige introspektive Fähigkeiten verfügt, d.h. er weiß genau, was er weiß und was er nicht weiß. Zu diesem Zweck wird ein modaler Operator L eingeführt. Lp steht für *es wird geglaubt, dass p*. Die bekannte Vogelregel wird folgendermaßen repräsentiert:

$$\text{Vogel}(x) \wedge \neg L \neg \text{Fliegt}(x) \Rightarrow \text{Fliegt}(x)$$

Es stellt sich nun die Frage, was die Mengen von Überzeugungen sind, die ein rationaler Agent auf der Basis einer gegebenen Menge von Prämissen annehmen sollte. Moore definiert diese Überzeugungsmengen, bei ihm heißen sie Expansionen einer Menge von Prämissen A , folgendermaßen:

Definition 6.2.4. T ist eine Expansion von A genau dann, wenn

$$T = \{p \mid A \cup \text{Bel}(T) \cup \text{Disbel}(T) \vdash p\}$$

wobei $\text{Bel}(T) = \{Lq \mid q \in T\}$, und $\text{Disbel}(T) = \{\neg Lq \mid q \notin T\}$.

Expansionen sind also Mengen von Formeln, die deduktiv abgeschlossen sind und Lp enthalten gdw. p enthalten ist, und $\neg Lp$ gdw. p nicht enthalten ist. Die Theorie

$$\begin{aligned} & \text{Vogel(Tweety)} \wedge \neg L \neg \text{Fliegt}(Tweety) \Rightarrow \text{Fliegt}(Tweety) \\ & \text{Vogel}(Tweety) \end{aligned}$$

etwa besitzt genau eine Expansion. Da $\neg \text{Fliegt}(Tweety)$ auch bei Hinzunahme einer beliebigen konsistenten Menge von Formeln der Form Lq oder $\neg Lq$ nicht abgeleitet werden kann,

muss

$$\neg L \neg Fliegt(Tweety)$$

in der Expansion enthalten sein, und damit auch *Fliegt(Tweety)*.

Im allgemeinen Fall kann es auch mehrere Expansionen geben. Das ist insbesondere dann der Fall, wenn sich, wie im Nixon-Beispiel, unterschiedliche Regeln gegenseitig widersprechen.

Es konnte inzwischen gezeigt werden, dass AEL und Reiters Default Logik in sehr enger Beziehung zueinander stehen [28, 42]. Konolige verwendet die folgende Übersetzung von Defaults in autoepistemische Formeln:

$$A: B_1, \dots, B_n / C \implies LA \wedge \neg L \neg B_1 \wedge \dots \wedge \neg L \neg B_n \Rightarrow C$$

Da andererseits jede AEL-Formel in eine AEL-Implikation obiger Art transformiert werden kann, kann man diese Übersetzung auch in umgekehrter Richtung, also von AEL zu Default-Logik, vornehmen.

Konolige hat gezeigt, dass bei Verwendung dieser Übersetzung die Extensionen einer Default-Theorie genau dem objektiven Teil, d.h. dem Teil ohne Vorkommen des Modaloperators *L*, einer bestimmten Klasse von AEL-Expansionen entsprechen. Die AEL-Expansionen, für die es keine entsprechenden Default-Logik-Extensionen gibt, entstehen deshalb, weil in AEL Formeln zirkuläre Begründungen haben können. Der Wert dieser Expansionen ist deshalb auch recht fraglich. Die AEL-Theorie $\{Lp \Rightarrow p\}$, z.B., hat zwei Expansionen, von denen eine *Lp* und *p* enthält. Die entsprechende Default-Theorie dagegen besitzt nur eine Extension, in der *p* nicht enthalten ist.

Es sind auch nichtmonotone Systeme vorgeschlagen worden, die zwei unabhängige Modaloperatoren verwenden [33, 37, 64]. Die Logik MKNF von Lifschitz etwa verwendet einen epistemischen Operator *K* und einen Operator *not* für negation as failure (MKNF = minimal knowledge with negation as failure). In diesem System ist es möglich, zu unterscheiden zwischen Anfragen der Form „Gibt es eine Klasse, die John unterrichtet?“, formal:

$$\exists x. unterrichtet(John, x)$$

und der Anfrage „Gibt es eine (bekannte) Klasse, von der man weiß, dass John sie unterrichtet“:

$$\exists x. Kunterrichtet(John, x)$$

Diese Unterscheidung ist wichtig für Datenbanken, die disjunktive Information enthalten, und erweiterte (disjunktive) logische Programme mit echter Negation.

Ein weiterer interessanter modaler Ansatz wurde von Levesque vorgeschlagen [31] und von Lakemeyer weitergeführt [30]. Levesque definiert eine monotone Modallogik, die zusätzlich zu *L* den Modaloperator *O* enthält. *Op* steht für „*p* ist alles, was gewußt wird“. Die grundlegende Idee ist die folgende: um zu bestimmen, ob beispielsweise „Tweety fliegt“ aus „Vögel fliegen typischerweise“ und „Tweety ist ein Vogel“ abgeleitet werden kann, muss überprüft werden, ob folgende Formel in der Logik gültig ist:

$$O[(Vogel(Tw) \wedge \neg L \neg Fliegt(Tw) \Rightarrow Fliegt(Tw)) \wedge Vogel(Tw)] \Rightarrow LFliegt(Tw)$$

In diesem Ansatz wird die Nichtmonotonie sozusagen vollständig in den Bereich des Operators O geschoben: $Op \Rightarrow Lq$ kann gültig sein, $O(p \wedge r) \Rightarrow Lq$ jedoch ungültig. Es stellt sich heraus, dass O eine intuitive Semantik besitzt, die auf dem Konzept möglicher Welten (possible worlds) basiert. Die AEL-Extensionen einer Formel p entsprechen genau den Interpretationen, die Op erfüllen.

6.2.3 Zirkumskription

Zirkumskription ist eine Technik, die es uns erlaubt, die Extension bestimmter ausgewählter Prädikate zu minimieren. Die Grundidee ist die folgende: es wird eine Präferenzrelation auf den Modellen einer Prämissemenge definiert. Diese Präferenzrelation bevorzugt Modelle, in denen die ausgewählten Prädikate eine möglichst kleine Extension besitzen. Folgerbarkeit wird dann nicht, wie üblich, als Gültigkeit in allen Modellen definiert, sondern als Gültigkeit in den bezüglich dieser Präferenzrelation besten (minimalen) Modellen.

Syntaktisch lässt sich diese semantische Idee auf folgende Weise umsetzen: die nichtmonotonen Theoreme einer (endlichen) Prämissemenge T werden definiert als die monotonen Theoreme von T plus gewissen zusätzlichen Formeln. Bei den zusätzlichen Formeln handelt es sich in der einfachsten Form der Zirkumskription um alle Instanzen eines bestimmten Formelschemas, bei anderen Varianten um eine Formel 2. Stufe. Diese Formeln dienen dazu, all die Modelle von T zu „eliminieren“, in denen die Extension des zu minimierenden Prädikates nicht minimal ist.

Eine beträchtliche Anzahl von Varianten der Zirkumskription sind definiert worden. Wir werden uns hier auf die einfachste Form, die Prädikaten-Zirkumskription, beschränken [46, 47]. Sie ist folgendermaßen definiert:

Definition 6.2.5. Sei T die Konjunktion einer endlichen Menge von logischen Formeln, die das Prädikatensymbol P enthalten. $T(\phi)$ entstehe aus T durch Ersetzen jedes Vorkommens von P durch den Parameter ϕ .

Die Prädikaten-Zirkumskription von P in T ist das Schema

$$T(\phi) \wedge (\forall x. \phi(x) \Rightarrow P(x)) \Rightarrow (\forall x. P(x) \Rightarrow \phi(x)).$$

x steht hier für einen Variablenvektor entsprechend der Stelligkeit von P . Das Schema besagt intuitiv folgendes: wenn ϕ ein Prädikat ist, das alle Eigenschaften erfüllt, die in T für P festgelegt sind, so kann ϕ nicht für weniger Objekte gelten als P . Mit anderen Worten: P ist das kleinste Prädikat, das die in T für P festgelegten Eigenschaften besitzt.

Alle Instanzen dieses Schemas können zusammen mit den ursprünglichen Prämissen für Ableitungen benutzt werden. Für ϕ werden dabei Prädikatsausdrücke entsprechender Stelligkeit substituiert, das sind Lambda-Ausdrücke der Form $\lambda x_1, \dots, x_n. F$, wobei F eine offene Formel ist und die x_i Variablen. Für diese Variablen werden bei der Substitution in F die jeweiligen Argumente von ϕ eingesetzt. Hier ist ein einfaches Beispiel [46]:

$$T = \text{Block}(A) \wedge \text{Block}(B) \wedge \text{Block}(C)$$

Prädikaten-Zirkumskription von Block in T ergibt das Schema:

$$\phi(A) \wedge \phi(B) \wedge \phi(C) \wedge (\forall x. \phi(x) \Rightarrow \text{Block}(x)) \Rightarrow (\forall x. \text{Block}(x) \Rightarrow \phi(x))$$

Substitution von $\lambda x.(x = A \vee x = B \vee x = C)$ für ϕ und Anwendung auf die jeweiligen Argumente ergibt:

$$\begin{aligned} & (A = A \vee A = B \vee A = C) \wedge \\ & (B = A \vee B = B \vee B = C) \wedge \\ & (C = A \vee C = B \vee C = C) \wedge \\ & (\forall x.(x = A \vee x = B \vee x = C) \Rightarrow \text{Block}(x)) \\ \Rightarrow & (\forall x.\text{Block}(x) \Rightarrow (x = A \vee x = B \vee x = C)) \end{aligned}$$

Die Vorbedingung dieser Implikation ist wahr in T und wir können ableiten, dass A , B und C die einzigen existierenden Blöcke sind. Es ist nicht schwer zu sehen, wie das Schema sich verändert, wenn die ursprüngliche Menge von Prämissen sich ändert (wenn wir etwa $\text{Block}(D)$ hinzufügen). Dadurch wird eine andere Substitution erforderlich, um die Vorbedingung der Implikation wahr zu machen und damit eine explizite Definition von Block abzuleiten. Auf diese Weise entsteht die Nichtmonotonie der Zirkumskription.

Die Präferenzrelation $<_P$ auf Modellen, die diesem Zirkumskriptionschema entspricht, ist die folgende:

Definition 6.2.6. Sei T eine Formel, M_1 und M_2 jeweils ein Modell von T . $M_1 <_P M_2$ gdw.

1. M_1 und M_2 haben denselben Individuenbereich (domain),
2. die Extension aller Prädikate außer P ist in M_1 dieselbe wie in M_2 ,
3. die Extension von P in M_1 ist eine echte Teilmenge der Extension von P in M_2 .

Wie anfangs erwähnt, können wir nun Folgerbarkeit als Gültigkeit in allen bezüglich $<_P$ minimalen Modellen definieren. McCarthy hat gezeigt, dass sein Schema für diesen Folgerbarkeitsbegriff korrekt ist. Für Vollständigkeit ist es erforderlich, statt des Schemas eine entsprechende Formel der Prädikatenlogik zweiter Stufe zu verwenden, in der über Prädikate quantifiziert werden kann.

McCarthy schlägt als allgemeines Prinzip für Default-Schließen mit Zirkumskription die Verwendung sogenannter Abnormalitätsprädikate vor. Default-Regeln wie unser Vogelbeispiel werden dabei folgendermaßen repräsentiert:

$$\forall x.\text{Vogel}(x) \wedge \neg \text{Ab}_1(x) \Rightarrow \text{Fliegt}(x)$$

Die intuitive Bedeutung der Regel ist: Vögel, die nicht abnormal sind, fliegen. Natürlich können Objekte in mancher Hinsicht normal, in anderer abnormal sein. Deshalb wird für jedes Default ein eigenes Abnormalitätsprädikat benötigt (daher der Index von Ab_1). Zirkumskription wird benutzt, um die Extension der Ab -Prädikate zu minimieren.

Leider hat sich herausgestellt, dass komplexere Versionen der Zirkumskription als die vorgestellte nötig sind, um adäquate Resultate zu erzielen. So müssen etwa bestimmte Prädikate bei der Minimierung variieren können. Des weiteren ist die Version 2. Stufe, bei der das Schema durch eine Formel zweiter Stufe ersetzt wird, aus theoretischen Gründen vorzuziehen. Einen lesenswerten Überblick gibt [34].

6.3 Default-Schließen als Behandlung von Inkonsistenz

Wie wir in Abschnitt 6.2 gesehen haben, gehen die Standardansätze zur Formalisierung nicht-monotonen Schließens von einer konsistenten Menge von Prämissen aus (andernfalls erhält man kein sinnvolles Resultat) und erweitern die Inferenzrelation, so dass mehr als nur die klassischen Theoreme ableitbar werden. Der im folgenden dargestellte Ansatz basiert auf einer grundsätzlich anderen Sichtweise. Was macht Default-Wissen zu Default-Wissen? Was unterscheidet es von sicheren Fakten? Sicherlich die Art und Weise, wie wir es im Falle von auftretenden Konflikten, d.h. Inkonsistenzen, behandeln. Wenn wir diese Sichtweise ernst nehmen, dann liegt die Idee nahe, Default-Schließen als einen speziellen Fall von Inkonsistenzbehandlung aufzufassen. Inkonsistente Prämissen stellen kein Problem dar, wenn wir Möglichkeiten vorsehen, mit Inkonsistenzen auf adäquate Weise umzugehen. Eine Voraussetzung dazu ist, die Inferenzrelationen so zu modifizieren, dass im Falle einer Inkonsistenz weniger, d.h. nicht alle, Formeln ableitbar werden.

In diesem Abschnitt werden wir zunächst einen allgemeinen Rahmen zur Definition nichtmonotoner Systeme vorstellen, der auf der oben diskutierten Sichtweise beruht. Wir werden dann zwei konkrete Instanziierungen dieses Rahmens untersuchen. Zuerst zeigen wir, dass Poole's Ansatz zum Default-Schließen [54] eine einfache Instanz unseres Rahmens ist. Allerdings gibt es in seinem Ansatz gewisse Beschränkungen, die darauf beruhen, dass Prioritäten zwischen Defaults nicht adäquat ausgedrückt werden können. Abschnitt 6.3.3 stellt eine Generalisierung von Poole's System vor, bei der verschiedene Stufen von möglichen Hypothesen eingeführt werden, die unterschiedliche Grade der Zuverlässigkeit dieser Hypothesen repräsentieren. Eine Formel ist beweisbar aus einer Prämissenmenge, wenn es möglich ist, ein konsistentes Argument für sie aus den zuverlässigsten verfügbaren Prämissen zu konstruieren.

6.3.1 Ein Rahmen für nichtmonotone Systeme

Eine übliche Methode zur Behandlung inkonsistenter Prämissenmengen betrachtet maximal konsistente Teilmengen der Prämissen. Da es im Allgemeinen mehr als eine solche maximal konsistente Teilmenge geben kann, wird Beweisbarkeit definiert als Ableitbarkeit aus all diesen Mengen. Die Idee hinter der Maximalitätsforderung ist klar: die verfügbare Information soll so wenig wie möglich modifiziert werden. Das Konzept der maximal konsistenten Teilmenge für sich genommen erlaubt allerdings noch nicht auszudrücken, dass z.B. *Tweety fliegt* aufgegeben werden soll und nicht *Tweety ist ein Pinguin*, wenn wir wissen, dass Pinguine nicht fliegen.

Um solche Präferenzen ausdrücken zu können, dürfen wir nicht *alle* maximal konsistenten Teilmengen betrachten, sondern nur bestimmte, die *bevorzugten* maximal konsistenten Teilmengen, oder einfacher: die *bevorzugten Teiltheorien* (ähnlich wie bei Zirkumskription Folgerbarkeit nicht als Gültigkeit in allen Modellen, sondern als Gültigkeit in bestimmten, bevorzugten Modellen definiert wird).

Der Begriff der bevorzugten Teiltheorie geht zurück auf [61]. Rescher hat eine spezielle Ordnung der maximal konsistenten Teilmengen für hypothetisches Schließen eingeführt, diese Idee jedoch nicht auf Default-Schließen angewendet.

Wir sind jetzt in der Lage, einen schwachen und einen strengen Ableitbarkeitsbegriff zu definieren:

Definition 6.3.1. Eine Formel p ist schwach beweisbar aus einer Prämisse T gdw. es eine bevorzugte Teiltheorie S von T gibt, so dass $S \vdash p$.

Definition 6.3.2. Eine Formel p ist streng beweisbar aus einer Formelmenge T gdw. für alle bevorzugten Teiltheorien S von T gilt, dass $S \vdash p$.

„ \vdash “ bezeichnet hier wieder die klassische Ableitbarkeitsrelation. Schwache und strenge Beweisbarkeit entsprechen, grob gesprochen, dem Enthaltensein in mindestens einer bzw. allen Extensionen in den Fixpunktansätzen zum Default-Schließen, wie etwa Default-Logik. Wir können auch den Begriff Extension einführen, und zwar auf folgende Weise:

Definition 6.3.3. E ist eine Extension einer Formelmenge T gdw. es eine bevorzugte Teiltheorie S von T gibt, so dass $E = Th(S)$.

$Th(A)$ bezeichnet hier wie üblich die Menge $\{p \mid A \vdash p\}$. Es bleibt natürlich noch zu spezifizieren, was jeweils die bevorzugten Teiltheorien sind. Zu diesem Zweck werden wir im Rest dieses Abschnittes den Prämissen T eine gewisse Struktur auferlegen. In einem Ansatz etwa (Abschnitt 6.3.3) werden wir T in verschiedene Levels T_1, \dots, T_n aufspalten. Diese zusätzliche Struktur wird benutzt, um die bevorzugten Teiltheorien von T zu definieren. Der Einfachheit halber werden wir häufig auch von bevorzugten Teiltheorien dieser Strukturen sprechen und die Prämissenmenge T implizit lassen.

Einen wichtigen Aspekt wollen wir hier schon diskutieren, bevor wir auf verschiedene Möglichkeiten eingehen, die bevorzugten Teiltheorien zu definieren: die beweisbaren Formeln hängen in unserem Ansatz von der syntaktischen Form der Prämissen ab. Es ist ein entscheidender Unterschied, ob z.B. eine Prämissenmenge sowohl A wie auch B enthält, oder die äquivalente einzelne Formel $A \wedge B$. Angenommen es gibt eine bevorzugte Teiltheorie S , die mit A inkonsistent ist, aber nicht mit B . Wenn beide Formeln als Prämissen gegeben sind, dann muss B in S enthalten sein. Ist jedoch die Prämisse $A \wedge B$, dann ist das nicht der Fall.

Syntaxabhängigkeit wird von Logikern häufig als Mangel empfunden. Prinzipiell wäre es möglich, durch die Einführung einer eindeutigen Normalform für Formeln dieses Problem zu umgehen. Allerdings betrachten wir in diesem Fall die Syntaxabhängigkeit gerade nicht als Nachteil. Im Gegenteil, sie erhöht die Ausdrucksmächtigkeit des Formalismus. Wir können ausdrücken, ob zwei Formeln A und B zusammen akzeptiert oder verworfen werden sollen oder nicht. Es ist durchaus sinnvoll, zwischen solchen Situationen zu unterscheiden, in denen A und B mögliche, unabhängige Hypothesen sind, und solchen, in denen $A \wedge B$ eine Hypothese darstellt.

Ein kleines Beispiel: nehmen wir an, jemand erzählt uns, er habe Uli auf dem Kaiserplatz Eis essen sehen. Nennen wir „Uli war auf dem Kaiserplatz“ A und „Uli hat Eis gegessen“ B . Erhalten wir später die zuverlässigere, mit A inkonsistente Information C : „Uli war in der Uni“, so macht es durchaus Sinn, auch B aufzugeben, obwohl B nicht mit C inkonsistent ist. In unserem Ansatz erreicht man das gerade dadurch, dass $A \wedge B$ als eine einzelne Hypothese aufgefasst wird.

Ersetzt man eine einzelne Prämisse P durch eine logisch äquivalente einzelne Prämisse Q , so hat das keine Auswirkung auf die erzeugten Extensionen.

Wir haben bisher einen allgemeinen Rahmen für nichtmonotone Systeme definiert. Um eine spezifische Instanz, also ein konkretes nichtmonotonen System, zu erhalten, muss noch definiert werden, was die bevorzugten Teiltheorien sind. Wir werden zunächst zeigen, dass Poole's Ansatz als solch eine Instanz aufgefaßt werden kann.

6.3.2 Poole's System

Poole [54] hat einen einfachen, eleganten und doch recht ausdrucksmächtigen Ansatz zum Default-Schließen entwickelt. In Poole's System definiert der Benutzer zwei Arten von Prämisen:

1. eine konsistente Menge F von (geschlossenen) Formeln³, die Fakten,
2. eine Menge Δ von, möglicherweise offenen, Formeln, die möglichen Hypothesen.

Definition 6.3.4. Ein Szenario von F und Δ ist eine Menge $D \cup F$, wobei D eine Menge von Grundinstanzen von Formeln aus Δ ist, so dass $D \cup F$ konsistent ist.

Definition 6.3.5. Eine Formel g ist erklärbar aus F und Δ gdw. es ein Szenario von F und Δ gibt, in dem g abgeleitet werden kann.

Definition 6.3.6. Eine Extension von F und Δ ist die Menge der logischen Konsequenzen eines maximalen Szenarios von F und Δ .

Poole's Terminologie spiegelt wieder, dass dieser Ansatz nicht nur für Vorhersagen, sondern auch für die Modellierung von Erklärungen von beobachteten Fakten verwendet werden kann. Da unser Hauptinteresse in diesem Kapitel auf Default-Schließen gerichtet ist, werden wir Δ auch einfach die Menge der Defaults nennen.

Es ist nicht schwierig zu sehen, dass Poole's Ansatz eine spezielle Instanz des Ansatzes der bevorzugten Teiltheorien darstellt. Wenn wir die bevorzugten Teiltheorien von $\Delta' \cup F$ (Δ' ergibt sich aus Δ durch Ersetzen der offenen Formeln durch alle ihre Grundinstanzen) gerade als diejenigen Teiltheorien definieren, die F enthalten, dann fallen – unter der Voraussetzung der Konsistenz von F – schwache Beweisbarkeit und Poole's Erklärbarkeit zusammen.

Hier ein kleines Beispiel, diesmal nicht aus dem Bereich der Ornithologie, das gleichzeitig eine Schwierigkeit mit Poole's Ansatz verdeutlicht. Δ besteht aus den offenen Formeln:

- (1) $verletzt(x, y) \Rightarrow schuldig(x)$
- (2) $greift_an(x, y) \Rightarrow notwehr(y)$
- (3) $notwehr(x) \Rightarrow \neg schuldig(x)$

F besteht aus folgenden Fakten:

- (4) $greift_an(Hans, Peter)$
- (5) $verletzt(Peter, Hans)$

³ Eine Formel ist geschlossen, wenn sie keine ungebundenen Variablen enthält, sonst offen.

Jetzt ist erklärbar $\neg schuldig(Peter)$, aber auch $schuldig(Peter)$ und damit ebenso $\neg notwehr(Peter)$. Der Grund ist, dass wir (1) verwenden können, um (2) und (3) zu blockieren. Was in obigem Beispiel fehlt, ist die Festlegung von geeigneten Prioritäten zwischen den Defaults.

Poole hat gezeigt, wie einige dieser Probleme in seinem System gelöst werden können. Er benutzt dabei benannte Defaults: für eine Hypothese $w(x) \in \Delta$ mit freien Variablen x wird ein neues Prädikatsymbol p_w derselben Stelligkeit eingeführt. Poole zeigt, dass $w(x)$ äquivalent durch $p_w(x)$ ersetzt werden kann, wenn die Formel

$$\forall x. p_w(x) \Rightarrow w(x)$$

zu F hinzugefügt wird. Poole benutzt dafür die Notation $p_w(x) : w(x)$ als abkürzende Schreibweise. Das zeigt, dass die Mächtigkeit des Systems nicht beeinträchtigt wird, wenn man nur atomare Formeln in Δ zulässt.

Die Benutzung von Namen erlaubt es, die Anwendbarkeit von Defaults bei Bedarf zu blockieren. Wenn ein Default $p_w(x)$ in einer Situation s nicht anwendbar sein soll, so können wir einfach $\forall x.s \Rightarrow \neg p_w(x)$ zu den Fakten hinzufügen. [54] enthält zahlreiche überzeugende Beispiele dafür, wie diese Technik verwendet werden kann. Dabei ist der Unterschied zwischen allquantifizierten Formeln und Schemata mit freien Variablen wesentlich: eine allquantifizierte Formel in Δ , etwa $\forall x.P(x) \Rightarrow Q(x)$, wird durch ein Gegenbeispiel in F , etwa $P(a)$ und $\neg Q(a)$, außer Kraft gesetzt. Enthält dagegen Δ die offene Formel $P(x) \Rightarrow Q(x)$, die alle ihre Grundinstanzen repräsentiert, so wird durch das Gegenbeispiel nur die entsprechende Instanz außer Kraft gesetzt.

Unser obiges Beispiel kann mit dieser Technik etwa folgendermaßen dargestellt werden: Δ wird zu der Menge

$$\{d_1(x, y), d_2(x, y), d_3(x)\}$$

und aus F wird

- (1) $\forall x, y. d_1(x, y) \Rightarrow (verletzt(x, y) \Rightarrow schuldig(x))$
- (2) $\forall x, y. d_2(x, y) \Rightarrow (greift_an(x, y) \Rightarrow notwehr(y))$
- (3) $\forall x, y. d_3(x) \Rightarrow (notwehr(x) \Rightarrow \neg schuldig(x))$
- (4) $greift_an(Hans, Peter)$
- (5) $verletzt(Peter, Hans)$
- (6) $\forall x, y. greift_an(x, y) \Rightarrow \neg d_1(y, x)$

Wichtig ist hier insbesondere (6), weil diese Formel explizit die Anwendung von (1) in bestimmten Fällen blockiert. Jetzt ist in unserem Beispiel, wie gewünscht, $schuldig(Peter)$ nicht mehr erklärbar.

Poole hat gezeigt, dass mit dieser Technik sehr viele der Standardbeispiele zum Default-Schließen aus der Literatur adäquat behandelt werden können [54]. Allerdings ist diese Art, Prioritäten zwischen Defaults auszudrücken, in einigen Fällen äußerst mühselig und unpraktisch [7]. Dieser Mangel ist die Motivation für die Verallgemeinerungen, die wir im nächsten Abschnitt beschreiben werden.

Bleibt noch zu sagen, dass mit dem System THEORIST eine recht effiziente Prolog-Implementierung von Pooles Ansatz zur Verfügung steht [56].

6.3.3 Zuverlässigkeitsstufen

Die Grundidee des Pooleschen Ansatzes war es, zwischen genau zwei Typen von Prämissen zu unterscheiden: solchen, die mit Sicherheit gelten (und konsistent sein) müssen (Fakten), und solchen, die weniger zuverlässig sind und möglicherweise nicht gelten (Hypothesen).

Wir verallgemeinern diese Idee in zwei Hinsichten: zum einen fordern wir nicht die Konsistenz der zuverlässigsten Formeln. In unserer Verallgemeinerung ist jede Formel prinzipiell widerlegbar. Zum anderen führen wir mehr als genau zwei Typen von Prämissen ein. Die Prämissen werden in verschiedene Stufen oder Levels eingeteilt. Die Idee dabei ist, dass die verschiedenen Stufen verschiedene Grade der Zuverlässigkeit repräsentieren. Die erste Stufe enthält die zuverlässigste Information, die zweite Stufe die nächst zuverlässige, usw. Wenn Inkonsistenzen auftreten, dann wird jeweils die zuverlässigere Information bevorzugt. Intuitiv gesprochen ist eine Formel dann ableitbar, wenn ein Argument für sie aus der zuverlässigsten verfügbaren Information konstruiert werden kann.

Natürlich kann es dabei widersprüchliche Information derselben Zuverlässigkeit geben. In diesem Fall erhalten wir wieder multiple Extensionen, ähnlich wie in Reiters Default Logik. Die Eigenschaft, dass es keine prinzipiell unwiderlegbaren Prämissen gibt, macht es möglich, alle Stufen auf uniforme Weise zu behandeln. Z.B. können wir stets Information hinzufügen, die zuverlässiger ist als die gegenwärtig erste Stufe.

Diese intuitiven Ideen müssen natürlich im Rahmen unseres Ansatzes präzisiert werden. Das heißt, wir müssen definieren, was die bevorzugten Teiltheorien in diesem Fall sein sollen.

Definition 6.3.7. Eine Default-Theorie T ist ein Tupel (T_1, \dots, T_n) , wobei jedes T_i eine Menge klassischer Formeln erster Ordnung ist.

Information in T_i ist zuverlässiger als solche in T_j , wenn $i < j$. Ein Default wie *Vögel fliegen* kann wiederum repräsentiert werden als die Menge aller Grundinstanzen des Schemas $\text{Vogel}(x) \Rightarrow \text{Fliegt}(x)$. Der Einfachheit halber schreiben wir $T_i = \{\dots, P(x), \dots\}$ um auszudrücken, dass T_i alle Grundinstanzen von $P(x)$ enthält.

Definition 6.3.8. Sei $T = (T_1, \dots, T_n)$ eine Default-Theorie. $S = S_1 \cup \dots \cup S_n$ ist eine bevorzugte Teiltheorie von T gdw. für alle k ($1 \leq k \leq n$) gilt: $S_1 \cup \dots \cup S_k$ ist eine maximal konsistente Teilmenge von $T_1 \cup \dots \cup T_k$.

In anderen Worten: um zu einer bevorzugten Teiltheorie zu gelangen, nimmt man eine beliebige maximal konsistente Teilmenge von T_1 , fügt so viele Formeln von T_2 hinzu, wie das konsistent möglich ist, usw. bis man bei T_n angelangt ist.

Die folgenden einfachen Beispiele zeigen, wie die verschiedenen Stufen benutzt werden können, um Prioritäten zwischen Defaults auszudrücken:

$$\begin{aligned} T1 : & \{ \text{Vogel}(Tweety), \forall x. \text{Pinguin}(x) \Rightarrow \neg \text{Fliegt}(x) \} \\ T2 : & \{ \text{Vogel}(x) \Rightarrow \text{Fliegt}(x) \} \end{aligned}$$

$\text{Fliegt}(\text{Tweety})$ ist streng beweisbar.

$$\begin{aligned} T1 &: \{\text{Vogel}(\text{Tweety}), \forall x. \text{Pinguin}(x) \Rightarrow \neg \text{Fliegt}(x), \text{Pinguin}(\text{Tweety})\} \\ T2 &: \{\text{Vogel}(x) \Rightarrow \text{Fliegt}(x)\} \end{aligned}$$

Jetzt ist $\neg \text{Fliegt}(\text{Tweety})$ streng beweisbar. Das Beispiel illustriert auch noch einmal die wichtige Unterscheidung zwischen Schemata und allquantifizierten Formeln. Würde in $T2$ die entsprechende quantifizierte Formel benutzt, so würde ein einziger nichtfliegender Vogel genügen, um die Regel außer Kraft zu setzen.

Gibt es auch fliegende Pinguine, dann können wir folgende Repräsentation benutzen, die der spezielleren Pinguin-Regel höhere Priorität verleiht als der Vogel-Regel:

$$\begin{aligned} T1 &: \{\text{Vogel}(\text{Tweety}), \text{Pinguin}(\text{Tweety}), \text{Pinguin}(\text{Tim}), \text{Fliegt}(\text{Tim}), \text{Vogel}(\text{Jim})\} \\ T2 &: \{\text{Pinguin}(x) \Rightarrow \neg \text{Fliegt}(x)\} \\ T3 &: \{\text{Vogel}(x) \Rightarrow \text{Fliegt}(x)\} \end{aligned}$$

Jetzt fliegen (streng beweisbar) Tim und Jim , Tweety fliegt nicht.

Hier ist eine Repräsentation unseres Notwehr-Beispiels ohne benannte Defaults.

$$\begin{aligned} T1 &: \{\text{greift_an}(\text{Hans}, \text{Peter}), \text{verletzt}(\text{Peter}, \text{Hans})\} \\ T2 &: \{\text{greift_an}(x, y) \Rightarrow \text{notwehr}(y), \text{notwehr}(x) \Rightarrow \neg \text{schuldig}(x)\} \\ T3 &: \{\text{verletzt}(x, y) \Rightarrow \text{schuldig}(x)\} \end{aligned}$$

Wie gewünscht ist $\neg \text{schuldig}(\text{Peter})$ streng beweisbar.

Bisher haben wir in unseren Beispielen die Formeln selbst in die entsprechenden Stufen einsortiert. Es sind auch Verfahren entwickelt worden, die automatisch für eine vorgegebene Menge von Default-Regeln eine Einsortierung vornehmen und dabei die Spezifizität der jeweiligen Regeln berücksichtigen [29, 52].

Eine Verfeinerung der hier vorgestellten Behandlung von Zuverlässigkeitssstufen wurde von Benferhat und seinen Toulouser Kollegen [4] sowie von Lehmann [38] vorgeschlagen. Dabei wird bei der Definition bevorzugter Teiltheorien zusätzlich die Anzahl der pro Stufe akzeptierten Default-Regeln berücksichtigt. Interessanterweise erfüllt die dadurch definierte strenge Beweisbarkeitsrelation die Eigenschaft der rationalen Monotonie, die besagt, dass Konklusionen erhalten bleiben, solange die Prämissen nicht um unerwartete Formeln erweitert werden. Formal:

$$\text{Wenn } X \vdash a \text{ und nicht } X \vdash \neg b \text{ dann } X \cup \{b\} \vdash a.$$

Manchmal möchte man offen lassen, ob eine Formel p größere, kleinere oder dieselbe Zuverlässigkeit besitzt wie eine andere Formel q . In diesem Fall kann man anstelle der Zuverlässigkeitssstufen auch eine beliebige strikte partielle Ordnung der Prämissen zulassen, die wiederum die relative Zuverlässigkeit der Formeln ausdrückt. Eine Default-Theorie ist dann ein Paar $(T, <)$, wobei die Relation $<$ definiert ist über T und $p < q$ ausdrückt, dass p zuverlässiger ist als q . Eine bevorzugte Teiltheorie erhält man, indem man die Prämissen in einer beliebigen mit $<$ kompatiblen Reihenfolge, beginnend mit den relevantesten, testet und jeweils die Prämissen mit aufnimmt, die konsistent mit den bereits akzeptierten Prämissen sind.

Insgesamt scheint es ein Vorteil des in diesem Abschnitt beschriebenen Ansatzes zu sein, dass er näher an der klassischen Logik ist als viele andere Systeme: wir brauchen keine Modaloperatoren, keine nichtklassischen Inferenzregeln, keine Fixpunktkonstruktionen, keine Logik höherer Ordnung oder Abnormalitätsprädikate. Des weiteren ist das Problem der Behandlung inkonsistenter Information – ein Problem, das jede Modellierung intelligenten Verhaltens sowieso behandeln muss – implizit gelöst.

Andererseits ist der hier vorgestellte Ansatz natürlich in bestimmter Hinsicht weniger ausdrucksfähig als etwa Reiters Default Logik: Defaults in diesem Ansatz entsprechen, wie Poole gezeigt hat, normalen Reiter-Defaults ohne Vorbedingungen. Deshalb wurden Methoden der Behandlung von expliziten Präferenzen zwischen Defaults auch für die Default Logik entwickelt, z.B. in [9]. Diese Arbeit geht noch einen Schritt weiter und zeigt, wie man die Information über die Präferenzen innerhalb der logischen Sprache selbst ausdrücken kann. Damit werden auch die Präferenzen selbst dynamisch ableitbar und Strategien zur Lösung von Konflikten zwischen Defaults können deklarativ beschrieben werden. Das ist für viele Anwendungen äußerst hilfreich, wie das zitierte Papier anhand eines Beispiels aus dem juristischen Schließen illustriert.

6.4 Nichtmonotonie und Logikprogrammierung

In diesem Abschnitt wollen wir die Zusammenhänge zwischen nichtmonotonen Logiken und Logikprogrammierung darstellen. Es stellt sich heraus, dass die Standardsemantiken der Logikprogrammierung in enger Beziehung insbesondere zu Reiters Default Logik stehen. Somit lassen sich Systeme aus der Logikprogrammierung, sofern sie diese Semantiken tatsächlich implementieren, als eingeschränkte Beweissysteme für die Default Logik auffassen.

Tatsächlich sind Einschränkungen bei der Implementierung von Beweissystemen für nichtmonotone Logiken unumgänglich, zumindest wenn es sich um nichtmonotone Erweiterungen der Prädikatenlogik erster Stufe handelt. Diese sind in ihrer allgemeinsten Form nämlich nicht semientscheidbar, d.h. es existiert nachweislich kein korrektes und vollständiges Beweisverfahren, das immer terminiert. Wir müssen also mit bestimmten Einschränkungen leben bzw. auf Vollständigkeit unserer Beweisverfahren verzichten.

Wie bereits in der Einleitung erwähnt, führt die besondere Behandlung der Negation (negation as failure) in der Logikprogrammierung zu Nichtmonotonie. Ursprünglich wurde die Bedeutung dieser Art von Negation prozedural erklärt: wenn Beweiser ein Literal der Form *not L* abzuleiten haben, so wird ein Beweis für *L* gestartet. Gelingt dieser Beweis, so gilt *not L* als falsch. Falls der Beweis nicht gelingt und der Beweiser terminiert, so gilt *not L* als wahr.

Diese Art der Charakterisierung der Bedeutung von Logikprogrammen ist aus zwei Gründen unbefriedigend: 1) die Semantik eines Programmes hängt von einer zugrundeliegenden Beweisprozedur ab (eigentlich sollte eine Semantik ja Maßstab für Korrektheit und Vollständigkeit einer Beweisprozedur sein), 2) die Bedeutung von Programmen, für die der Beweiser nicht terminiert, bleibt offen.

Es sind deshalb eine ganze Reihe von Semantiken für Logikprogramme entwickelt worden, die unabhängig von der zugrundeliegenden Beweisprozedur sind. Die zwei wichtigsten Semanti-

ken, nämlich die Semantik stabiler Modelle und die wohlfundierte Semantik, wollen wir hier vorstellen.

6.4.1 Stabile Modelle

Zunächst müssen wir präzisieren, was unter einem Logikprogramm zu verstehen ist. Im einfachsten Fall bestehen solche Programme aus Regeln der Form

$$c \leftarrow a_1, \dots, a_n, \text{not } b_1, \dots, \text{not } b_m$$

wobei a_i, b_j und c atomare Formeln sind. Wenn in den Regeln Variablen auftauchen, so repräsentieren die Regeln alle ihre Grundinstanzen. Verallgemeinerungen, die Disjunktionen in den Regelköpfen oder zwei Arten von Negation zulassen, werden später kurz diskutiert. Die hier beschriebenen Programme werden auch normale Logikprogramme genannt.

Ein stabiles Modell [24] M ist nun, intuitiv, eine Menge von Grundatomen, die man aus den Regeln eines Programmes ableiten kann, wenn man die Negation in den Regelkörpern in Bezug auf M selbst auswertet. Es begegnet uns also wieder die Selbstbezüglichkeit, die uns schon von Reiters Default Logik bekannt ist. Insgesamt will man wieder erreichen, dass keine anwendbare Regel des Programms „vergessen“ wird, und dass andererseits nur solche Atome in M sind, für die es eine Ableitung auf Basis von anwendbaren Regeln gibt. Hier ist die formale Definition:

Definition 6.4.1. Sei P ein Logikprogramm, M eine Menge von Atomen. Die M -Reduktion von P , P^M , ist das Programm, das man aus P erhält, indem man

1. alle Regeln entfernt, in deren Körper $\text{not } l$ für ein $l \in M$ vorkommt,
2. alle negierten Atome aus den übrigen Regeln entfernt.

M heißt stabiles Modell von P , wenn M die kleinste unter den Regeln in P^M abgeschlossene Menge von Atomen ist.

Die in dieser Definition durchgeführte Reduktion des Programms entspricht gerade der Auswertung von Literalen der Form $\text{not } l$ in Bezug auf M . Bezeichnen wir die kleinste Menge von Atomen, die unter einer Menge R von Regeln ohne Negation abgeschlossen⁴ ist, mit $Cn(R)$, so können wir die stabilen Modelle von P auch als Fixpunkte eines Operators γ_P einführen, der folgendermaßen definiert ist:

$$\gamma_P(X) = Cn(P^X)$$

M ist also stabiles Modell von P gdw. $M = \gamma_P(M)$. Ein Atom wird als ableitbar betrachtet, wenn es in allen stabilen Modellen eines Programms enthalten ist.

⁴ S ist abgeschlossen unter einer Regelmenge R genau dann wenn die Konsequenzen aller Regeln, deren Voraussetzungen in S sind, bereits in S enthalten sind.

Hier ein kleines Beispiel:

$$\begin{aligned} a &\leftarrow d, \text{not } b \\ b &\leftarrow d, \text{not } a \\ c &\leftarrow a \\ c &\leftarrow b \\ d &\leftarrow \text{not } e \end{aligned}$$

Es existieren zwei stabile Modelle, $M_1 = \{a, c, d\}$ sowie $M_2 = \{b, c, d\}$. Wir zeigen, dass M_1 stabiles Modell ist. Dazu bilden wir P^{M_1} . Da $a \in M_1$, entfällt die zweite Regel. Die negierten Literale der übrigen Regeln entfallen ebenso, und wir erhalten:

$$\begin{aligned} a &\leftarrow d \\ c &\leftarrow a \\ c &\leftarrow b \\ d & \end{aligned}$$

Die kleinste unter diesen Regeln abgeschlossene Menge von Atomen ist M_1 , und damit ist M_1 stabiles Modell. Der Nachweis für M_2 ist analog. Da c in allen stabilen Modellen enthalten ist, ist c ableitbar.

Was hat das alles mit Reiters Default Logik zu tun? Die Antwort ist einfach: es handelt sich um einen Spezialfall. Die Regel

$$c \leftarrow a_1, \dots, a_n, \text{not } b_1, \dots, \text{not } b_m$$

entspricht dem Default

$$a_1 \wedge \dots \wedge a_n : \neg b_1, \dots, \neg b_m / c$$

Wir können also jedes Logikprogramm P in eine Default Theorie $Tr(P)$ übersetzen, indem wir die einzelnen Regeln in Defaults übersetzen. Die Menge W bleibt dabei leer. Es gilt nun folgender Zusammenhang:

Proposition 6.4.1. *Sei P ein Logikprogramm, $Tr(P)$ seine Übersetzung in Default Logik. M ist stabiles Modell von P gdw. es eine Extension E von $Tr(P)$ gibt, so dass $E \cap Atoms = M$.*

Hierbei bezeichnet $Atoms$ die Menge der Atome. Die Definition der stabilen Modelle kann also als Spezialisierung der Reiterschen Definition der Extensionen aufgefaßt werden, zugeschnitten auf die einfachere Sprache der Logikprogramme.

6.4.2 Wohlfundierte Semantik

Die ursprüngliche Formulierung der wohlfundierten Semantik (wellfounded semantics) von Gelder, Ross und Schlipf [22] basiert auf einem bestimmten partiellen Modell. Przymusinski rekonstruierte diese Definition in 3-wertiger Logik [57]. Wir verwenden hier die Darstellung von Baral and Subrahmanian [3], die auf dem bereits bekannten Operator γ_P beruht, ihn aber völlig anders benutzt.

Es ist leicht zu sehen, dass dieser Operator anti-monoton ist, d.h. $S \subseteq S'$ impliziert $\gamma_P(S') \subseteq \gamma_P(S)$. Das liegt daran, dass S' bei der Programmreduktion mindestens soviele Regeln aus P eliminieren kann wie S . Die zweifache Hintereinanderausführung von γ_P definiert damit einen monotonen Operator. Nach dem bekannten Knaster-Tarski Theorem [65] besitzt jeder monotone Operator einen kleinsten Fixpunkt. Dieser definiert nun gerade die Menge der ableitbaren Atome.

Definition 6.4.2. Sei P ein Logikprogramm, A ein Atom. A ist wohlfundierte Konklusion von P gdw. A im kleinsten Fixpunkt von γ_P^2 enthalten ist.

Der kleinste Fixpunkt von γ_P^2 kann iterativ erzeugt werden, indem man ausgehend von der leeren Menge γ_P^2 immer wieder anwendet, bis keine neuen Atome mehr hinzukommen. Diese Vorgehensweise verdeutlicht auch die eigentliche Intuition hinter dieser Definition. Nehmen wir an, wir wissen bereits, dass eine Menge von Atomen S ableitbar ist. $\gamma_P(S)$ ist dann die Menge aller Atome, die aufgrund dieser Information überhaupt noch als mögliche Konklusionen in Frage kommen. Wenden wir nun γ_P auf diese Menge an, so können wir sicher sein, dass es sich bei den nun resultierenden Atomen wieder um ableitbare Atome handelt.

Wir wollen das anhand des Beispiel-Programmes aus dem letzten Abschnitt illustrieren, das wir zwecks besserer Lesbarkeit wiederholen:

```
a ← d, not b
b ← d, not a
c ← a
c ← b
d ← not e
```

Wir erhalten folgende Mengen von Atomen:

$\gamma_P(\emptyset) = \{a, b, c, d\}$	keine Regeln werden eliminiert
$\gamma_P^2(\emptyset) = \{d\}$	die ersten beiden Regeln werden eliminiert
$\gamma_P(\{d\}) = \{a, b, c, d\}$	keine Regeln werden eliminiert
$\gamma_P^2(\{d\}) = \{d\}$	kleinster Fixpunkt erreicht

Damit ist d die einzige wohlfundierte Konklusion unseres Programmes. a, b und c sind zwar potentielle Konklusionen, können aber nicht abgeleitet werden.

Es ist kein Zufall, dass die wohlfundierten Konklusionen hier eine Teilmenge der Atome sind, die unter der stabilen Semantik abgeleitet werden können. Das ist in der Tat immer so. Die wohlfundierte Semantik ist also korrekt bzgl. der stabilen Semantik (und damit wegen des Zusammenhangs stabile Semantik/Default Logik auch korrekt bzgl. Default Logik), und wir können sie als Approximation der letzteren Semantik auffassen. Tatsächlich ist das aussagenlogische Ableitbarkeitsproblem (ist ein Atom A aus einem aussagenlogischen Programm P ableitbar?) der wohlfundierten Semantik in quadratischer Zeit lösbar, während es für die stabile Semantik co-NP-vollständig ist. Dieser beachtliche Komplexitätsgewinn hat viel zur Popularität der wohlfundierten Semantik beigetragen.

6.4.3 Antwortmengenprogrammierung

Die syntaktischen Beschränkungen der Logikprogramme, die wir bisher behandelt haben, sind erheblich. Deshalb hat man immer wieder versucht, die Syntax der Regeln zu verallgemeinern. Gegenstand der Forschung waren insbesondere Logikprogramme mit zwei Arten von Negation, in denen neben dem *not* auch das klassische Negationssymbol \neg auftreten kann, sowie Regeln mit Disjunktion in den Köpfen.

Die Einführung einer zweiten, „echten“ Negation ist relativ unproblematisch. Gelfond und Lifschitz [24] haben gezeigt, wie sich stabile Modelle zu sogenannten Antwortmengen verallgemeinern lassen, die zwei Arten von Negation adäquat behandeln. Antwortmengen sind nicht mehr Mengen von Atomen, wie die stabilen Modelle, sondern Mengen von Literalen. Sie entsprechen deshalb nicht zweiwertigen Modellen, was zur Einführung des neuen Namens geführt hat.

Entsprechende Erweiterungen der wohlfundierten Semantik wurden von verschiedenen Autoren vorgeschlagen [2, 10, 35, 53, 58].

Mehr Schwierigkeiten bereiten disjunktive Logikprogramme. Hier gibt es eine ganze Reihe von konkurrierenden Ansätzen, stabile und wohlfundierte Semantik entsprechend zu erweitern. Eine gute Übersicht bietet [6].

Interessanter Weise hat die Einführung von Antwortmengen durch Gelfond und Lifschitz nicht nur zu einer Semantik für erweiterte Logikprogramme geführt, sondern auch zur Entwicklung eines neuen Problemösungs-Paradigmas, der Antwortmengenprogrammierung (answer set programming, ASP). Ausgehend von Arbeiten von Marek/Truszczynski sowie Niemelä [44, 50] konnte gezeigt werden, dass man Logikprogramme auch zur Spezifikation von Problemen verwenden kann, und zwar so, dass die Antwortmengen der Programme den Lösungen des Problems entsprechen.

Die Grundidee lässt sich einfach anhand des Graphfärbeproblems erläutern. Gegeben sei eine Beschreibung eines Graphen, etwa mithilfe der Prädikate *node* und *edge*, also etwa

$$\text{node}(a), \text{node}(b), \dots, \text{edge}(a, b), \dots$$

Jeder Knoten soll mit einer der Farben *red*, *blue*, *green* gefärbt werden:

$$\begin{aligned} \text{col}(X, \text{red}) &\leftarrow \text{node}(X), \text{not col}(X, \text{blue}), \text{not col}(X, \text{green}) \\ \text{col}(X, \text{blue}) &\leftarrow \text{node}(X), \text{not col}(X, \text{red}), \text{not col}(X, \text{green}) \\ \text{col}(X, \text{green}) &\leftarrow \text{node}(X), \text{not col}(X, \text{blue}), \text{not col}(X, \text{red}) \end{aligned}$$

Jetzt fehlt noch die Bedingung, dass Nachbarknoten nicht dieselbe Farbe haben dürfen. Antwortmengen, in denen diese Bedingung verletzt ist, lassen sich durch folgende Regel ausschließen:

$$\text{false} \leftarrow \text{not false}, \text{edge}(X, Y), \text{col}(X, Z), \text{col}(Y, Z)$$

Hier ist *false* ein neues Atom, dass sonst nicht im Programm vorkommt. Der Effekt dieser Regel ist, dass jede Antwortmenge, die zwei Nachbarknoten dieselbe Farbe zuweist, eliminiert wird. Da es nicht darauf ankommt, wie das neue Atom heisst, schreibt man auch einfach:

$$\leftarrow \text{edge}(X, Y), \text{col}(X, Z), \text{col}(Y, Z)$$

Tabelle 6.2: *ASP Instanzierer*

LPARSE	www.tcs.hut.fi/Software/smodels/
DLV	www.dbaï.tuwien.ac.at/proj/dlv/
GRINGO	potassco.sourceforge.net/#gringo/

Tabelle 6.3: *ASP Systeme*

ASSAT	assat.cs.ust.hk/
CLASP	potassco.sourceforge.net/#clasp/
CMODELS	www.cs.utexas.edu/users/tag/cmodels/
DLV	www.dbaï.tuwien.ac.at/proj/dlv/
GNT	www.tcs.hut.fi/Software/gnt/
SMODELS	www.tcs.hut.fi/Software/smodels/
XASP	xsb.sourceforge.net/ , vertrieben gemeinsam mit XSB

Jetzt enthalten alle Antwortmengen eine Farbe für jeden Knoten, und Nachbarknoten haben verschiedene Farben, wie gewünscht.

Diese Vorgehensweise hat sich als äußerst fruchtbar erwiesen, und es gibt inzwischen zahlreiche erfolgreiche Anwendungen dieses Paradigmas. Einen guten Überblick gibt [12].

Für den Erfolg des Paradigmas ist die Verfügbarkeit entsprechender Solver Voraussetzung. Hier ist in den vergangenen Jahren erheblicher Fortschritt erzielt worden. Die Berechnung von Antwortmengen erfolgt typischer Weise in zwei Schritten. Zunächst wird das Programm grundinstanziert, d.h. Variablen werden durch alle möglichen Grundterme ersetzt (Gründung, engl. grounding). Das so entstehende Programm wird dann durch einen propositionalen ASP Solver prozessiert und seine Antwortmengen werden berechnet. Die meisten implementierten ASP Systeme unterscheiden klar zwischen diesen zwei Schritten und stellen unterschiedliche Werkzeuge dafür zur Verfügung.

Tabelle 6.2 gibt einen Überblick über die 3 derzeit am meisten verwendeten Instanzierer. Tabelle 6.3 stellt Links zu einigen der wichtigsten aktuellen ASP Solver zusammen. Es finden seit einiger Zeit regelmäßig Wettbewerbe statt, bei denen CLASP derzeit stets exzellent abschneidet, siehe etwa www.mat.unical.it/aspcmp2011. Interessanter Weise nimmt CLASP seit 2009 auch am SAT Solver Wettbewerb teil und gewann sowohl 2009 wie 2011 zwei Tracks dieses Wettbewerbs – ein überzeugendes Zeichen für die Effizienz dieses Systems.

6.5 Argumentation

Ein Übersichtsartikel zum Thema nichtmonotonen Schließen wäre unvollständig ohne eine Diskussion des Gebietes der (formalen) Argumentation, das sich in den letzten Jahren stark entwickelt hat und zunehmend Aufmerksamkeit auf sich zieht. Grundidee ist es hier, das Konzept des Arguments in den Vordergrund zu stellen und auf diese Weise zu modellieren, wie wir zu Überzeugungen und Entscheidungen kommen. Dabei geht es insgesamt nicht allein um das Akzeptieren von Propositionen, sondern auch darum, wie man etwa Argumentationsprozesse und entsprechende Dialoge geeignet gestalten kann, so dass man zu fairen Gruppenentscheidungen

kommt. Wir werden uns in der kurzen Diskussion hier allerdings auf den Aspekt der Evaluation von Propositionen auf der Basis von Argumenten beschränken, da dieser Aspekt die deutlichste Beziehung zum nichtmonotonen Schließen darstellt. Für einen Überblick über das gesamte Gebiet ist das Buch [59] zu empfehlen.

Ein Argument ist, intuitiv gesprochen, eine Proposition mit einer entsprechenden Begründung. In einem auf klassischer Logik basierenden Ansatz etwa kann man ein Argument als Aussage mit entsprechendem deduktiven Beweis auffassen. Legt man einen Formalismus zugrunde, der auch Default-Regeln kennt, dann können natürlich auch diese in Argumenten verwendet werden. Dung hat in einer höchst einflussreichen Arbeit [18] gezeigt, dass sich die Frage, welche Argumente akzeptiert werden können, unabhängig von der Art und Weise analysieren lässt, wie die Argumente erzeugt werden und welche Struktur sie haben. Dazu hat er die sogenannten abstrakten Argumentationsrahmen (abstract argumentation frameworks, AFs) eingeführt, die wir hier etwas ausführlicher behandeln wollen. Grundsätzlich werden AFs in der Argumentation wie folgt verwendet: ausgehend von einer logischen Sprache, in der das verfügbare Wissen repräsentiert ist, werden Argumente und Konflikte zwischen ihnen (Attacken) berechnet, die zusammen einen Argumentationsrahmen bilden. Die für AFs definierten Semantiken spezifizieren nun, welche Mengen von Argumenten als akzeptable Positionen eines rationalen Agenten betrachtet werden können.

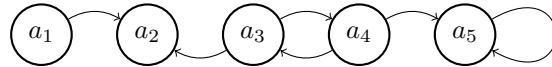
Formal ist ein AF ein Paar $F = (A, R)$ bestehend aus einer Menge von Argumenten A und einer Angriffsrelation $R \subseteq A \times A$, die beschreibt, welches Argument ein Argument attackiert (x attackiert y genau dann wenn $(x, y) \in R$). Ein AF ist also formal nichts anderes als ein gerichteter Graph (und lässt sich deshalb entsprechend graphisch repräsentieren). Die Argumente in A selbst bleiben hierbei völlig abstrakt und besitzen keinerlei Struktur. Es wird davon ausgegangen, dass für die Akzeptanz eines Argumentes einzig und allein entscheidend ist, ob es gegen Angriffe geeignet verteidigt werden kann.

Sei nun ein AF $F = (A, R)$ gegeben. Welche Teilmengen von A lassen sich sinnvoll von einem rationalen Agenten akzeptieren? Zunächst sollten sich die akzeptierten Argumente nicht selbst attackieren, also konflikt-frei sein: $S \subseteq A$ ist konflikt-frei, wenn es keine $x, y \in S$ gibt mit $(x, y) \in R$. Allerdings reicht das noch nicht aus, denn nicht jede konflikt-freie Menge von Argumenten kann sich gegen attackierende Argumente verteidigen.

Wir nennen eine Menge $S \subseteq A$ von Argumenten zulässig (admissible), wenn sie konflikt-frei ist und genau diese zusätzliche Eigenschaft erfüllt. Es muss also neben der Konfliktfreiheit gelten: wenn ein $x \in A$ und ein $y \in S$ existieren, so dass $(x, y) \in R$ (also ein Element von A attackiert ein Element von S), dann muss S ein Element z enthalten, das y verteidigt, also $(z, y) \in R$ für ein $z \in S$.

Zulässige Mengen sind damit gute Kandidaten für die gesuchten Mengen von Argumenten, die ein rationaler Agent akzeptieren kann. Allerdings ist z.B. die leere Menge trivialer Weise zulässig, und eine grundlose Weigerung, Argumente zu akzeptieren, kann sicher nicht als rational bezeichnet werden. Aus diesem Grund identifiziert Dung die gesuchten akzeptablen Mengen von Argumenten mit den maximalen zulässigen Mengen und nennt sie bevorzugte Extensionen von F . Auf der Basis dieser Definition kann man nun, ähnlich wie in der Default Logik, einen skeptischen Akzeptanzbegriff einführen (ein Argument wird skeptisch akzeptiert, wenn es in allen bevorzugten Extensionen ist), oder man kann jede der bevorzugten Extensionen als alternative akzeptable Position betrachten.

Betrachten wir ein kleines Beispiel. Es sei $F = (A, R)$ mit $A = \{a_1, a_2, a_3, a_4, a_5\}$ und $R = \{(a_1, a_2), (a_3, a_2), (a_3, a_4), (a_4, a_3), (a_4, a_5), (a_5, a_5)\}$. Es ergibt sich folgende graphische Darstellung:



In diesem Beispiel sind die Mengen $\emptyset, \{a_1\}, \{a_3\}, \{a_4\}$ sowie $\{a_1, a_3\}$ und $\{a_1, a_4\}$ zulässig. Damit sind die letzten beiden Mengen bevorzugte Extensionen.

Dung hat weitere Semantiken für AFs definiert, also alternative Definitionen von Extensionen. Die sogenannten stabilen Extensionen sind konflikt-freie Teilmengen von A , die jedes nicht enthaltene Argument attackieren, d.h. eine konflikt-freie Menge S ist stabile Extension, wenn es für jedes Element $x \in A \setminus S$ ein $y \in S$ gibt mit $(y, x) \in R$. Stabile Extensionen sind immer auch bevorzugte Extensionen. Allerdings gibt es Fälle, in denen keine stabile Extension existiert. Bevorzugte Extensionen dagegen gibt es immer. Im obigen Beispiel ist die einzige stabile Extension $\{a_1, a_4\}$.

Eine weitere interessante Semantik liefert die gegründete Extension (grounded extension). Jedes AF besitzt genau eine solche Extension. Sie kann iterativ erzeugt werden, indem man zunächst alle Argumente akzeptiert, die nicht attackiert werden. Man löscht nun die Argumente (mit den zugehörigen Attacken), die ihrerseits von den bereits akzeptierten Argumenten attackiert werden, und sucht in dem so reduzierten AF wiederum nach nicht attackierten Argumenten. Das wird solange iteriert, bis sich keine Änderung mehr ergibt. Die gegründete Extension ist Teilmenge jeder bevorzugten Extension. Man kann also sagen, sie beinhaltet die Argumente, die zumindest unter Berücksichtigung der bisher bekannten Argumente über jeden Zweifel erhaben sind. In unserem Beispiel enthält sie nur das Argument a_1 .

Natürlich können zusätzliche Argumente die entstehenden Extensionen komplett ändern, AFs sind also nichtmonoton – vielleicht der einfachste nichtmonotone Formalismus, der je entwickelt wurde. Bezüge zwischen AFs und Logikprogrammen gibt es in beide Richtungen. Dung hat gezeigt, dass man jedes Logikprogramm P in ein (allerdings exponentiell größeres) AF F übersetzen kann, so dass die stabilen Modelle von P und die stabilen Extensionen von F übereinstimmen. Umgekehrt kann man $F = (A, R)$ in ein Programm überführen, indem man für jedes Argument a eine Regel der Form

$$a \leftarrow \text{not } b_1, \dots, \text{not } b_n$$

einführt. Die b_i sind hier die Angreifer von a , also $\{b_1, \dots, b_n\} = \{b \mid (b, a) \in R\}$. Wieder entsprechen die stabilen Modelle von F den stabilen Modellen des so erzeugten Programms.

6.6 Ausblick

Das Forschungsgebiet des nichtmonotonen Schließens ist inzwischen mehr als 30 Jahre alt. Das Gebiet war in den Anfängen sicherlich theorieelastig. Im Vordergrund stand zunächst die Entwicklung neuartiger Logiken, von denen wir die wichtigsten in diesem Kapitel beschrieben haben. Aufgrund der Vielzahl der Ansätze waren dann zunächst deren theoretische Zusam-

menhänge zu klären. Insgesamt machte das nichtmonotonen Schließen zumindest nach außen oftmals den Eindruck, wenig anwendungsorientiert zu sein.

Das hat sich in der Zwischenzeit insbesondere mit der Entwicklung der – semantisch fundierten – Logikprogrammierung deutlich geändert. Die inzwischen existierenden ASP Solver etwa bieten sowohl von ihrer Funktionalität wie von der Effizienz her hervorragende Anwendungsmöglichkeiten, und eine ganze Reihe solcher Anwendungen existieren bereits [12].

Auch das Gebiet der Argumentation hat – neben neuartigen Ansätzen zur Evaluation von Argumenten – zu interessanten Implementierungen und Anwendungen insbesondere beim juristischen Schließen geführt, und auch hier sind sicherlich weitere spannende Entwicklungen zu erwarten.

In dieser knappen Darstellung konnten wir nur einige der relevantesten Ansätze vorstellen. Als weiterführende Literatur in diesem Gebiet wäre das Buch [11] zu nennen, das einen wesentlich ausführlicheren Überblick gibt, als es hier möglich war. Es geht auch auf den konditionalen Ansatz von Kraus, Lehmann und Magidor [29] oder den verwandten Ansatz von Pearl ein, der die Zusammenhänge zwischen nichtmonotonem Schließen und Schließen mit (infinitesimalen) Wahrscheinlichkeiten erhellt [52]. Weitere empfehlenswerte Bücher sind [43], [63] sowie [1]. Eine exzellente Übersicht über verschiedene Eigenschaften nichtmonotoner Inferenzrelationen und die sie erfüllenden Ansätze bietet [41].

Literaturverzeichnis

- [1] Antoniou, Grigoris, Nonmonotonic Reasoning, MIT Press, 1997
- [2] Baral, C., Gelfond, M., Logic Programming and Knowledge Representation, Journal of Logic Programming, 19,20:73–148, 1994
- [3] Baral, C., Subrahmanian, V.S., Duality between Alternative Semantics of Logic Programs and Nonmonotonic Formalisms, Intl. Workshop on Logic Programming and Nonmonotonic Reasoning, 1991
- [4] Benferhat, Salem, Cayrol, Claudette, Dubois, Didier, Lang, Jerome, Prade, Henri: Inconsistency Management and Prioritized Syntax-Based Entailment, Proc. IJCAI-93, Chambéry, France, 1993
- [5] Boutilier, Craig: Conditional Logics of Normality: A Modal Approach, Artificial Intelligence 68, No 1, July 1994
- [6] Brass, Stefan, Dix, Jürgen, Semantics of (Disjunctive) Logic Programs Based on Partial Evaluation, Journal of Logic Programming 38(2), 111–157, 1999
- [7] Brewka, Gerhard: Nonmonotonic Reasoning – Logical Foundations of Commonsense, Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 1990
- [8] Brewka, Gerhard: Cumulative Default Logic – In Defense of Nonmonotonic Inference Rules, Artificial Intelligence 50, 1991
- [9] Brewka, Gerhard: Reasoning About Priorities in Default Logic, Proc. AAAI 94, Seattle, 1994
- [10] Brewka, Gerhard: Well-Founded Semantics for Extended Logic Programs with Dynamic Preferences. *Journal of Artificial Intelligence Research*, 4:19–36, 1996.

- [11] Brewka, Gerhard, Dix, Jürgen, Konolige, Kurt: Nonmonotonic Reasoning – An Overview, CSLI Publications, Stanford, 1997
- [12] Brewka, Gerhard, Eiter, Thomas, Truszczyński, Miroslaw: Answer set programming at a glance. *Commun. ACM* 54(12): 92–103, 2011.
- [13] Delgrande, James P.: An Approach to Default Reasoning Based on a First-Order Conditional Logic: Revised Report, *Artificial Intelligence* 36, 1988
- [14] Delgrande, James P., Schaub, Torsten, und W. Ken Jackson, Alternative Approaches to Default Logic. *Artificial Intelligence* 70(1–2): 167–237, 1994
- [15] Doyle, J.: A Truth Maintenance System, *Artificial Intelligence* 12, 1979
- [16] Dix, Jürgen: Default Theories of Poole-Type and a Method for Constructing Cumulative Versions of Default Logic, Proc. ECAI 92, Wien, 1992
- [17] Dix, Jürgen, Furbach, Ulrich, Niemelä, Ilkka, Nonmonotonic Reasoning: Towards Efficient Calculi and Implementations, to appear in: A. Robinson, A. Voronkov, *Handbook of Automated Reasoning*, Elsevier
- [18] Dung, Phan Minh : On the Acceptability of Arguments and its Fundamental Role in Non-monotonic Reasoning, Logic Programming and n-Person Games. *Artif. Intell.* 77(2): 321–358, 1995
- [19] Eiter, T., Leone, N., Mateis, C., Pfeifer, G., Scarcello, F.: The KR System dlv: Progress Report, Comparisons and Benchmarks, Proc. 6th Intl. Conference on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann, 1998
- [20] Gabbay, Dov: Theoretical Foundations for Non-monotonic Reasoning in Expert Systems, In: *Logics and Models of Concurrent Systems*, Berlin, Springer Verlag, 1985
- [21] Geffner, Hector, Default Reasoning: Causal and Conditional Theories, Cambridge, MIT Press, 1992
- [22] van Gelder, A., Ross, K., Schlipf, J., The Well-Founded Semantics for General Logic Programs, *Journal of ACM*, p.221–230, 1990
- [23] Gelfond, M., Lifschitz, V.: Compiling Circumscriptive Theories into Logic Programs, Proc. 2nd Int. Workshop on Nonmonotonic Reasoning, Springer, LNCS 346, 1988
- [24] Gelfond, M., Lifschitz, V., Logic Programs with Classical Negation, Proc. 7th Intl. Conference on Logic Programming, 1990
- [25] Gordon, Thomas F.: Oblog 2 – A Hybrid Knowledge Representation System for Defeasible Reasoning, Proc. 1st Intl. Conference on Artificial Intelligence and Law, Boston, ACM Press, 1987
- [26] Gordon, Thomas F., The Pleadings Game: An Artificial Intelligence Model of Procedural Justice. Kluwer, Dordrecht, 1995.
- [27] Hanks, Steven, McDermott, Drew: Nonmonotonic Logic and Temporal Projection, *Artificial Intelligence* 33, 1987
- [28] Konolige, Kurt: On the Relation Between Default and Autoepistemic Logic, *Artificial Intelligence* 35 (3), 1988
- [29] Kraus, Sarit, Lehmann, Daniel, Magidor, Menachem: Nonmonotonic Reasoning, Preferential Models and Cumulative Logics, *Artificial Intelligence* 44, 167–207, 1990
- [30] Lakemeyer, Gerhard: All You Ever Wanted to Know About Tweety, Proc. 3rd Intl. Conference on Principles of Knowledge Representation and Reasoning, Cambridge, 1992
- [31] Levesque, Hector: All I know: A Study in Autoepistemic Logic, *Artificial Intelligence* 42, 263–309, 1990

- [32] Lifschitz, Vladimir: Computing Cirkumskription, Proc. IJCAI 85, 1985
- [33] Lifschitz, Vladimir: Nonmonotonic Databases and Epistemic Queries, Proc. 9th National Conference on Artificial Intelligence, 1991
- [34] Lifschitz, Vladimir: Circumscription. In: Gabbay, Robinson, Hogger (eds.), Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 3, pages 297–352. Oxford University Press, 1994
- [35] Lifschitz, Vladimir: Foundations of Declarative Logic Programming, in: G. Brewka (ed.), Principles of Knowledge Representation, Studies in Logic, Language and Information, CSLI publications, 1996
- [36] Lifschitz, Vladimir: Success of Default Logic, in: H. Levesque, F. Pirri (eds.), Logical Foundations for Cognitive Agents: Contributions in Honour of Ray Reiter, Springer, pages 357–373, 1999
- [37] Lin, Fangzhen, Shoham, Yoav: Epistemic Semantics for Fixed-Points Nonmonotonic Logics, Proc. Theoretical Aspects of Reasoning About Knowledge (TARK), 1990
- [38] Lehmann, Daniel: Another Perspective on Default Reasoning, Technical Report TR 92–12, Leibniz Center for Computer Science, Hebrew University, Jerusalem, 1992
- [39] Lukaszewicz, Witold: Considerations on Default Logic, Computational Intelligence 4, 1988
- [40] Makinson, David: General Theory of Cumulative Inference, In: Reinfrank et al. (eds), Non-Monotonic Reasoning, Springer, LNAI 346, 1989
- [41] Makinson, David: General Patterns in Nonmonotonic Reasoning, In: Gabbay, D.M., Hogger, C.J., Robinson, J.A. (Hrsg.), Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 3, Oxford University Press, 1994
- [42] Marek, Wiktor, Truszczyński, Miroslaw: Relating Autoepistemic and Default Logics, Proc. First Intl. Conference on Princiles of Knowledge Representation and Reasoning, 1989
- [43] W. Marek and M. Truszczyński. *Nonmonotonic Logics – Context-Dependent Reasoning*. Springer, 1993.
- [44] Marek, Wiktor, Truszczyński, Miroslaw: Stable Models and an Alternative Logic Programming Paradigm. In K. Apt, V. W. Marek, M. Truszczyński, und D. S. Warren (eds.), The Logic Programming Paradigm – A 25-Year Perspective, pp. 375–398. Springer, 1999.
- [45] McCarthy, John, Situations, Actions and Causal Laws, Stanford Artificial Intelligence Project, Memo 2, 1963
- [46] McCarthy, John: Circumscription – A Form of Nonmonotonic Reasoning, Artificial Intelligence 13, 1980
- [47] McCarthy, John: Applications of Circumscription to Formalizing Common Sense Knowledge, Proc. AAAI-Workshop Non-Monotonic Reasoning, 1984 (auch in Artificial Intelligence 28, 1986)
- [48] McCarthy, John, Hayes, Pat, Some Philosophical Problems from the Standpoint of Artificial Intelligence, in B. Meltzer, D. Michie (eds.), Machine Intelligence 4, pp. 463–502, 1969
- [49] Moore, Robert C.: Semantical Considerations on Nonmonotonic Logic, Artificial Intelligence 25, 1985 (Kurzfassung in Proc. IJCAI 83)
- [50] Niemelä, Ilkka: Logic Programming with Stable Model Semantics as Constraint Programming Paradigm. Annals of Mathematics and Artificial Intelligence, 25(3–4):241–273, 1999.

- [51] Niemelä, I., Simons, P., Efficient Implementation of the Stable Model and Well-Founded Semantics for Normal Logic Programs, Proc. 4th Intl. Conference on Logic Programming and Nonmonotonic Reasoning, Dagstuhl, Germany, Springer Verlag, 1997
- [52] Pearl, Judea: System Z: A Natural Ordering of Defaults with Tractable Applications to Nonmonotonic Reasoning, Proc. Third Conference on Theoretical Aspects of Reasoning About Knowledge, 1990
- [53] Pereira, L.M., Alferes, J.J., Well Founded Semantics for Logic Programs with Explicit Negation, Proc. 10th European Conference on Artificial Intelligence, Vienna, 1992
- [54] Poole, D.: A Logical Framework for Default Reasoning, Artificial Intelligence 36, 1988
- [55] Poole, D.: What the Lottery Paradox Tells Us about Default Reasoning, Proc. 1. Intl. Conference Principles of Knowledge Representation and Reasoning, 1989
- [56] Poole, D., Goebel, R., Aleliunas, R.: A Logical Reasoning System for Defaults and Diagnosis, University of Waterloo, Dep. of Computer Science, Research Rep. CS-86-06, 1986
- [57] Przymusinski, T., The Well-Founded Semantics Coincides with the Three-Valued Stable Semantics, Fundamenta Informaticae, 1989
- [58] Przymusinski, T., Stable Semantics for Disjunctive Programs, New Generation Computing, 9:401–424, 1991
- [59] Rahwan, Iyad, Simari, Guillermo: Argumentation in Artificial Intelligence, Springer Verlag, 2009
- [60] Reiter, Raymond: A Logic for Default Reasoning, Artificial Intelligence 13, 1980
- [61] Rescher, Nicholas: Hypothetical Reasoning, North Holland Publ., Amsterdam, 1964
- [62] Sagonas, K., Swift, T. and David S. Warren. An Abstract Machine for Computing the Well-Founded Semantics. In Proceedings of the Joint International Conference and Symposium on Logic Programming Bonn, Germany, pages 274–288, MIT Press, 1996
- [63] Schaub, Torsten, The Automation of Reasoning with Incomplete Information: From Semantic Foundations to Efficient Computation, Springer Verlag, 1998.
- [64] Siegel, Pierre: A Modal Language for Nonmonotonic Logic, Proc. DRUMS Workshop, Marseille, 1990
- [65] Tarski, Alfred, A lattice-theoretical fixpoint theorem and its applications, Pacific Journal of Mathematics, 5:285–309, 1955

7 Constraints

Petra Hofstedt

Der Begriff *Constraint* bezeichnet ein sprachliches Konzept zur deklarativen Beschreibung von Bedingungen und Relationen zwischen Variablen oder Objekten mit dem Ziel einer effizienten Modellierung und Lösung von Problemen, über die nur unvollständige Information vorhanden ist. Die Constraint-Programmierung hat sich in den 1980er Jahren als eigenständiges Forschungs- und Anwendungsfeld herausgebildet und ab etwa 1990 eine besondere Relevanz in Praxis und Wissenschaft erlangt.

Wir geben zunächst in Abschnitt 7.1 eine Einführung in das Gebiet der Constraints, benennen Anwendungen und stellen grundlegende Begriffe vor. Constraints werden an Hand ihrer Lösungsmethoden und Definitionsbereiche klassifiziert. In Abschnitt 7.2 stellen wir exemplarisch eine solche Klasse, die sog. Finite-Domain-Constraints vor und diskutieren ihre Lösungsmethoden. Dabei betrachten wir insb. auch globale Constraints, die sich als wichtiges Forschungsfeld etabliert haben. In Abschnitt 7.3 stellen wir einige bedeutende und typische Formen der Integration von Constraints in Programmiersprachen vor. Soft-Constraints erlauben den Umgang mit überspezifizierten Constraint-Problemen und die Modellierung von Präferenzen. Wir geben einen Überblick über Soft-Constraints in Abschnitt 7.4. In Abschnitt 7.5 diskutieren wir die Modellierung und Inferenz mit temporalen Constraints. Abschließend geben wir in Abschnitt 7.6 eine zusammenfassende Übersicht über weitere Teilgebiete des Forschungsgebiets Constraints und verweisen auf weiterführende Literatur.

7.1 Einführung

Constraints erlauben eine elegante und effiziente Modellierung und Lösung von Problemen mit unvollständiger Information. *Constraints* sind prädikatenlogische Formeln über Variablen, denen bestimmte Definitionsbereiche, ihre *Domänen* zugeordnet werden. In der Constraint-Programmierung beschreibt man deklarativ mit Hilfe von Constraints die Lösungsmenge eines Problems. Lösungen sind Belegungen der Variablen, die die Constraints erfüllen.

Beispiel 7.1.1. Ein pythagoreisches Tripel wird von drei natürlichen Zahlen x , y und z gebildet, welche die Längen der Seiten eines rechtwinkligen Dreiecks repräsentieren. Eine constraint-basierte Modellierung solcher Tripel über dem Definitionsbereich $\{1, \dots, 100\}$ kann durch folgende Constraint-Konjunktion angegeben werden:

$$x^2 + y^2 = z^2 \wedge x, y, z \in \{1, \dots, 100\}$$

Eine mögliche Lösung ist das Tripel $(x, y, z) = (3, 4, 5)$.

Eugene Freuder [16] gibt eine Charakterisierung der Constraint-Programmierung:

„Constraint programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it.“

Der Nutzer beschreibt auf deklarative Weise das Problem, während der Lösungsprozess aus Nutzersicht in den Hintergrund tritt und vom Computer bzw. vom Laufzeitsystem der constraint-basierten Sprache übernommen wird.

Constraint-Systeme (vgl. [25]) formalisieren Syntax, Semantik und Domänen von Constraints. Sie verlangen i.d.R. unterschiedliche Methoden zur Constraint-Behandlung, z.B. zur Überprüfung der Erfüllbarkeit von Constraint-Konjunktionen, zu ihrer Vereinfachung oder zur Berechnung konkreter Lösungen oder Lösungsmengen und ggf. optimaler Lösungen.

Constraint-Systeme und Lösungsmethoden liefern eine Klassifikation von Constraints. Hierunter zählen lineare arithmetische Constraints, die z.B. mit der Simplex-Methode oder mittels Eliminationsverfahren gelöst werden können, arithmetische Constraints, die z.B. durch Intervall-Arithmetik behandelbar sind und Boolesche Constraints. Eine wichtige Klasse sind die sog. Finite-Domain-Constraints, deren Variablen endliche Domänen zugeordnet werden. Finite-Domain-Constraint-Probleme spannen somit a priori einen endlichen Suchraum auf, so dass entsprechende Lösungsmethoden letztendlich auf Domänenreduktion durch Constraints und einer geschickten Suche basieren. Wir gehen hierauf im Detail in Abschnitt 7.2 ein.

Constraints werden als syntaktische Konstrukte in Sprachen integriert. Die entsprechenden Lösungsverfahren werden als sog. *Constraint-Löser* (engl. *constraint solver*) in die Auswertung des Programms integriert. Aus Sicht des Nutzers treten sie in den Hintergrund. Der Nutzer muss die Verfahren zur Constraint-Behandlung nicht explizit beschreiben, kann sie aber i.d.R. spezialisieren und an sein Problem anpassen. Die Einbettung von Lösungsverfahren in Sprachen diskutieren wir in Abschnitt 7.3.

Ein Constraint-Löser bietet verschiedene Operationen auf Constraints eines Constraint-Systems an, hierunter insb. einen Erfüllbarkeitstest. Der Löser arbeitet auf einem *Constraint-Speicher* (engl. *constraint store*), dem er schrittweise Constraints aus dem Programm hinzufügt, wobei er die Erfüllbarkeit der aktuellen Constraint-Menge oder Konjunktion überprüft. Weitere (optionale) Funktionen umfassen die Berechnung konkreter Lösungen, die Vereinfachung von Constraints und die Prüfung der Folgerbarkeit von Constraints aus dem aktuellen Constraint-Speicher.

Anwendungsfelder der constraint-basierten Programmierung liegen sowohl im wissenschaftlichen Kontext, z.B. bei der Verarbeitung natürlicher Sprache, im Theorem-Beweisen, in der Analyse von Programmen und der Molekularbiologie, als auch in der industriellen Praxis. Typische Anwendungen sind hier Optimierungsprobleme und Schedulingaufgaben, Konfiguration, Schaltkreisdesign und -verifikation, graphische Systeme und Nutzer-Interfaces. In [47], insb. Kapitel 22 bis 26, und [24], Abschnitt 4.4, finden sich viele Beispiele und Literaturverweise.

7.2 Finite-Domain-Constraints

In diesem Abschnitt betrachten wir Constraint-Systeme mit endlichen Domänen, sog. Finite-Domain- bzw. FD-Constraint-Systeme. Dieses Teilgebiet der Constraint-Programmierung ist gut erforscht und wird vielfältig in der Praxis, z.B. zur Beschreibung und Lösung von Planungsaufgaben und von Scheduling- und Optimierungsproblemen angewendet.

In Abschnitt 7.2.1 stellen wir Constraint-Satisfaction-Probleme zur Modellierung von FD-Problemen vor, wir skizzieren Lösungstechniken dieser Klasse von Constraints in den Abschnitten 7.2.2 bis 7.2.4.

7.2.1 Constraint-Satisfaction-Probleme

Zur Modellierung mit FD-Constraints nutzt man Constraint Satisfaction Probleme.

Definition 7.2.1 (Constraint Satisfaction Problem (CSP), Constraint). *Ein Constraint Satisfaction Problem (CSP) \mathcal{P} ist ein Tripel $\mathcal{P} = (X, D, C)$. Dabei ist $X = \{x_1, x_2, \dots, x_n\}$ eine Menge von Variablen x_1, \dots, x_n , $D = (D_{x_1}, D_{x_2}, \dots, D_{x_n})$ ist ein n -Tupel endlicher Domänen, so dass D_{x_i} die Menge der möglichen Werte von x_i ist und $C = \{c_1, c_2, \dots, c_k\}$ ist eine Menge von Constraints.*

Ein Constraint c ist eine Relation über einer Menge $Y \subseteq X$ von Variablen, d.h. c ist eine Teilmenge des kartesischen Produkts der Domänen der Variablen aus Y .

Die Variablen eines Constraints c bezeichnen wir im Folgenden mit $vars(c)$.

Ein *unäres* Constraint ist ein Constraint der Stelligkeit eins, ein *binäres* Constraint hat die Stelligkeit zwei. Weiterhin sprechen wir von n -stelligen Constraints, $n > 0$.

Wir bezeichnen mit $C_\wedge = \bigwedge_{c \in C} C$ die Konjunktion der Constraints einer Menge C .

Definition 7.2.2 (Lösung). *Eine Lösung eines CSPs $\mathcal{P} = (X, D, C)$ über den Variablen $X = \{x_1, \dots, x_n\}$ ist eine Belegung $\sigma : X \rightarrow \bigcup_{i \in \{1, \dots, n\}} D_{x_i}$ mit $\sigma(x_i) \in D_{x_i}$, die die Konjunktion C_\wedge der Constraints aus C erfüllt.*

Eine Lösung eines CSPs $\mathcal{P} = (X, D, C)$ ist also eine Belegung, unter der die Konjunktion C_\wedge gilt, d.h. alle Constraints aus C gelten.

Eine Lösung σ eines CSPs über den Variablen $X = \{x_1, \dots, x_n\}$ mit $\sigma(x_i) = d_i$, $i \in \{1, \dots, n\}$, $d_i \in D_{x_i}$, schreiben wir auch als Menge $\sigma = \{x_1/d_1, \dots, x_n/d_n\}$ von Zuordnungen.

Beispiel 7.2.1. *Ziel des n -Damen-Problems ist die Platzierung von n Damen auf einem $n \times n$ -Schachbrett, so dass keine Dame im Einflussbereich einer anderen steht. Es gelten hierbei die üblichen Schachregeln, d.h. keine Dame darf sich mit einer anderen die Zeile, die Spalte oder eine Diagonale teilen.*

Wir modellieren das Problem mit Hilfe von Constraints. Da in einer gültigen Konfiguration jeder Spalte des Schachbretts genau eine Dame zugeordnet werden muss, stellen wir das Schachbrett durch ein Folge von Damen q_1, \dots, q_n dar, so dass i der Spaltenposition der Dame q_i entspricht. Die Werte der Variablen stehen für die Zeilenpositionen der jeweiligen Damen.

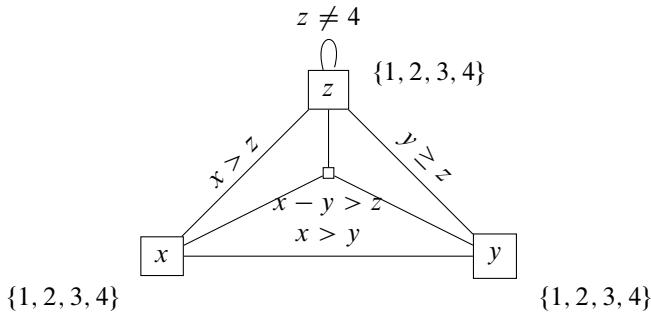


Abbildung 7.1: Constraint-Graph.

Das CSP $\mathcal{P}_{Q4} = (X, D, C)$ mit $X = \{q_1, q_2, q_3, q_4\}$, $D = (D_{q_1}, D_{q_2}, D_{q_3}, D_{q_4})$, $D_{q_i} = \{1, 2, 3, 4\}$ und $C = C_1 \cup C_2$, wobei gilt

$$\begin{aligned} C_1 &= \bigcup_{i, j \in \{1, \dots, 4\}, i < j} \{q_i \neq q_j\} \\ C_2 &= \bigcup_{i, j \in \{1, \dots, 4\}, i < j} \{(q_i - i \neq q_j - j), (q_i + i \neq q_j + j)\} \end{aligned}$$

beschreibt das 4-Damen-Problem. Die Constraints C_1 legen unterschiedliche Zeilenpositionen für alle Damen fest, C_2 beschreiben die Einflussbereiche der Damen auf den Diagonalen.

Eine Lösung des CSPs \mathcal{P}_{Q4} ist $\sigma = \{q_1/2, q_2/4, q_3/1, q_4/3\}$.

In ähnlicher Weise wie das n -Damen-Problem lassen sich z.B. kryptoarithmetische Puzzles, Sudoku, magische Sequenzen und Quadrate constraint-basiert formulieren.

Constraint-Graphen

CSPs werden auch als *Constraint-Netzwerke* (vgl. [5]) bezeichnet und lassen sich durch Graphen visualisieren. Hierbei werden Variablen durch Knoten und Constraints durch Kanten repräsentiert. Für Constraints einer Stelligkeit $n > 2$ verwendet man Hyperkanten.

Beispiel 7.2.2. Wir betrachten ein CSP $\mathcal{P} = (X, D, C)$ mit $X = \{x, y, z\}$, den Domänen $D_x = D_y = D_z = \{1, 2, 3, 4\}$ und den Constraints $C = \{x > y, x > z, y \geq z, x - y > z, z \neq 4\}$. Abbildung 7.1 zeigt einen Graphen für \mathcal{P} . Neben den die Variablen repräsentierenden Knoten sind die jeweiligen Variablen-Domänen angegeben.

Constraint-Lösungstechniken

Eine Möglichkeit, für ein gegebenes CSP Lösungen zu ermitteln oder seine Inkonsistenz nachzuweisen, ist die systematische Untersuchung des Suchraums. Da der Suchraum u.U. sehr groß sein kann, versucht man zunächst, ihn mit Hilfe der einzelnen Constraints des CSPs lokal einzuschränken.

Die Behandlung eines Constraints setzt sich aus drei Stufen zusammen: (1) Domänenreduktion, (2) Propagation und (3) Suche. Die Domänenreduktion oder -Filterung entfernt Werte aus

den Domänen der an einem Constraint c beteiligten Variablen, wenn sie nicht mit dem Constraint c konsistent sind. Die Propagation bezeichnet die weitere Domänenreduktionen auf Grund von Constraints c' , die mit c gemeinsame Variable haben. Dieser iterative Prozess wird fortgesetzt bis ein Fixpunkt erreicht ist. In diesem Zustand ist das CSP lokal konsistent, d.h. jedes Constraint für sich ist lokal bzw. individuell konsistent. Insbesondere bedeutet das aber auch, dass in den Domänen Werte verblieben sein können, die an keiner Lösung beteiligt sind und sogar, dass trotz nicht-leerer Variablen-Domänen das CSP inkonsistent ist. Eine Suche nach Lösungen findet schließlich verschränkt mit der Herstellung der lokalen Konsistenz statt. Die Domänenreduktion führt zu einer Reduktion des Suchraums und damit zu einer Beschleunigung der Lösungsfindung, wobei gleichzeitig der Aufwand für die Domänenreduktion selbst zu berücksichtigen ist.

Wir gehen in Abschnitt 7.2.2 auf die Domänenreduktion ein und stellen Konsistenzbegriffe vor. In Abschnitt 7.2.3 diskutieren wir Propagation und Suchtechniken.

7.2.2 Lokale und globale Konsistenz

Lokale Konsistenz bezeichnet Konsistenz-Zustände eines CSPs nach der Propagierung einzelner Constraints. Die im Folgenden diskutierten Konsistenz-Algorithmen beseitigen lokale Inkonsistenzen eines Constraint-Netzwerks, d.h. sie identifizieren und löschen anhand lokaler Constraints Variablenbelegungen, die niemals Teil einer Lösung sein können. Die Lösungsmenge eines CSPs \mathcal{P}' mit durch Konsistenzherstellung reduzierten Variablen-Domänen entspricht der des ursprünglichen CSPs \mathcal{P} , d.h. durch die Herstellung lokaler Konsistenz gehen weder Lösungen verloren noch kommen neue hinzu.

Die einfachste Form der Konsistenz ist die *Knotenkonsistenz*. Die Analyse der Knotenkonsistenz identifiziert nicht mit unären Constraints verträgliche Werte der Variablen-Domänen, die somit entfernt werden können.

Definition 7.2.3 (Knotenkonsistenz (engl. node consistency)). *Gegeben sei ein CSP $\mathcal{P} = (X, D, C)$. Ein unäres Constraint $c \in C$, $vars(c) = \{x\}$, ist knotenkonsistent, wenn $D_x \subseteq c$. Das CSP \mathcal{P} ist knotenkonsistent, wenn alle unären Constraints aus C knotenkonsistent sind.*

Listing 1 gibt einen Algorithmus zur Herstellung der Knotenkonsistenz für ein gegebenes CSP \mathcal{P} über den Variablen $\{x_1, \dots, x_n\}$ an. Anhand der Menge $C_{un} \subseteq C$ der unären Constraints von \mathcal{P} werden die Variablen-Domänen eingeschränkt.

Listing 1 Knotenkonsistenz

```

1  nodeConsistency( $\mathcal{P}$ ) =
2  foreach  $c \in C_{un}$  do {
3      Sei  $\{x_i\} = vars(c)$ .
4       $D_{x_i} := D_{x_i} \cap c$ 
5  }
6  return  $\mathcal{P}$  mit neuen Domänen  $D$ 

```

Kantenkonsistenz bezeichnet die Verträglichkeit der Domänen von Variablen binärer Constraints. Das Ziel ist auch hier wieder die Entfernung von Domänenwerten, die die lokalen Constraints nicht erfüllen.

Definition 7.2.4 (Kantenkonsistenz (engl. arc consistency)). *Gegeben sei ein CSP $\mathcal{P} = (X, D, C)$. Ein binäres Constraint $c \in C$ über den Variablen x_i und x_j , $i \neq j$, ist kantenkonsistent, wenn gilt:*

$$\begin{aligned}\forall d_i \in D_{x_i}, \exists d_j \in D_{x_j} \text{ mit } (d_i, d_j) \in c \text{ und} \\ \forall d_j \in D_{x_j}, \exists d_i \in D_{x_i} \text{ mit } (d_i, d_j) \in c.\end{aligned}$$

Das CSP \mathcal{P} ist kantenkonsistent, wenn alle binären Constraints aus C kantenkonsistent sind.

Listing 2 zeigt das Vorgehen des Algorithmus AC-1 [37]. Für jedes binäre Constraint $c \in C_{bin} \subseteq C$ werden die Werte der beteiligten Variablen aus den Domänen gelöscht, die nicht mit c verträglich sind (Zeilen 4–8). Aufgrund gemeinsamer Variablen von Constraints kann eine Domänenreduktion durch ein Constraint c weitere Reduktionen durch ein anderes Constraint c' zur Folge haben. Daher wird der vollständige Durchlauf über alle binären Constraints wiederholt, bis es zu keinen Änderungen der Domänen mehr kommt (Zeilen 2–9). Man beachte, dass ein CSP als inkonsistent erkannt werden kann, wenn während der Herstellung der lokalen Konsistenz mindestens eine Variablen domäne leer wird.

Listing 2 Kantenkonsistenz AC-1

```

1  arcConsistency( $\mathcal{P}$ ) =
2    do {
3       $D' := D$ 
4      foreach  $c \in C_{bin}$  do {
5        Sei  $\{x_i, x_j\} = vars(c)$ .
6         $D_{x_i} := \{d \in D_{x_i} \mid \exists e \in D_{x_j} : (d, e) \in c\}$ 
7         $D_{x_j} := \{e \in D_{x_j} \mid \exists d \in D_{x_i} : (d, e) \in c\}$ 
8      }
9    } while  $D \neq D'$ 
10   return  $\mathcal{P}$  mit neuen Domänen  $D$ 

```

Der Algorithmus AC-1 birgt Verbesserungsmöglichkeiten. Selbst wenn nur die Variablen domäne einer einzigen Variablen x reduziert wurde, betrachtet der Algorithmus im darauf folgenden Durchlauf erneut sämtliche binären Constraints, insb. auch solche, deren Variablen möglicherweise gar nicht mit der Variablen x in Verbindung stehen. Genau diese Beobachtung wird zu einer Verbesserung im Algorithmus AC-3 [37] in Listing 3 ausgenutzt. Hier wird eine Menge C_{work} binärer, noch einmal zu betrachtender Constraints verwaltet (Zeile 2). Bereits untersuchte Constraints werden aus C_{work} entfernt. Wurde die Domäne einer Variablen x reduziert, so werden all die binären Constraints in C_{work} aufgenommen, die x enthalten (Zeilen 7 und 10). AC-3 ist weit verbreitet und häufig genutzt. Der Algorithmus hat im schlechtesten Fall einen

Aufwand von $\mathcal{O}(ed^3)$, wobei e die Anzahl der Constraints ist und d die Kardinalität der größten Domäne. Weitere Verbesserungen von Kantenkonsistenz-Algorithmen (bis hin zu einem Aufwand von $\mathcal{O}(ed^2)$ im schlechtesten Fall) stellt im Detail [5] vor.

Die Verallgemeinerung der Kantenkonsistenz für n -stellige Constraints, $n \geq 2$, führt zur sog. *generalisierten Kantenkonsistenz* [14, 61].

Definition 7.2.5 (Generalisierte Kantenkonsistenz (engl. generalized arc consistency, GAC, hyper-arc consistency)). *Gegeben sei ein CSP $\mathcal{P} = (X, D, C)$. Ein n -stelliges Constraint $c \in C$, $n \geq 2$, über den Variablen x_1, \dots, x_n ist hyperkantenkonsistent, wenn gilt:*

Für alle $i \in \{1, \dots, n\}$ gilt: Für alle Werte $d_i \in D_{x_i}$ gibt es für alle $j \in \{1, \dots, n\}$, $j \neq i$, Werte $d_j \in D_{x_j}$, so dass die Belegung σ mit $\sigma(x_k) = d_k$, $k \in \{1, \dots, n\}$ das Constraint c erfüllt.

Generalisierte Kantenkonsistenz bedeutet also, dass es für jede Variable eines Constraints c für jeden ihrer möglichen Domänenwerte für alle anderen Variablen von c gleichzeitig konsistente Domänenwerte geben muss. Bei der Herstellung von GAC werden zur Domänenreduktion einer Variablen x eines Constraints c Domänenwerte aufgespürt, für die es keine kompatiblen Wertetupel für die anderen an c beteiligten Variablen gibt.¹

Listing 3 Kantenkonsistenz AC-3

```

1  arcConsistency3( $\mathcal{P}$ ) =
2     $C_{work} := C_{bin}$ 
3    do {
4       $c := arb(C_{work})$ ,  $C_{work} := C_{work} \setminus \{c\}$ 
5      Sei  $\{x_i, x_j\} = vars(c)$ .
6       $D'_{x_i} := \{d \in D_{x_i} \mid \exists e \in D_{x_j} : (d, e) \in c\}$ 
7      if  $D'_{x_i} \neq D_{x_i}$  then  $C_{work} := C_{work} \cup \{c' \in C_{bin} \setminus \{c\} \mid x_i \in vars(c')\}$ 
8       $D_{x_i} := D'_{x_i}$ 
9       $D'_{x_j} := \{e \in D_{x_j} \mid \exists d \in D_{x_i} : (d, e) \in c\}$ 
10     if  $D'_{x_j} \neq D_{x_j}$  then  $C_{work} := C_{work} \cup \{c' \in C_{bin} \setminus \{c\} \mid x_j \in vars(c')\}$ 
11      $D_{x_j} := D'_{x_j}$ 
12   } while  $C_{work} \neq \emptyset$ 
13   return  $\mathcal{P}$  mit neuen Domänen  $D$ 

```

Als weitere Stufe der lokalen Konsistenz wurde Pfadkonsistenz (engl. *path consistency*) untersucht. Entsprechende Algorithmen beschränken die VariablenDomänen durch Betrachtung impliziter Constraints, die durch die transitive Verknüpfung binärer Constraints bestehen [5]. Eine Verallgemeinerung der vorgenannten Konsistenzbegriffe ist die k -Konsistenz, die beschreibt, dass jede konsistente Belegung von $(k-1)$ Variablen zu einer solchen über k Variablen erweitert

¹ Dies entspricht im Kantenkonsistenz-Algorithmus in Listing 3 den Zeilen 6 und 9.

werden kann [15]. Die starke k -Konsistenz eines CSPs setzt darüber hinaus seine i -Konsistenz für $i \in \{1, \dots, k\}$ voraus. Ist ein CSP mit n Variablen stark n -konsistent, so sind die Domänen der Variablen minimal, d.h. sie enthalten keine Werte, die nicht an einer Lösung beteiligt sind. Das CSP ist dann *global konsistent* [5].

Definition 7.2.6 (Globale Konsistenz). *Sei $\mathcal{P} = (X, D, C)$ ein CSP mit den Variablen $X = \{x_1, \dots, x_n\}$, deren Domänen $D = (D_{x_1}, \dots, D_{x_n})$ und einer Menge von Constraints $C = \{c_1, \dots, c_k\}$ über X .*

Das CSP \mathcal{P} ist global konsistent wenn gilt:

$\forall i \in \{1, \dots, n\}: \forall d_i \in D_{x_i}:$

$\exists d_1 \in D_{x_1}, \dots, \exists d_{i-1} \in D_{x_{i-1}}, \exists d_{i+1} \in D_{x_{i+1}}, \dots, \exists d_n \in D_{x_n}, \text{ so dass}$

die Belegung σ mit $\sigma(x_k) = d_k, k \in \{1, \dots, n\}$ eine Lösung für $C \wedge$ ist.

Beispiel 7.2.3. *Nach Herstellung von Knoten- und Kantenkonsistenz bzw. GAC erhalten wir für das CSP \mathcal{P} aus Beispiel 7.2.2 ein äquivalentes CSP \mathcal{P}' mit folgenden reduzierten Domänen:*

$D_x = \{3, 4\}, D_y = D_z = \{1, 2\}.$

Das CSP \mathcal{P}' ist nicht global konsistent, da der Wert 2 aus der Domäne der Variablen z an keiner Lösung von \mathcal{P}' beteiligt ist.

Das CSP \mathcal{P} bzw. \mathcal{P}' hat drei Lösungen:

$\sigma_1 = \{x/3, y/1, z/1\}, \sigma_2 = \{x/4, y/1, z/1\}$ und $\sigma_3 = \{x/4, y/2, z/1\}$.

Auf Grund des mit dem Grad der Konsistenz steigenden Aufwands der Algorithmen zu ihrer Herstellung beschränkt man sich i.d.R. auf Knoten- und Kantenkonsistenz bzw. GAC. Zur Lösung eines CSPs werden Konsistenz-Algorithmen und Suchverfahren verzahnt angewendet.

7.2.3 Suchtechniken

Ein lokal konsistentes CSP liefert weder direkt Lösungen, noch sichern Knoten- und Kantenkonsistenz bzw. GAC deren Vorhandensein. Im Folgenden betrachten wir daher (vollständige) Suchtechniken. Wie bisher gehen wir von CSPs mit endlichen Domänen aus, somit ist eine vollständige und terminierende Untersuchung des Suchraums möglich.

Da ein einfaches *Generieren und Testen* einer Belegung zur Berechnung von Lösungen aufgrund des i.A. sehr großen Suchraums sehr ineffizient ist, nutzt man das Wissen aus den Constraints, um die Domänen bereits vor bzw. während der Generierung von Teillösungen einzuschränken.

In einfacher Weise wendet man dies bereits beim (*chronologischen*) *Backtracking* an. Schrittweise werden Variable mit Werten ihrer Domänen belegt bis eine Lösung gefunden ist oder die Inkonsistenz des CSPs nachgewiesen wurde. Der Suchraum wird in Form eines sog. Suchbaums durchlaufen. In jedem Schritt, d.h. nach jeder Belegung einer Variablen wird die Konsistenz der aktuellen Teilbelegung hinsichtlich der Constraints des CSPs überprüft. Treten Inkonsistenzen auf, so wird die Belegung für diese Variable rückgängig gemacht und eine alternative Wertauswahl getroffen. Ist die Wertemenge leer, so wird weiter zurückgegangen und die Wertauswahl der vorhergehenden Variablen widerrufen.

Tatsächlich wird Backtracking aber in Kombination mit Konsistenztechniken, Suchstrategien und Heuristiken verwendet, um schneller eine, alle oder eine beste Lösung zu generieren oder die Inkonsistenz des Netzwerks nachzuweisen.

Konsistenztechniken

Techniken zur Herstellung lokaler Konsistenz können unmittelbar vor der Suche zur initialen Suchraumbeschränkung oder mit der Suche verzahnt, d.h. im Wechsel mit dieser angewendet werden. Letzteres erlaubt den Suchraum sukzessive und nach jeder Erweiterung einer Teillösung einzuschränken.

Suchstrategien

Man unterscheidet rückwärts gerichtete (look-back) und vorwärts gerichtete (look ahead) Suchstrategien. Wir skizzieren kurz Verfahren beider Klassen, eine ausführliche Darstellung, die auch Kombinationen solcher Strategien diskutiert, findet man z.B. in [45, 60].

Look-back-Verfahren zielen darauf ab, das wiederholte Durchlaufen fehlgeschlagener Zweige im Suchbaum zu vermeiden. So versucht bspw. *Backjumping* diejenige Variable im Suchbaum zu identifizieren, deren inkonsistente Belegung für das Backtracking verantwortlich war und springt an diese Stelle im Suchbaum zurück, möglicherweise über mehrere Ebenen. *Back-checking* und *Backmarking* vermerken weiterhin inkonsistente und konsistente Teilbelegungen, um diese nicht erneut prüfen zu müssen.

Look-ahead-Strategien schließen voraus schauend inkonsistente Belegungen aus, indem sie Konsistenztechniken innerhalb der Suche nach jeder Wertauswahl anwenden, um so den Suchraum zu reduzieren. Diese Strategien unterscheiden sich in Form und Umfang der verwendeten Konsistenztechniken. Während z.B. *Forward Checking* Kantenkonsistenz nur für Constraints mit noch genau einer ungebundenen Variablen herstellt, tut der MAC-Algorithmus (*Maintaining Arc Consistency* [53]) dies für alle Constraints mit noch unbelegten Variablen.

Listing 4 zeigt den MAC-Algorithmus. Zunächst wird in Zeile 2 lokale Konsistenz, also bspw. Knoten- und verallgemeinerte Kantenkonsistenz hergestellt. Wird hierbei bereits die Domäne mindestens einer Variablen leer, so können wir keine Lösung finden und das gegebene CSP war inkonsistent (Zeilen 3, 4). Sind zu diesem Zeitpunkt hingegen die Domänen aller Variablen einelementig, so haben wir eine Lösung gefunden (Zeilen 5, 6): Wir geben das aktuelle CSP mit seinen einelementigen Domänen, die die Lösung definieren, aus. Andernfalls erweitern wir die aktuelle Teillösung, indem wir eine Variable y mit mindestens zweielementiger Domäne auswählen (Zeile 8) und für sie eine Belegung d_y bestimmen (Zeile 10). Mit dieser neuen Domänenbeschränkung wird erneut der Algorithmus MAC aufgerufen (Zeilen 12, 13). Hierdurch wird die Herstellung der lokalen Konsistenz mit der eigentlichen Suche verschränkt. Die Zeilen 9–15 realisieren Backtracking über der Wertauswahl.

Variablenauswahl- und Wertauswahlheuristiken

Heuristiken bei der Variablen- und Wertauswahl können sich deutlich auf eine schnelle Lösungsfindung von Constraint-Problemen auswirken. In Listing 4 wurden in den Zeilen 8 und 10 Freiheitsgrade bei der Auswahl der nächsten zu instanzierenden Variablen und des nächsten Domänenwertes gelassen. Hier setzen Variablen- und Wertauswahlheuristiken an.

Beide können statisch oder dynamisch sein, d.h. die Reihenfolgen der Variablen bzw. Werte zur Auswahl ist entweder a priori vorgegeben oder wird während des Suchprozesses ermittelt.

Listing 4 Maintaining Arc Consistency (MAC)

```

1  MAC( $\mathcal{P}$ ) =
2     $\mathcal{P}$  := localConsistency( $\mathcal{P}$ )
3    if  $\exists i \in \{1, \dots, n\} : D_{x_i} = \emptyset$           // inkonsistentes CSP
4      then return false
5    if  $\forall i \in \{1, \dots, n\} : |D_{x_i}| = 1$            // Lösung gefunden
6      then return  $\mathcal{P}$ 
7    else {                                                 // erweiterte Teillösung
8      wähle  $y \in X$  mit  $|D_y| \geq 2$ .
9      while  $D_y \neq \emptyset$  do {
10        wähle  $d_y \in D_y$ .
11         $D_y := D_y \setminus \{d_y\}$ 
12         $\mathcal{R} := \text{MAC}(\mathcal{P}')$ , wobei
13           $\mathcal{P}' = (X, (D_{x_1}, \dots, D'_y, \dots, D_{x_n}), C)$  und  $D'_y = \{d_y\}$ 
14        if  $\mathcal{R} \neq \text{false}$  then return  $\mathcal{R}$ 
15      }
16      return false
17    }

```

Dynamische Variablenauswahlheuristiken wählen die nächste zu instanzierende Variable entweder anhand der Größe der aktuell verbleibenden Domäne oder basierend auf der Struktur des CSPs. Die Variablenauswahl beeinflusst die Struktur des Suchbaums. Ihr Ziel ist es, im Sinne einer schnellen Lösungsfindung den Suchbaum möglichst schlank zu halten. Weit verbreitet und einfach zu entscheiden ist bspw. das Fail-First-Prinzip: Man wählt die Variable mit der aktuell kleinsten Domäne. Eine andere Heuristik, die auch in Kombination mit der ersten angewendet wird, ist das Most-Constrained-Prinzip. Dabei werden Variablen, die in vielen Constraints auftreten, bevorzugt ausgewählt.

Die Wertauswahlheuristik verändert die Suchbaumstruktur nicht. Solche Heuristiken sind somit nur von Nutzen, um schnell eine erste Lösung zu finden. Sollen alle Lösungen aufgezählt werden oder ist das CSP inkonsistent, so muss der gesamte Suchbaum abgelaufen werden und die Reihenfolge der Wertauswahl ist irrelevant.

Anstelle einer konkreten Wertauswahl und Instanziierung einer Variablen ist es auch möglich, ein Domänen-Splitting-Verfahren anzuwenden. Dies bedeutet für das aktuell betrachtete CSP die Domäne D_x einer Variablen x aufzuteilen und verschiedene Instanzen des CSPs mit den Teildomänen D_x^1, \dots, D_x^m , wobei $D_x = \bigcup_{i \in \{1, \dots, m\}} D_x^i$, von x zu untersuchen.

Eine ausführlichere Diskussion von Heuristiken zur Variablen- und Wertauswahl findet man z.B. in [25, 60].

7.2.4 Globale Constraints

Globale Constraints erlauben eine globale Sicht bestimmter logischer Verknüpfungen lokaler Constraints, die bei der Modellierung einer Anwendung einen gemeinsamen Sachverhalt beschreiben. Ein globales Constraint legt i.d.R. eine Relation über einer nicht-fixierten Anzahl von Variablen fest. Bei seiner Behandlung und Lösung kann der Lösungsalgorithmus die Problemstruktur mit ausnutzen, eine Information, die bei der individuellen Betrachtung lokaler Constraints eher verloren geht.

Das `alldifferent`-Constraint

Die paarweise Verschiedenheit von n Variablen x_1, \dots, x_n , $n \geq 2$, wird durch das globale Constraint $\text{alldifferent}(x_1, \dots, x_n)$ repräsentiert. Es ist äquivalent zu einer Konjunktion von $\frac{n*(n-1)}{2}$ Ungleichungen:

$$\text{alldifferent}(x_1, \dots, x_n) \iff \bigwedge_{i < j, i, j \in \{1, \dots, n\}} x_i \neq x_j$$

Das `alldifferent`-Constraint ist ein Paradebeispiel eines globalen Constraints und wird bei der Modellierung vieler Probleme genutzt, z.B. in der Ressourcen- und Personalplanung [61]. Es eignet sich insb. zur Darstellung von Permutationen und von Zyklen in Graphen. Eine ganze Reihe weiterer globaler Constraints kann als Erweiterung des `alldifferent`-Constraints angesehen werden [62].

Beispiel 7.2.4. Wir nutzen das `alldifferent`-Constraint zur Formulierung des 4-Damen-Problems. Dazu greifen wir auf die Modellierung aus Beispiel 7.2.1 zurück, wobei wir die Konjunktionen der Ungleichheitsconstraints durch `alldifferent`-Constraints ersetzen.

Das 4-Damen-Problem wird dargestellt durch das CSP $\mathcal{P} = (X, D, C)$ mit $X = \{q_1, q_2, q_3, q_4\}$, $D = (D_{q_1}, \dots, D_{q_4})$, $D_{q_i} = \{1, 2, 3, 4\}$ und $C = \{c_1, c_2, c_3\}$, wobei gilt

$$c_1 = \text{alldifferent}(q_1, \dots, q_4)$$

$$c_2 = \text{alldifferent}(q_1 - 1, q_2 - 2, \dots, q_4 - 4) \text{ und}$$

$$c_3 = \text{alldifferent}(q_1 + 1, q_2 + 2, \dots, q_4 + 4).$$

Das `alldifferent`-Constraint c_1 entspricht der Constraint-Menge C_1 aus Beispiel 7.2.1, die Konjunktion $c_2 \wedge c_3$ der Menge C_2 .

Das `circuit`-Constraint

Ein weiteres globales Constraint ist `circuit`(k, x_1, \dots, x_n). Betrachtet man die Variablen x_i als Knoten eines Graphen, deren Werte ihre jeweiligen Nachfolger repräsentieren, so stehe k für die Anzahl der Zyklen im Graphen. Hierbei gelte $D_k, D_{x_i} \in \{1, \dots, n\}$.

Für den in Abbildung 7.2 gezeigten Graphen sind die Constraints `circuit`(3, 2, 1, 3, 6, 4, 5) und `circuit`(1, 2, 3, 4, 6, 1, 5) erfüllt.

Das `circuit`-Constraint eignet sich z.B. zur Modellierung des für viele praktische Anwendungen relevanten Traveling Salesman Problems (TSP).

Der *Global Constraint Catalog* [20] systematisiert eine Vielzahl weiterer globaler Constraint hierunter z.B. auch das `gcc`-Constraint (*global cardinality constraint*) als Generalisierung des

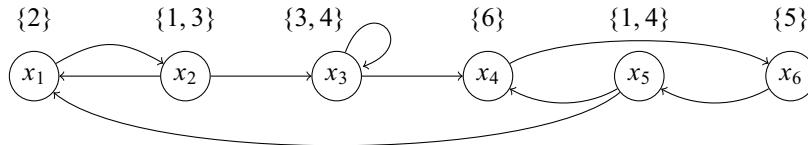


Abbildung 7.2: Graph zur Illustration des *circuit*-Constraints.

alldifferent-Constraints, das jedem Domänenwert eine bestimmte Anzahl von Variablen zuordnet und das cumulative-Constraint zur Beschreibung von Scheduling-Problemen unter Ressourcen-Beschränkungen. Van Hoeve und Katriel [62] geben einen ausführlichen Überblick über den Forschungsbereich der globalen Constraints.

Lösung von globalen Constraints am Beispiel des alldifferent-Constraints

Typischerweise lässt sich ein globales Constraint äquivalent durch eine logische Verknüpfung individueller Constraints ausdrücken. Bei der Behandlung des globalen Constraints nutzt man die inhärente Struktur des Problems aus, die bei der individuellen Betrachtung der lokalen Constraints hingegen meist verloren geht.

Wir skizzieren im Folgenden die Behandlung des alldifferent-Constraints, dessen Geschichte bis 1978 [34] zurückreicht und verschiedene Lösungsstrategien umfasst. Diese unterscheiden sich in den unterliegenden Techniken (und deren Komplexität) und damit einhergehend in den erreichbaren Konsistenzstufen.

Kantenkonsistenz durch binäre Dekomposition

Ein erster naiver Ansatz zur Konsistenzherstellung dekomponiert das alldifferent-Constraint in eine äquivalente Konjunktion von Ungleichheitsconstraint zwischen je zwei verschiedenen Variablen.

Beispiel 7.2.5. Wir betrachten das Constraint $\text{alldifferent}(x_1, x_2, x_3)$ als Menge von Constraints $C = \{x_1 \neq x_2, x_2 \neq x_3, x_1 \neq x_3\}$ über den Variablen x_1, x_2, x_3 mit den Domänen $D_{x_i} = \{1, 2\}, i \in \{1, 2, 3\}$ eines entsprechenden CSPs.

Die Anwendung des Algorithmus AC1 bzw. AC3 zur Herstellung von Kantenkonsistenz kann nicht erkennen, dass die Constraint-Menge C inkonsistent ist, da die einzelnen Ungleichheitsconstraints jeweils nur lokal untersucht werden. So existiert bspw. für jede Belegung der Variablen x_1 mit Sicht auf das Teilconstraint $x_1 \neq x_2$ eine zulässige Belegung für x_2 und umgekehrt. Gleiches gilt für die anderen Teilconstraints.

Diese Herangehensweise führt also dazu, dass die Herstellung der Kantenkonsistenz keine Domänenreduktion hervorruft, so lange nicht die Domäne mindestens einer Variablen nur noch einen Wert enthält. Eine nachfolgende Suche nach Lösungen muss dann ggf. die Inkonsistenz des Netzwerks erkennen. Bei großen Probleminstanzen, d.h. insb. bei Problemen mit großen Domänen, führt dies zu einem erheblichen Aufwand bei der Suche.

Lokale Konsistenz mit Graphalgorithmen

Régis [48] beschreibt für das alldifferent-Constraint einen Algorithmus zur Domänenreduktion basierend auf generalisierter Kantenkonsistenz (GAC).

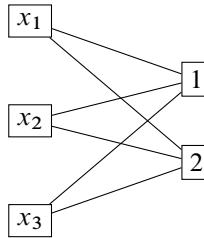


Abbildung 7.3: Bipartiter Graph: $\text{alldifferent}(x_1, x_2, x_3)$ mit $x_i \in \{1, 2\}$, $i \in \{1, 2, 3\}$.

Der Algorithmus nutzt Konzepte aus der Matching-Theorie. Man betrachtet dabei ein CSP als bipartiten Graphen, dessen Knotenmenge einerseits die Variablen des Constraints und andererseits alle Domänenwerte umfasst. Die Menge der Kanten wird durch die Variablen-domänen bestimmt.

Beispiel 7.2.6. Das CSP aus Beispiel 7.2.5 wird durch den Graphen in Abbildung 7.3 repräsentiert. Es ist sofort ersichtlich, dass kein Matching existieren kann, das alle Variablen überdeckt.

Eine wichtige Beobachtung, die der Ansatz nach Régin nutzt, ist, dass eine Lösung eines alldifferent -Constraints c zu einem Matching korrespondiert, das die Menge der Variablen von c überdeckt. Ein CSP ist nun hyperkantenkonsistent, wenn der korrespondierende Graph nur noch Kanten von Matchings enthält, die c überdecken.

Der Algorithmus berechnet zunächst ein Maximum-Matching M . Überdeckt M die Variablen aus c , so entspricht es einer Lösung des Constraints. Andernfalls ist das alldifferent -Constraint inkonsistent. Ausgehend von M werden alle sog. geraden alternierende Pfade und Zyklen berechnet. Sie, zusammen mit M repräsentieren gerade die Menge der Lösungen von c , so dass alle sonstigen Kanten des Graphen, d.h. alle sonstigen Domänenwerte der Variablen, entfernt werden können. Das CSP ist dann hyperkantenkonsistent.

Beispiel 7.2.7. Wir betrachten ein CSP $\mathcal{P} = (X, D, C)$ über den Variablen $X = \{x_1, \dots, x_6\}$, mit $D = (D_{x_1}, \dots, D_{x_6}) = (\{1, 3\}, \{2, 3\}, \{2, 3\}, \{1, 3, 4\}, \{4, 6\}, \{5, 6, 7\})$ und $C = \{\text{alldifferent}(x_1, \dots, x_6)\}$. Ein bipartiter Graph, der \mathcal{P} repräsentiert, ist in Abbildung 7.4(a) angegeben.

Die hervorgehobenen Kanten in Abbildung 7.4(b) stellen ein Maximum-Matching dar, das die Variablen des alldifferent -Constraints überdeckt, d.h. eine Lösung.

Abbildung 7.4(c) schließlich enthält nur noch die als Ergebnis des Algorithmus verbliebenen Kanten des nun hyperkantenkonsistenten CSPs. Dieser Graph repräsentiert genau die Lösungsmenge des alldifferent -Constraints.

Der Aufwand des Algorithmus wird mit $\mathcal{O}(m\sqrt{n})$ angegeben [61], wobei m die Summe der Kardinalitäten der Domänen ist und n die Anzahl der Variablen.

Grenzenkonsistenz mit Hall-Intervallen

Puget [46] stellte ein Verfahren zur Domänenreduktion mit Grenzenkonsistenz vor, das u.a. von Lopez-Ortiz et. al. [36] verbessert wurde. Das Verfahren nutzt sog. Hall-Intervalle.

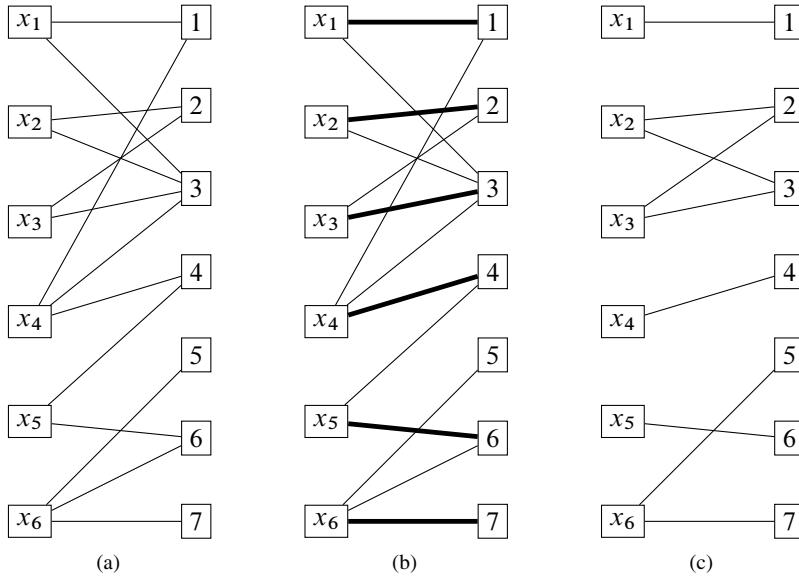


Abbildung 7.4: Das CSP aus Beispiel 7.2.7: (a) Repräsentation als bipartiter Graph (b) Ein Maximum-Matching, das die Variablenmenge überdeckt (c) Graph nach Herstellung der GAC.

Definition 7.2.7 (Hall-Intervall [61]). Seien x_1, \dots, x_n die Variablen eines CSPs $\mathcal{P} = (X, D, C)$. Für ein Intervall $I = [m_i n_i, max_i]$ sei $K_I = \{x_i \mid D_{x_i} \subseteq I\}$ die Menge aller Variablen, deren Domänen I umfasst.

Das Intervall I ist ein Hall-Intervall, wenn die Anzahl der Variablen, deren Domänen I umfasst, der Anzahl der Werte in I entspricht, d.h. es gilt $|I| = |K_I|$.

Beispiel 7.2.8. Wir betrachten erneut das CSP \mathcal{P} aus Beispiel 7.2.7.

Das Intervall $I_1 = [2, 3]$ ist ein Hall-Intervall mit der Variablenmenge $K_{I_1} = \{x_2, x_3\}$, denn es gilt $|I_1| = |K_{I_1}| = 2$. Weitere Hall-Intervalle sind $I_2 = [1, 3]$ mit $K_{I_2} = \{x_1, x_2, x_3\}$ und $I_3 = [1, 4]$ mit $K_{I_3} = \{x_1, x_2, x_3, x_4\}$.

Das Beispiel demonstriert bereits die Idee der Ausnutzung von Hall-Intervallen: Wenn wir ein Hall-Intervall für eine bestimmte Menge von Variablen identifizieren können, so können wir diesen Domänenbereich für alle anderen Variablen ausschließen.

Da wir hier auf Intervallen arbeiten, zielt das Verfahren auf das Erreichen einer weiteren Konsistenzstufe für das `alldifferent`-Constraint, der sog. *Grenzenkonsistenz*.

Definition 7.2.8 (Grenzenkonsistenz (engl. bounds consistency) [61]). Gegeben sei ein CSP $\mathcal{P} = (X, D, C)$. Ein Constraint $c \in C$ über den Variablen x_1, \dots, x_n , $n \geq 1$ ist grenzenkonsistent, wenn für alle $i \in \{1, \dots, n\}$ gilt:

Für die Werte $d_i \in \{\min(D_{x_i}), \max(D_{x_i})\}$ gibt es Werte $d_j \in [\min(D_{x_j}), \max(D_{x_j})]$, $j \in \{1, \dots, n\}$, $i \neq j$, so dass die Belegung σ mit $\sigma(x_k) = d_k$, $k \in \{1, \dots, n\}$ das Constraint c erfüllt.

Grenzenkonsistenz stellt sicher, dass für den minimalen und den maximalen Domänenwert einer Variablen jeweils Werte in den Intervallen der Domänen der anderen Variablen existieren, die das Constraint erfüllen.

Theorem 7.2.1 (aus [61]). *Das Constraint $\text{alldifferent}(y_1, \dots, y_m)$ ist grenzenkonsistent, genau dann wenn $|D_{x_i}| \geq 1, i \in \{1, \dots, n\}$ und*

1. *für jedes Intervall I gilt: $|K_I| \leq |I|$ und*
2. *für jedes Hall-Intervall I gilt: $\{\min(D_{x_i}), \max(D_{x_i})\} \cap I = \emptyset$ für alle $x_i \notin K_I$.*

Ein alldifferent -Constraint ist also dann grenzenkonsistent, wenn die Domänen nicht leer sind und weiterhin gilt: 1. Jedes Intervall umschließt mindestens so viele Werte, wie ihm Variablen zugeordnet sind. 2. Für jedes Hall-Intervall I gilt: Die Domänengrenzen einer Variablen x , die I nicht zugeordnet ist, d.h. $x \notin |K_I|$, sind auch nicht in I enthalten.

Beispiel 7.2.9. *Das CSP aus Beispiel 7.2.7 ist nicht grenzenkonsistent. Beispielsweise ist Punkt 2. aus Theorem 7.2.1 für das Hall-Intervall $I_1 = [2, 3]$ verletzt, denn es gilt $x_1 \notin K_{I_1}$ aber $\{\min(D_{x_1}), \max(D_{x_1})\} = \{1, 3\} \cap I_1 = \{3\} \neq \emptyset$.*

Das Hauptkonzept des Vorgehens bei der Herstellung von Grenzenkonsistenz des alldifferent -Constraints nach [36, 46] ist die Berechnung von Hall-Intervallen und eine darauf basierende Reduktion von Domänen durch Beschneiden von Domänengrenzen nicht am Hall-Intervall beteiligter Variabler. Zur Suche nach Hall-Intervallen werden die Variablen des alldifferent -Constraints aufsteigend nach ihren maximalen Domänenwerten geordnet und in dieser Reihenfolge schrittweise größer werdende Intervalle mit dem Ziel untersucht, maximale Hall-Intervalle zu entdecken. Wird ein solches gefunden, so erfolgt eine entsprechende Domänenreduktion.

Beispiel 7.2.10. *Wir betrachten noch einmal das CSP \mathcal{P} aus Beispiel 7.2.7.*

Die Entdeckung des Hall-Intervalls $I_1 = [2, 3]$ mit der die Variablenmenge $K_{I_1} = \{x_2, x_3\}$ erlaubt bspw. die Berechnung einer neuen oberen Domänengrenze für x_1 mit $\max(D_{x_1}) = 1$; das Hall-Intervall $I_2 = [1, 2, 3]$ mit $K_{I_2} = \{x_1, x_2, x_3\}$ führt zu einer neuen unteren Domänengrenze $\min(D_{x_4}) = 4$ für die Variable x_4 .

Wir erhalten schließlich ein grenzenkonsistentes CSP \mathcal{P}' mit neuen Variabldomänen $D' = (D'_{x_1}, \dots, D'_{x_6}) = (\{1\}, \{2, 3\}, \{2, 3\}, \{4\}, \{6\}, \{5, 6, 7\})$.

Man kann beobachten, dass die berechneten Domänen dabei i.A. weniger eingeschränkt werden, als unter Anwendung des Algorithmus nach Régin (vgl. Beispiel 7.2.7). Insbesondere verbleibt nun der Wert 6 in der Domäne von x_6 , da lediglich die Domänengrenzen untersucht und ggf. beschneitten werden.

Der Aufwand des Algorithmus zur Herstellung der Grenzenkonsistenz (im schlechtesten Fall) ist $\mathcal{O}(n \log n)$ [61], wobei n die Anzahl der Variablen ist.

7.3 Constraint-basierte Programmierung

Die constraint-basierte Programmierung (engl. *Constraint Programming*, CP) orientiert an folgendem *allgemeinen Schema*:

1. Deklaration der Constraint-Variablen und ihrer Domänen,
2. Definition der Problem-Constraints,
3. Aufruf von Such- oder Lösungsverfahren, ggf. unter Nutzung einer Zielfunktion.

Neben der Integration von Constraints und ihren Lösungsmechanismen in die Programmiersprache, erlauben constraint-basierten Programmiersysteme i.d.R. die Definition neuer Constraints, vergleichbar zu Funktionen oder Methoden in funktionalen bzw. objekt-orientierten Sprachen und darüber hinaus die Beschreibung eigener oder die Anpassung vordefinierter Such- und Lösungstechniken und Heuristiken.

Während Constraints zunächst vor allem logische Programmiersprachen erweiterten, gibt es heute auf Grund ihrer enormen Bedeutung in der Praxis verschiedenste Formen der Einbettung in Sprachen anderer Paradigmen, insb. auch in imperative und objekt-orientierte Sprachen.

Im Wesentlichen kann man zwei Formen constraint-basierter Programmiersysteme beobachten [23, 24]: Constraint-basierte Programmiersprachen und Constraint-Bibliotheken.

Eine *constraint-basierte Sprache* (im engeren Sinne) ist eine semantische Erweiterung einer Sprache um neue Konzepte und Auswertungsmechanismen bis hin zu einem vollständigen Neuentwurf. Typische Vertreter dieser Kategorie sind constraint-logische Sprachen, die eine Generalisierung der logischen Sprachen darstellen, z.B. ECLⁱPS^e-PROLOG [3], CHIP und SICSTUS-PROLOG. Sehr interessante und erfolgreiche Entwicklungen stellen weiterhin constraint-basierte Modellierungssprachen dar, wie z.B. OPL [22] und COMET [63]. Darüber hinaus zählen zur Kategorie der constraint-basierten Sprachen auch regelbasierte Sprachen, wie z.B. CHR [17] und Sprachen der nebenläufigen Constraint-Programmierung (engl. *Concurrent Constraint Programming*, CCP) [25, Kapitel 7]. Wir gehen auf Sprachen der constraint-logischen Programmierung in Abschnitt 7.3.1 ein, diskutieren Beispiele unter Nutzung von Modellierungssprachen in Abschnitt 7.3.2 und skizzieren das CCP-Modell in Abschnitt 7.3.4.

Constraint-Bibliotheken erlauben die funktionale und syntaktische Erweiterung einer existierenden Sprache um Constraints unter Ausnutzung existierender Sprachkonzepte. Häufig wird dieser Ansatz bei der Erweiterung objekt-orientierter Sprachen, aber auch bspw. in funktionalen Sprachen wie durch das MCP-Framework [55] umgesetzt. In Abschnitt 7.3.3 diskutieren wir die Integration von Constraints in objekt-orientierte Sprachen unter Nutzung von Bibliotheken. Konzepte der Constraint-Programmierung, wie Constraints, Constraint-Löser, Suchmechanismen, Heuristiken, etc. werden dabei als Klassen bzw. Objekte direkt in der Host-Sprache modelliert. Objekt-orientierte Sprachen wie JAVA und C++ sind in der Praxis anerkannt und verbreitet, ein deutlicher Vorteil hinsichtlich der Nutzerakzeptanz und des Erfolgs constraint-basierter Bibliotheken. Beispiele hierfür sind die JAVA-Bibliotheken CHOCO [35] und JACOP [33] und die C++-Bibliothek GECODE [56].

7.3.1 Constraint-logische Programmierung

Die constraint-logische Programmierung (engl. *Constraint Logic Programming*, CLP) [27, 28] ist aus der logischen Programmierung (mit PROLOG) hervorgegangen.

Sie verallgemeinert die logische Programmierung, indem sie im Regelkörper zusätzlich zu den Zielen Constraints spezieller Constraint-Domänen zulässt. Während die Constraints der logischen Sprachen, i.e. syntaktische Gleichheiten, weiterhin durch Unifikation behandelt werden, werden die zusätzlichen Constraints gesammelt, in den Constraint-Speicher übertragen und von einem Constraint-Löser behandelt und z.B. auf Erfüllbarkeit überprüft.

Die Sprachen lassen oft verschiedene Constraint-Domänen (FD-Constraints, arithmetische Constraints, Boolesche Constraints, etc.) mit entsprechenden Lösungsverfahren zu, die dann z.B. in Form von Bibliotheken vorliegen.

Beispiel 7.3.1. Das Listing 5 zeigt eine Implementierung des *n*-Damen-Problems in ECLⁱPS^e-PROLOG. Eingabe ist eine natürliche Zahl *N*, Ausgabe eine Liste *Queens* der Zeilenpositionen der Damen. Genutzt werden hier das Finite-Domain-Constraint *alldifferent* sowie arithmetische Constraints *E # D+I* und *F # D-I* aus der Bibliothek *ic*, die in Zeile 1 geladen wird.

Das Programm in Listing 5 setzt das allgemeine Schema der CP um: In den Zeilen 3–4 werden die an der Problembeschreibung beteiligten Variablen erzeugt und ihre Domänen fixiert. Zeilen 5–11 geben eine constraint-basierte Problembeschreibung. Dabei erzeugen wir in den Zeilen 5–8 zunächst die Listen der Diagonalen *LAdd* und *LSub* und geben dann, gemäß der Problem-Modellierung in Beispiel 7.2.4, *alldifferent*-Constraints auf den Zeilenpositionen *Queens* und den Diagonalenlisten an. Zeile 12 initiiert den Suchprozess.

Listing 5 ECLⁱPS^e-PROLOG: Das *n*-Damen-Problem

```

1  :- lib(ic).
2  queens(N,Queens) :-
3      length(Queens,N),           % erzeuge Queens = [Q1,...Qn]
4      Queens :: 1..N,
5      ( for(I,1,N), foreach(D,Queens), % erzeuge
6          foreach(E,LAdd),            % LAdd = [Q1 + 1,...,Qn + n] und
7          foreach(F,LSub) do        % LSub = [Q1 - 1,...,Qn - n]
8              E #= D+I, F #= D-I ),
9          alldifferent(Queens),
10         alldifferent(LAdd),
11         alldifferent(LSub),
12         labeling(Queens).
```

7.3.2 Constraint-basierte Modellierungssprachen

Auch constraint-basierte Modellierungssprachen, wie ZINC, OPL und COMET realisieren neue syntaktische und semantische Konzepte, trennen sich dabei aber im Gegensatz zur CLP von der logischen Programmierung.

ZINC/MINIZINC

ZINC mit seiner Untermenge MINIZINC [51] ist eine Constraint-Modellierungssprache, die auf eine Standardisierung der Constraint-Programmierung abzielt und die Integration verschiedener Löser und auch hybride Lösungstechniken unterstützt.

ZINC erlaubt die Spezifikation von CSPs und Optimierungsproblemen über Integern und reellen Zahlen. Mittels Annotationen können solver-spezifische Informationen zu einem Constraint-Modell hinzugefügt und ein unterliegender Solver angesteuert werden. ZINC-Modelle werden nach FLATZINC transformiert, das eine Spezialisierung für individuelle Backend-Solver unterstützt. Dies ermöglicht, auf einfache Weise verschiedene Constraint-Löser als Backend zu integrieren.

OPL

Die Optimization Programming Language IBM ILOG OPL [22, 26] ist eine Modellierungssprache zur mathematischen Programmierung und kombinatorischen Optimierung. Sie erlaubt die Spezifikation von Constraint-Problemen und die Definition und Adaption von Lösungs- und Suchtechniken und Heuristiken. Im Gegensatz zu der ihr unterliegenden IBM ILOG Solver C++-Bibliothek zur Constraint-Programmierung vermeidet sie die explizite Handhabung imperativer und objekt-orientierter Programmierkonzepte und unterstützt so eine klare mathematische bzw. deklarative Problemformulierung.

Im Folgenden greifen wir auf ein Beispiel aus [30] zurück. Eine magische Folge (engl. *magic sequence*) ist eine Folge $S = (s_0, \dots, s_{n-1})$ über n natürlichen Zahlen, wobei jedes $s_i, i \in \{0, \dots, n-1\}$, der Anzahl der in S vorkommenden Elemente vom Wert s_i entspricht, d.h. es gilt $\#_i S = s_i$. Beispielsweise ist $S_4 = (2, 0, 2, 0)$ eine magische Folge der Länge 4: Sie enthält zweimal die 0, nullmal die 1, zweimal die 2 und nullmal die 3. Ziel ist die Berechnung einer magischen Folge für eine gegebene natürliche Zahl.

Listing 6 zeigt eine Implementierung des Problems in OPL. Entsprechend dem allgemeinen Schema der Constraint-Programmierung werden in den Zeilen 1–3 die Variablen zur Problembeschreibung zusammen mit ihren Definitionsbereichen festgelegt. Die Formulierung der Constraints erfolgt in den Zeilen 5–8, der eigentliche Lösungsprozess wird in Zeile 4 angestoßen.

Zusätzlich zu den die magische Sequenz-Eigenschaft $\#_i S = s_i, i \in \{0, \dots, n-1\}$, beschreibenden Constraints in den Zeilen 5 und 6 sind hier zwei weitere, sog. redundante Constraints $\sum s_i = n$ und $\sum s_i \cdot i = n$ in den Zeilen 7 und 8 angegeben. Zwar folgen redundante Constraints aus einem gegebenen Constraint-Modell, jedoch erlauben sie typischerweise eine schnellere und stärkere Reduktion des Suchraums mit dem Ziel einer Beschleunigung des Lösungsprozesses ([47, Kapitel 12.4.5]).

COMET

Die Sprache COMET [63] unterscheidet sich fundamental von den bisher skizzierten Sprachen in ihrem Berechnungsmodell und damit verbunden, in der Klasse ihrer Anwendungen. COMET

Listing 6 OPL: Magische Folgen (nach [47, Kapitel 13])

```

1 int n << "Number of Variables:" ;
2 range Size 0..n-1;
3 var Size s[Size];
4 solve {
5   forall(i in Size)
6     s[i] = sum(j in Size) (s[j] = i);
7   sum(j in Size) s[j] = n;
8   sum(j in Size) s[j] * j = n;
9 };

```

realisiert eine durch Constraints gerichtete lokale Suche. Bei der lokalen Suche werden ausgehend von einer Startkonfiguration benachbarte bzw. ähnliche Konfigurationen untersucht und die beste unter diesen als nächster Ausgangspunkt gewählt. In COMET werden Constraints nicht wie bisher dafür verwendet, den Suchraum zu beschneiden, sondern als Mittel zur Steuerung der lokalen Suche.

Eine constraint-basierte Modellierung des Problems der magischen Folgen entspricht in weiten Teilen der OPL-Implementierung in Listing 6. Im Gegensatz zu diesem wird in einem entsprechenden COMET-Programm (vgl. [30, 63]) aber eine initiale, i.d.R. inkonsistente Variablenbelegung vorgenommen. Diese Belegung bildet die Ausgangskonfiguration für die lokale Suche. Zur Umsetzung der Suche können in COMET unter Nutzung vordefinierter Konstrukte und Abstraktionen in einfacher Weise komplexe Heuristiken und Metaheuristiken definiert werden.

7.3.3 Constraints als Objekte

Die Kombination imperativer Programmierung und somit von zeitbehafteten Berechnungen mit deklarativen, d.h. zeitlosen Constraints stellt eine besondere Herausforderung dar. Neben Ansätzen der sog. constraint-imperativen Programmierung (engl. *Constraint Imperative Programming*, CIP) (vgl. [24, Kapitel 5.6]) kommen hier vor allem Constraint-Bibliotheken zum Einsatz. Constraint-basierte Konzepte, wie Constraints, Constraint-Löser und -Speicher werden als Klassen oder Objekte direkt in der Host-Sprache modelliert. Erfolgreiche Vertreter dieses Integrationsansatzes sind bspw. die JAVA-Bibliotheken CHOCO [35] und JACoP [33] und die C++-Bibliotheken GECODE [56] und IBM ILOG Solver [26].

Beispiel 7.3.2. Listing 7 zeigt eine Implementierung der magischen Folgen mit der JAVA-Bibliothek CHOCO.

Auch hier ist wieder das allgemeine CP-Schema offensichtlich: Wir erzeugen zunächst die Constraints mit den ihnen zugeordneten Domänen, geben dann Constraints zur Problemmodellierung an und initiieren schließlich den Lösungsprozess.

Allerdings, müssen beim Bibliotheksansatz constraint-basierte Konzepte, wie das Modell und der Löser explizit gehandhabt werden (Zeilen 4, 14–16), was die Lesbarkeit erschwert.

Interessant an Listing 7 ist die Realisierung der magischen Sequenzeigenschaft $\#_i S = s_i$, $i \in \{0, \dots, n - 1\}$ in Zeile 9 mit Hilfe des globalen *occurrence*-Constraints, der eine Spezialisierung des in Abschnitt 7.2.4 genannten *gcc*-Constraints darstellt.

Listing 7 CHOCO: Magische Folgen [30]

```

1 int n = ...
2 int[] indices = new int[] {0, ..., n-1};
3 // erzeuge das Constraint-Modell
4 CPModel m = new CPModel();
5 // erzeuge die Variablen
6 IntegerVariable[] seq = makeIntVarArray("seq", n, 0, n-1);
7 // propagiere die Constraints
8 for (int i : indices) {
9     m.addConstraint(occurrence(seq[i], seq, i));
10 }
11 m.addConstraint(eq(n, sum(seq)));
12 m.addConstraint(eq(n, scalar(indices, seq)));
13 // erzeuge den Solver und löse
14 CPSolver s = new CPSolver();
15 s.read(m);
16 s.solve();

```

7.3.4 Nebenläufige Constraint-Programmierung

Im Modell der nebenläufigen Constraint-Programmierung (engl. *Concurrent Constraint Programming*, CCP) [50] dienen Constraints zur Beschreibung des Verhaltens nebenläufig agierender Prozesse. Die Prozesse kommunizieren und synchronisieren sich über den gemeinsamen Constraint-Speicher mit Hilfe der beiden Basisoperationen *ask* und *tell* dieses Modells. Die Operationen *ask* und *tell* entsprechen *read*- und *write*-Operationen (eines herkömmlichen Speichermodells) auf partieller Information.

Die Operation *tell* fügt ein Constraint zum Speicher hinzu und stellt somit Information für andere Prozesse zur Verfügung. Im CC-Modell (und CCP-Sprachen) tritt die Operation in 2 unterschiedlichen Formen auf: Man spricht vom *atomic tell*, wenn ein Constraint nur dann zum Speicher hinzugefügt werden kann, wenn ihre Konjunktion nicht erfüllbar bleibt. Andernfalls spricht man vom *eventual tell*. Der Constraint-Speicher verhält sich monoton, d.h. es können keine Informationen aus dem Speicher gelöscht werden.

Die Operation *ask* realisiert die Prozess-Synchronisation. Ein Prozess erfragt mittels *ask* die Folgerbarkeit eines Constraints c aus dem aktuellen Constraint-Speicher. Der Prozess wird blockiert bzw. suspendiert, wenn die Folgerbarkeit des Constraints c nicht entschieden werden kann (*blocking ask*). Er bleibt suspendiert bis ein anderer Prozess dem gemeinsamen Constraint-Speicher C mittels *tell* neue Information hinzugefügt hat, so dass der neue Speicher C' nun das Constraint c impliziert.

7.4 Soft-Constraints

Bei der constraint-basierten Modellierung größerer realer Probleme, wie z.B. von Stundenplanungsproblemen, kommt es schnell dazu, dass die spezifizierte Constraint-Menge nicht mehr erfüllbar ist. Man spricht dann von einem überspezifizierten (engl. *overconstrained*) Constraint-Problem. Diese Situation tritt z.B. dann auf, wenn Constraints nicht nur zur Beschreibung von zwingend notwendigen, sondern auch für erwünschte Bedingungen und Eigenschaften (und damit im Prinzip „missbräuchlich“) genutzt werden. Eine manuelle und mglw. heuristische Verfeinerung des Constraint-Problems mit dem Ziel, die Überspezifikation aufzulösen ohne ganz auf Präferenzen zu verzichten, ist aufwendig und nicht zwingend zielführend.

Der Begriff *Soft-Constraints* steht für eine Reihe von Verfahren, die eine partielle Erfüllung von Constraints unterstützen. Sie eignen sich insb. zur Modellierung von Präferenzen, Wahrscheinlichkeiten, Unsicherheiten und Kosten, aber auch zur Bewertung und dem Vergleich von Lösungen.

Während bei der Lösung eines Soft-Constraint-Problems sog. harte Constraints (engl. *hard/crisp constraints*) zwingend erfüllt sein müssen, beschreiben Soft-Constraints Bedingungen, deren Erfüllung lediglich erwünscht ist. Eine Lösung ist hier eine Belegung, die einen besten Wert hinsichtlich eines bestimmten vorgegebenen Kriteriums darstellt.

Bei der Stundenplanung in Universitäten ist das Ziel die Zuordnung von Kursen und Lehrenden zu Räumen, Zeitblöcken und Hörsälen unter Berücksichtigung von zwingenden Bedingungen, aber auch von Präferenzen. Harte Constraints sind bspw. die Anzahl und Größe der Räume, die Zahl der Kurse, geschätzte Zuhörerzahlen, Lage und Anzahl der Zeitblöcke. Auch kann ein Lehrender nicht gleichzeitig mehrere Kurse durchführen. Hingegen stellen Zeit- und Raumwünsche der Lehrenden und z.B. Universitätsrichtlinien, wie eine bevorzugte Nutzung bestimmter Räume oder Zeitblöcke, Soft-Constraints dar.

Partielle Constraint-Erfüllung (engl. *partial constraint satisfaction*) und Constraint-Hierarchien wurden Ende der 80er Jahre zur Behandlung überspezifizierter Constraint-Probleme entwickelt. Ab Anfang der 90er Jahre wurden spezielle Soft-Constraint-Konzepte untersucht, hierunter bspw. gewichtete Constraints und Fuzzy-Constraints. Eine Generalisierung bieten zwei wesentliche Soft-Constraint-Frameworks. Eine detaillierte Darstellung von Soft-Constraints und entsprechenden Lösungsalgorithmen, die sich häufig an denen zur Lösung von harten Constraints anlehnen, findet man in [40]. Wir skizzieren im Folgenden wesentliche Konzepte.

Spezielle Ausprägungen

In *gewichteten Constraint-Netzen* [40, 54] wird jedem Constraint ein Gewicht bzw. ein Kostenwert zugeordnet. Die Kosten einer Belegung berechnet sich aus der Summe der Gewichte aller Constraints, die von dieser Belegung verletzt werden. Eine Lösung mit minimalen Kosten ist optimal.

Gewichtete Constraint-Netze sind ein sehr ausdrucksstarker Formalismus, auf den andere Soft-Constraint-Konzepte, hierunter *possibilistische Constraints* [52] und *probabilistische Constraints* [13] effizient reduziert werden können. Werden die Kosten für alle Constraints mit 1 bewertet, so handelt es sich um ein sog. *MaxCSP* [18]. In diesem Fall charakterisiert das Gewicht einer Belegung die Anzahl der verletzten Constraints.

Fuzzy-Constraints [11, 49] trennen nicht zwischen zulässigen und unzulässigen Variablenbelegungen, sondern beschreiben statt dessen, inwieweit Wertetupel von jeweiligen Soll-Werten abweichen. Hierzu wird für jedes Wertetupel ein Grad der Erfüllung (ein Wert zwischen 0 und 1) berechnet. Basierend auf den Werten der atomaren Constraints wird der Erfüllungsgrad der Belegung (dies entspricht dem Minimum der Erfüllungsgrade) bestimmt. Eine Optimallösung ist hier eine Belegung mit maximalem Erfüllungsgrad. Eine typische Anwendung von Fuzzy-Constraints sind kritische Anwendungen, an die man möglichst vorsichtig herangeht. Das bedeutet, man betrachtet insb. die ungünstigsten Folgen oder Ergebnisse, wählt unter diesen aber die beste Lösung aus.

Generelle Soft-Constraint-Frameworks

Den speziellen Ausprägungen der Soft-Constraints unterliegt eine gemeinsame Struktur: Für eine gegebene Variablenbelegung eines Constraint-Problems wird für jedes Constraint angegeben, ob und zu welchem Anteil es erfüllt oder verletzt ist. Hieraus wird ein Grad der Erfüllung für die Belegung berechnet. Eine Optimallösung ist eine Lösung mit optimalem Erfüllungsgrad. Die Ansätze differieren somit im Wesentlichen hinsichtlich des Operators zur Kombination der Bewertungen von Wertetupeln und der Ordnung der Erfüllungsgrade.

Generelle Soft-Constraint-Frameworks, insb. Semiring-basierte Constraints und Valued Constraints (eine Darstellung und Diskussion beider findet sich in [8]), unterstützen eine verallgemeinerte Darstellungsform der spezialisierten Ansätze.

Semiring-basierte Constraints gründen auf einer Semiring-Struktur. Ihre Operationen erlauben die Formalisierung der Kombination von Erfüllungsgraden und der Ordnung über diesen. In einem Semiring-basierten Constraint-Netzwerk bildet jedes Constraint seine Wertetupel auf Werte des Semirings ab. Durch Instantiierung des Semirings erhält man die oben genannten speziellen Ausprägungen, aber auch die klassischen CSPs.

Valued Constraints sind eine alternative Beschreibungsform zum Semiring-basierten Constraint-Formalismus. Sie basieren auf einer Monoid-artigen Struktur und haben die gleiche Ausdrucksstärke wie total geordnete Semiring-basierte Constraints.

Partielle Constraint-Erfüllung und Constraint-Hierarchien

Zur Behandlung überspezifizierter Constraint-Probleme wurden Ende der 80er Jahre zunächst (und vor den verschiedenen Soft-Constraint-Konzepten) die partielle Constraint-Erfüllung und Constraint-Hierarchien entwickelt.

Partielle Constraint-Erfüllung [18] hat das Ziel zu einem überspezifizierten Constraint-Problem ein anderes Constraint-Netzwerk zu finden, das erfüllbar ist und sich dabei weitestgehend am ursprünglichen Problem orientiert. Dies wird durch Abschwächung der Constraints (engl. *constraint relaxation*) auf vier möglichen Wegen erreicht: durch die Erweiterung von Variablenmodänen, durch das Hinzufügen neuer Wertetupel zu Constraints, durch das Löschen von Constraints und durch das Löschen von Variablen. Die unvollständige Formalisierung des Ansatzes behindert seine Abbildung in ein generelles Soft-Constraint-Framework (siehe [40]).

Hierarchische Constraints [6, 7] verfolgen einen ähnlichen Ansatz wie gewichtete Constraints. Jedem Constraint wird eine Prioritätsstufe (z.B. weak – medium – strong – required) innerhalb einer totalen Ordnung zugeordnet. Anhand der Prioritätsstufen werden Kosten von Belegungen ermittelt und verglichen. Eine Belegung ist dabei nur dann eine Lösung, wenn sie mindestens

die Hard-Constraints erfüllt. Die Möglichkeiten zur Definition von Vergleichsoperatoren (engl. *comparators*) und zur Kostenberechnung gehen über die der generalisierten Frameworks hinaus (vgl. [40]).

7.5 Modellierung und Schließen mit temporalen Constraints

Viele Anwendungen der KI, hierunter Planungsprobleme, die Spezifikation reaktiver Systeme, Model Checking von Hard- und Software, die Verarbeitung natürlicher Sprache und die Analyse zeitabhängiger Daten z.B. in Expertensystemen beinhalten zeitliche Aspekte.

Kautz [29] unterteilt Verfahren und Ansätze zum zeitlichen Schließen in drei Kategorien: temporale Logiken, Aktionslogiken und algebraische Systeme. Während mit temporalen Logiken, wie bspw. LTL und CTL [4, 9, 44] temporale Beziehungen zwischen Aussagen formuliert und mit Model-Checking-Techniken überprüft werden können, werden bei Aktionslogiken wie dem Situationskalkül [39], dem Fluentenkalkül [58] und der dynamischen Logik [21] Abläufe durch Überführung von Zuständen in Folgezustände mittels Aktionen beschrieben, wobei temporales Wissen nur implizit auftritt. *Algebraische Systeme* hingegen beschreiben qualitativ oder quantitativ Relationen zwischen Zeitpunkten oder Zeitintervallen. In dieser Weise beschriebene Probleme können als temporale Constraint-Probleme betrachtet und behandelt werden. Beispielsweise kann die Konsistenz der Beschreibungen zeitlicher Beziehungen überprüft werden, konsistente Belegungen können berechnet und Zeitalüfe optimiert werden. Zeit wird dabei als kontinuierliche lineare Struktur aufgefasst.

Temporal Constraint Programming

Beim *Temporal Constraint Programming* werden drei wesentliche Forschungsrichtungen unterschieden:

1. das Schließen auf der Basis metrischer Information (sog. quantitative Ansätze),
2. qualitative Ansätze basierend auf Allens Intervall-Algebra und
3. gemischte Ansätze (z.B. [31, 38]) basierend auf metrischen und qualitativen Constraints mit dem Ziel einer verbesserten Ausdrucksstärke bei Vermeidung einer Erhöhung der Komplexität.

Wir gehen kurz auf quantitative und qualitative Ansätze ein und verweisen auf weiterführende Literatur, hierunter [19, 32, 57].

Quantitative Algebren erlauben das zeitliche Schließen mit metrischer Information. Als primitive temporale Objekte werden Zeitpunkte als reelle (oder rationale) Zahlen betrachtet. Constraints beschreiben reellwertige Intervalle. Die Algorithmen zur Behandlung solcher Constraints nutzen die metrischen Eigenschaften der Domänen. In [19, 57] werden effizient handhabbare Problemklassen diskutiert, hierunter das STP (Simple Temporal Problem) und seine Erweiterungen sowie Verfahren zur Überprüfung der Erfüllbarkeit und zur Berechnung von Lösungen.

<i>a before b</i>		<i>a finishes b</i>	
<i>b after a</i>		<i>b finished-by a</i>	
<i>a meets b</i>		<i>a overlaps b</i>	
<i>b met-by a</i>		<i>b overlapped-by a</i>	
<i>a during b</i>		<i>a starts b</i>	
<i>b includes a</i>		<i>b started-by a</i>	
<i>a = b</i>			

Abbildung 7.5: Atomare Constraints der IA.

Bei den *qualitativen Ansätzen* werden Constraints als Relationen zwischen Intervallen oder Zeitpunkten definiert. Die bekannteste Variation ist Allens Intervall-Algebra (IA) [1]. Andere qualitative Ansätze betreffen Subalgebren von IA, hierunter die qualitative Punkt-Algebra (PA) [59, 64] und die NB-Algebra [42], eine maximale Subalgebra von IA, bei der die Erfüllbarkeit in polynomieller Zeit entscheidbar ist.

Die zeitlichen Primitive von Allens Intervall-Algebra sind reell- oder rational-wertige Intervalle. Abbildung 7.5 zeigt die 13 primitiven Relationen oder Constraints über Paaren von Intervallen, sie sind paarweise disjunkt. Die Vereinigung der 2^{13} Kombinationen der atomaren Constraints bildet Allens Intervall-Algebra.

Mit Hilfe des Pfadkonsistenz-Algorithmus können eine minimale Constraintmenge ermittelt und unerfüllbare Constraint-Netzwerke erkannt werden. Hierbei werden die Relationen zwischen je zwei Intervallen durch Komposition propagiert. Die Kompositionstabelle ist in [1] angegeben. Da die Pfadkonsistenz-Methode unvollständig ist, können auch unerfüllbare Constraintmengen als pfadkonsistent erkannt werden. Die Herstellung der globalen Konsistenz und die Überprüfung der Erfüllbarkeit eines temporalen Constraint-Netzwerks der IA ist (im Gegensatz zur Pfadkonsistenz-Methode mit kubischem Aufwand) allerdings NP-schwierig [65].

Beispiel 7.5.1. Während wir durch Komposition der Constraints *a before b* und *b before c* die Relation *a before c* folgern können (vgl. Abbildung 7.6(a)), führt die Propagation der Constraints *a finishes b* und *b overlaps c* zu einer Disjunktion möglicher Alternativen zwischen *a* und *c*: *a* {*overlaps*, *starts*, *during*} *c*.

Abbildung 7.6(b) stellt die alternativen Fälle dar.

Temporale Constraints zur Modellierung reaktiver Systeme

Zeitliche Constraints spielen auch im Kontext der Beschreibung reaktiver Systeme eine Rolle. Entsprechende Kalküle und Sprachen basieren hierbei auf einer Erweiterung der nebenläufigen Constraint-Programmierung (vgl. Abschnitt 7.3.4) um temporale Konzepte.

CCP erlaubt nur die Darstellung positiver Information. Lässt sich ein *ask*-Constraint nicht aus dem aktuellen Constraint-Speicher folgern, dann wird der anfragende Prozess blockiert und wartet möglicherweise unendlich lange. Konzepte wie Time-Out und Präemption, die an einer solchen Stelle zur Modellierung praktischer Anwendungen sinnvoll und relevant wären, lassen sich in CCP nicht darstellen. Daher haben sich verschiedene temporale Erweiterungen des CCP-Modells herausgebildet, einen Überblick gibt [43].

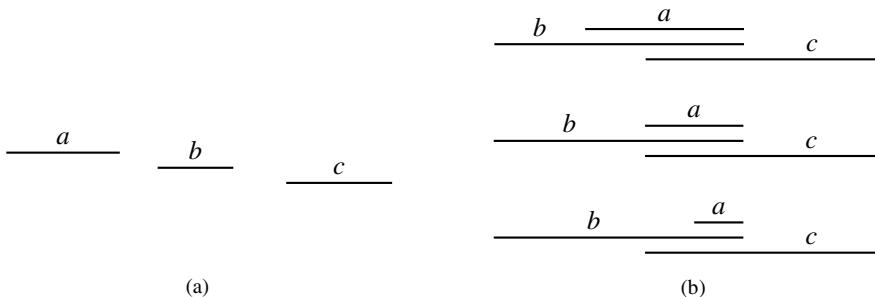


Abbildung 7.6: Komposition von IA-Constraints: (a) (*a* before *b*) und (*b* before *c*), (b) (*a* finishes *b*) und (*b* overlaps *c*).

7.6 Zusammenfassung

Die constraint-basierte Programmierung hat sich von ihren Ursprüngen als Erweiterung der logischen Programmierung hin zu einem etablierten Bereich der Künstlichen Intelligenz entwickelt. Ursache des Erfolgs der Constraint-Programmierung sind vor allem ihre vielfältigen Anwendungsbereiche in Praxis und Wissenschaft und ihre Integration mit Methoden der klassischen KI und des Operations Research.

Dieser Artikel gibt einen Überblick und eine Einführung in den Bereich der Constraints, ihrer Lösungsverfahren, Programmiermethodik und spezieller Ausprägungen wie Soft-Constraints und temporale Constraints. Andere Themen, hierunter weitere Domänen und ihre Lösungsmethoden, die Kombination von Lösungsverfahren, Implementierungstechniken, die verteilte Constraint-Programmierung sowie Anwendungen haben wir ausgelassen. Das Handbook of Constraint Programming [47] gibt eine weitreichende Darstellung.

Constraints, constraint-basierte Programmierung und Verfahren sind Thema vieler Konferenzen und Workshops, insb. der *International Conference on Principles and Practice of Constraint Programming (CP)* und der *International Conference on Integration of AI and OR Techniques in Constraint Programming (CPAI OR)* sowie weiterer KI-Konferenzen. Die Zeitschrift *Constraints* (Springer-Verlag) vertritt das Themengebiet ebenso wie *Theory and Practice of Logic Programming* (Cambridge University Press). Seit 2005 ist die *Association of Constraint Programming (ACP)* eine Vereinigung von Wissenschaftlern mit Themengebieten im Bereich Constraints.

Seit der letzten Dekade ist die Constraint-Programmierung auch im Informatik-Studienplan vieler Universitäten vertreten, weiterhin sind eine Reihe von Büchern im Lehr- und Forschungskontext erschienen, darunter [2, 10, 12, 25, 41, 47].

Literaturverzeichnis

- [1] Allen, James F.: *Maintaining Knowledge about Temporal Intervals*. Communications of the ACM, 26(11):832–843, 1983.
- [2] Apt, Krzysztof: *Principles of Constraint Programming*. Cambridge University Press, 2003.
- [3] Apt, Krzysztof und Mark Wallace: *Constraint Logic Programming using Eclipse*. Cambridge University Press, 2007.
- [4] Bérard, Béatrice, Michel Bidoit, Alain Finkel, François Laroussinie, Antoine Petit, Laure Petrucci, Philippe Schnoebelen und Pierre McKenzie: *Systems and Software Verification*. Springer, 2010.
- [5] Bessiere, Christian: *Constraint Propagation*. In: Rossi, Francesca et al. [47], Kapitel 3, Seiten 29–83.
- [6] Borning, Alan, Bjørn N. Freeman-Benson und Molly Wilson: *Constraint Hierarchies*. Lisp and Symbolic Computation, 5(3):223–270, 1992.
- [7] Borning, Alan, Michael J. Maher, Amy Martindale und Molly Wilson: *Constraint Hierarchies and Logic Programming*. In: Levi, Giorgio und Maurizio Martelli (Herausgeber): *International Conference on Logic Programming (ICLP)*, Seiten 149–164. MIT Press, 1989.
- [8] Bistarelli, Stefano, Ugo Montanari, Francesca Rossi, Thomas Schiex, Gérard Verfaillie und Hélène Fargier: *Semiring-Based CSPs and Valued CSPs: Frameworks, Properties, and Comparison*. Constraints, 4(3):199–240, 1999.
- [9] Clarke, Edmund M. und E. Allen Emerson: *Design and Synthesis of Synchronization Skeletons using Branching Time Temporal Logic*. In: *Logics of Programs Workshop*, Band 131 der Reihe *Lecture Notes in Computer Science*, Seiten 52–71. Springer, 1981.
- [10] Dechter, Rina: *Constraint Processing*. Morgan Kaufmann, 2003.
- [11] Dubois, Didier, Hélène Fargier und Henri Prade: *The Calculus of Fuzzy Restrictions as a Basis for Flexible Constraint Satisfaction*. In: *2nd IEEE International Conference on Fuzzy Systems*, Band 2, Seiten 1131–1136, 1993.
- [12] Frühwirth, Thom und Slim Abdennadher: *Essentials of Constraint Programming*. Springer Verlag, 2003.
- [13] Fargier, Hélène und Jérôme Lang: *Uncertainty in Constraint Satisfaction Problems: a Probabilistic Approach*. In: Clarke, Michael, Rudolf Kruse und Serafín Moral (Herausgeber): *Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU)*, Band 747 der Reihe *Lecture Notes in Computer Science*, Seiten 97–104. Springer, 1993.
- [14] Freuder, Eugene C. und Alan K. Mackworth: *Constraint Satisfaction: An Emerging Paradigm*. In: Rossi, Francesca et al. [47], Kapitel 2, Seiten 13–27.
- [15] Freuder, Eugene C.: *Synthesizing Constraint Expressions*. Communications of the ACM, 21(11):958–966, 1978.
- [16] Freuder, Eugene C.: *In Pursuit of the Holy Grail*. Constraints, 2(1):57–61, 1997.
- [17] Frühwirth, Thom: *Constraint Handling Rules*. Cambridge University Press, 2009.
- [18] Freuder, Eugene C. und Richard J. Wallace: *Partial Constraint Satisfaction*. Artificial Intelligence, 58(1–3):21–70, 1992.
- [19] Gennari, Rosella: *Temporal Reasoning and Constraint Programming. A Survey*. CWI Quarterly, 11(2&3):163–214, 1998. CWI, University of Amsterdam.

- [20] *Global Constraint Catalog.*
<http://www.emn.fr/z-info/sdemasse/gccat/>. Zuletzt besucht 2012-07-11.
- [21] Harel, David: *First-Order Dynamic Logic*, Band 68 der Reihe *Lecture Notes in Computer Science*. Springer, 1979.
- [22] Hentenryck, Pascal Van: *The OPL Optimization Programming Language*. The MIT Press, 1999.
- [23] Hofstedt, Petra: *Constraint-Based Object-Oriented Programming*. IEEE Software, 27(5):53–56, 2010.
- [24] Hofstedt, Petra: *Multiparadigm Constraint Programming Languages*. Springer, 2011.
- [25] Hofstedt, Petra und Armin Wolf: *Einführung in die Constraint-Programmierung*. Springer, 2007.
- [26] IBM: *IBM ILOG CPLEX Optimization Studio Modeling*.
<http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/modeling/>. Zuletzt besucht 2012-07-11.
- [27] Jaffar, Joxan und Jean-Louis Lassez: *Constraint Logic Programming*. In: *14th ACM Symposium on Principles of Programming Languages (POPL)*, Seiten 111–119, 1987.
- [28] Jaffar, Joxan und Michael J. Maher: *Constraint Logic Programming: A Survey*. Journal of Logic Programming, 19/20:503–581, 1994.
- [29] Kautz, Henry: *Temporal Reasoning*. In: Wilson, Robert A. und Frank C. Keil (Herausgeber): *The MIT Encyclopedia of the Cognitive Science*, Seiten 829–831. MIT Press, Cambridge, 1999.
- [30] König, Thomas und Petra Hofstedt: *Constraint-basierte Programmiersprachen – Von den Ursprüngen in der logischen Programmierung bis zu heutigen Anwendungen*. Künstliche Intelligenz – KI, 26(1):47–54, 2012.
- [31] Kautz, Henry A. und Peter B. Ladkin: *Integrating Metric and Qualitative Temporal Reasoning*. In: Dean, Thomas L. und Kathleen McKeown (Herausgeber): *9th National Conference on Artificial Intelligence (AAAI), Volume 1*, Seiten 241–246. AAAI Press / The MIT Press, 1991.
- [32] Koubarakis, Manolis: *Temporal CSPs*. In: Rossi, Francesca et al. [47], Kapitel 19, Seiten 665–697.
- [33] Kuchcinski, Krzysztof und Radosław Szymanek: *JaCoP Library User's Guide*, 2011. version 3.1.
- [34] Laurière, Jean-Louis: *A Language and a Program for Stating and Solving Combinatorial Problems*. Artificial Intelligence, 10(1):29–127, 1978.
- [35] Laburthe, François und Narendra Jussien: CHOCO SOLVER. Ecole des mines de Nantes, 2012.
- [36] López-Ortiz, Alejandro, Claude-Guy Quimper, John Tromp und Peter van Beek: *A Fast and Simple Algorithm for Bounds Consistency of the AllDifferent Constraint*. In: Gottlob, Georg und Toby Walsh (Herausgeber): *Joint Conference on Artificial Intelligence (IJCAI)*, Seiten 245–250. Morgan Kaufmann, 2003.
- [37] Mackworth, Alan K.: *Consistency in Networks of Relations*. Artificial Intelligence, 8(1):99–118, 1977.
- [38] Meiri, Itay: *Combining Qualitative and Quantitative Constraints in Temporal Reasoning*. Artificial Intelligence, 87(1–2):343–385, 1996.
- [39] McCarthy, John und Patrick J. Hayes: *Some Philosophical Problems from the Standpoint of Artificial Intelligence*. Machine Intelligence, 4:463–502, 1969.

- [40] Meseguer, Pedro, Francesca Rossi und Thomas Schiex: *Soft Constraints*. In: Rossi, Francesca et al. [47], Kapitel 9, Seiten 281–328.
- [41] Marriott, Kim und Peter J. Stuckey: *Programming with Constraints: An Introduction*. The MIT Press, 1998.
- [42] Nebel, Bernhard und Hans-Jürgen Bürckert: *Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra*. Journal of the ACM, 42(1):43–66, 1995.
- [43] Olarte, Carlos, Camilo Rueda und Frank D. Valencia: *Concurrent Constraint Programming: Calculi, Languages, and Emerging Applications*. The ALP (Association for Logic Programming) Newsletter, 21(2–3), August 2008.
- [44] Pnueli, Amir: *The Temporal Logic of Programs*. In: *18th IEEE Symp. Foundations of Computer Science (FOCS)*, Seiten 46–57, 1977.
- [45] Prosser, Patrick: *Hybrid Algorithms for the Constraint Satisfaction Problem*. Computational Intelligence, 9:268–299, 1993.
- [46] Puget, Jean-François: *A Fast Algorithm for the Bound Consistency of alldiff Constraints*. In: Mostow, Jack und Chuck Rich (Herausgeber): *Artificial Intelligence and Innovative Applications of Artificial Intelligence (AAAI)*, Madison, Wisconsin, USA, Seiten 359–366. AAAI Press / The MIT Press, 1998.
- [47] Rossi, Francesca, Peter van Beek und Toby Walsh (Herausgeber): *Handbook of Constraint Programming*. Elsevier, 2006.
- [48] Régin, Jean-Charles: *A Filtering Algorithm for Constraints of Difference in CSPs*. In: Hayes-Roth, Barbara und Richard E. Korf (Herausgeber): *12th National Conference on Artificial Intelligence, Seattle, WA, USA (AAAI 94)*, Volume 1, Seiten 362–367. AAAI Press / The MIT Press, 1994.
- [49] Ruttkay, Zsófia: *Fuzzy Constraint Satisfaction*. In: *Third IEEE International Conference on Fuzzy Systems*, Seiten 1263–1268. IEEE Press, 1994.
- [50] Saraswat, Vijay A.: *Concurrent Constraint Programming*. MIT Press, 1993.
- [51] Stuckey, Peter J., Ralph Becket, Sebastian Brand, Mark Brown, Thibaut Feydy, Julien Fischer, Maria Garcia de la Banda, Kim Marriott und Mark Wallace: *The Evolving World of MiniZinc*. In: Frisch, Alan M. und Jimmy Lee (Herausgeber): *International Workshop on Constraint Modelling and Reformulation (ModRef)*, Seiten 156–170, 2009.
- [52] Schiex, Thomas: *Possibilistic Constraint Satisfaction Problems or How to Handle Soft Constraints?* In: Dubois, Didier und Michael P. Wellman (Herausgeber): *Uncertainty in Artificial Intelligence (UAI)*, Seiten 268–275, 1992.
- [53] Sabin, Daniel und Eugene C. Freuder: *Understanding and Improving the MAC Algorithm*. In: Smolka, Gert (Herausgeber): *Principles and Practice of Constraint Programming (CP)*, Band 1330 der Reihe *Lecture Notes in Computer Science*, Seiten 167–181. Springer, 1997.
- [54] Shapiro, Linda G. und Robert M. Haralick: *Structural Descriptions and Inexact Matching*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 3:504–519, 1981.
- [55] Schrijvers, Tom, Peter Stuckey und Philip Wadler: *Monadic Constraint Programming*. Journal of Functional Programming, 19(6):663–697, 2009.
- [56] Schulte, Christian, Guido Tack und Mikael Z. Lagerkvist: *Modeling and Programming with Gecode*, 2012. Gecode 3.7.3.
- [57] Schwalb, Eddie und Lluís Vila: *Temporal Constraints: A Survey*. Constraints, 3(2/3):129–149, 1998.

- [58] Thielscher, Michael: *Introduction to the Fluent Calculus*. Electronic Transactions on Artificial Intelligence, 2(3–4):179–192, 1998.
- [59] Beek, Peter van: *Reasoning about Qualitative Temporal Information*. Artificial Intelligence, 58(1–3):297–326, 1992.
- [60] Beek, Peter van: *Backtracking Search Algorithms*. In: Rossi, Francesca et al. [47], Kapitel 4, Seiten 85–134.
- [61] Hoeve, Willem-Jan van: *The Alldifferent Constraint: A Survey*. In: *Sixth Annual Workshop of the ERCIM Working Group on Constraints*, 2001. Prague.
- [62] Hoeve, Willem-Jan van und Irit Katriel: *Global Constraints*. In: Rossi, Francesca et al. [47], Kapitel 6, Seiten 169–208.
- [63] Van Hentenryck, Pascal und Laurent Michel: *Constraint-Based Local Search*. MIT Press, 2005.
- [64] Vilain, Marc B. und Henry A. Kautz: *Constraint Propagation Algorithms for Temporal Reasoning*. In: Kehler, Tom (Herausgeber): *5th National Conference on Artificial Intelligence (AAAI)*, Band 1, Seiten 377–382. Morgan Kaufmann, 1986.
- [65] Vilain, Marc B., Henry A. Kautz und Peter van Beek: *Constraint Propagation Algorithms for Temporal Reasoning: A revised Report*. In: deKleer, J. und D. Weld (Herausgeber): *Readings in Qualitative Reasoning about Physical Systems*, Seiten 373–381. Morgan-Kaufman, 1989.

8 Unsicheres und vages Wissen

Christian Borgelt, Christian Braune, Heiko Timm und Rudolf Kruse

Wissensbasierte Systeme sind Programme, die auf der Grundlage von Wissen über einen bestimmten Anwendungsbereich Schlussfolgerungen ziehen können, und die so einem Benutzer helfen, ein Problem zu lösen oder eine Entscheidung zu treffen. Diese Systeme lassen sich kaum auf die Verarbeitung absolut präzisen und sicheren Wissen beschränken. Denn schon eine Allerweltsaussage wie „Alle Vögel können fliegen.“ ist nicht sicher, da es ja auch Pinguine und Strauß gibt, und eine Aussage wie „Sabine ist zu Hause.“ ist vage, da ihr genauer Aufenthaltsort (Wohnzimmer, Arbeitszimmer, Küche usw.) unbestimmt bleibt. Aber obwohl diese Sätze unsicher und vage sind, sind sie dennoch nicht unbrauchbar: Menschen können aus ihnen ohne große Schwierigkeiten Schlussfolgerungen ziehen. So wird etwa ein Arzt bestimmte Krankheiten ausschließen, wenn er erfährt, dass ein Patient kein Fieber hat, auch wenn er die genaue Körpertemperatur nicht (oder nicht sicher) kennt. Folglich müssen, wenn wissensbasierte Systeme eine echte Unterstützung des Menschen sein sollen, Methoden zur Darstellung von unsicherem und vagem Wissen in einem Rechner und Methoden zum automatischen Schlussfolgern mit solchem Wissen bereitgestellt werden. Eine Auswahl solcher Methoden stellen wir in diesem Kapitel vor.

Nachdem wir in Abschnitt 8.1 einige Begriffe eingeführt haben, betrachten wir in Abschnitt 8.2 einen recht alten, aber sehr naheliegenden Ansatz: logische Aussagen und Regeln – die klassischen Elemente von Schlussfolgerungssystemen – werden durch sogenannte *Sicherheitsfaktoren*. So wird ein System zum Schlussfolgern unter Unsicherheit geschaffen. Es zeigt sich jedoch, dass ein solcher Ansatz sehr starke, oft nicht erfüllte Unabhängigkeitsannahmen macht, die zu falschen Ergebnissen führen können.

Da der i.w. heuristisch motivierte Sicherheitsfaktorenansatz hauptsächlich an den starken impliziten Unabhängigkeitsannahmen scheitert, liegt es nahe, zunächst die geltenden Abhängigkeiten und Unabhängigkeiten zu modellieren und den Schlussfolgerungsmechanismus dann so zu wählen, dass er auf diese Abhängigkeiten und Unabhängigkeiten explizit Rücksicht nimmt. Ein sehr populärer Ansatz in dieser Richtung, den wir in Abschnitt 8.3 vorstellen, sind die *probabilistischen Schlussfolgerungsnetze*, die Zerlegungen hochdimensionaler Wahrscheinlichkeitsverteilungen beschreiben und von denen Bayes-Netze und Markow-Netze am bekanntesten sind.

Als vorherrschender Ansatz zum Umgang mit Vagheit hat sich, vor allem wegen ihres erfolgreichen Einsatzes in Konsumgütern, die von Lotfi Zadeh (*1921) begründete Fuzzy-Mengentheorie bzw. Fuzzy-Logik etabliert. Sie arbeitet mit sogenannten *linguistischen Variablen*, deren Werten *unscharfe Mengen* (*Fuzzy-Mengen*) zugeordnet sind. In *Fuzzy-Regelsystemen*, die wir in Abschnitt 8.4 behandeln, werden mit Hilfe von linguistischen Variablen formulierte Regeln benutzt, um Funktionen (oder allgemein Abhängigkeiten) vage zu beschreiben. Die den Werten der linguistischen Variablen zugeordneten Fuzzy-Mengen lassen sich als Interpolations- oder Glättungsmechanismus deuten.

8.1 Begriffe

In diesem Abschnitt führen wir einige Begriffe ein, die wir in diesem Kapitel benötigen. In Abschnitt 8.1.1 erläutern wir, was wir unter dem Kernbegriff des Wissens verstehen und betrachten in Abschnitt 8.1.2 drei wesentliche Eigenschaften, die Wissen haben kann, nämlich Impräzision, Unsicherheit und Vagheit. In Abschnitt 8.1.3 erklären wir, was es bedeutet, mit unsicherem oder vagem Wissen Schlussfolgerungen zu ziehen, und welche Probleme beim Entwurf eines wissensbasierten Systems, das mit solchem Wissen umgehen können soll, zu lösen sind. In den Abschnitten 8.1.4 und 8.1.5 wenden wir uns dann zwei speziellen Konzepten zu, die zur Darstellung von Unsicherheit bzw. Vagheit verwendet werden, nämlich den Begriffen der Wahrscheinlichkeit bzw. der Fuzzy-Menge.

8.1.1 Wissen

In diesem Abschnitt fassen wir *Wissen* auf als eine Zuordnung von bestimmten Werten, etwa von Wahrheitswerten, Wahrscheinlichkeiten, Possibilitätsgraden o.ä., zu Aussagen, Ereignissen, Zuständen etc. Die Unterscheidung von Aussagen, Ereignissen, Zuständen etc. kann man ggf. fallenlassen, denn soll Wissen der intersubjektiven Prüfung und einer sorgfältigen Analyse zugänglich gemacht werden, so muss es in einer Sprache formuliert werden. Man kann sich daher auf Aussagen beschränken, die dann mit den Ereignissen oder Zuständen, die sie beschreiben, identifiziert werden. Je nach Kalkül, das zur Darstellung von Wissen verwendet wird, findet man jedoch auch die anderen Begriffe. So spricht man etwa üblicherweise von der Wahrscheinlichkeit von Ereignissen.

Oft – speziell bei einer Beschränkung auf Wahrheitswerte – findet man Wissen auch definiert als die Menge der (als wahr) bekannten Aussagen. Man beachte, dass eine solche Definition nicht im Widerspruch zu unserer Charakterisierung steht. Es werden lediglich die Wahrheitswerte auf bestimmte Weise zugeordnet, nämlich durch eine Auflistung oder anderweitige Auszeichnung der wahren Aussagen. Wir weichen von dieser Definition ab, weil sie zur Behandlung von unsicherem und vagem Wissen nicht brauchbar ist. Denn man muss hier genauer unterscheiden als nur zwischen wahr und falsch und folglich reicht eine einfache Menge zur Charakterisierung nicht aus.

8.1.2 Impräzision, Unsicherheit und Vagheit

Wir unterscheiden drei Eigenschaften, die man Wissen zuordnen kann: *Impräzision*, *Unsicherheit* und *Vagheit*. Während sich Impräzision noch mit klassischer Logik behandeln lassen, braucht man zur Darstellung von Unsicherheit und Vagheit andere Kalküle.

Impräzision. Aussagen wie „Der Ball ist grün oder blau oder türkis.“ oder „Die Geschwindigkeit des Autos lag zwischen 50 und 60 km/h.“ nennen wir *impräzise*. Sie sind impräzise, weil sie nicht nur einen Wert einer Eigenschaft angeben sondern eine *Menge von Alternativen*. Im Gegensatz dazu nennen wir Aussagen, die nur einen Wert für eine Eigenschaft angeben, *präzise*. Ein Beispiel einer präzisen Aussage ist „Der Patient hat 39.3° Fieber.“¹

¹ Man könnte einwenden, dass es in dieser Aussage so etwas wie implizite Impräzision gibt, da die Temperatur nur mit endlicher Genauigkeit angegeben ist, d.h., eigentlich sind alle Temperaturen zwischen 39.25° und 39.35° möglich. Wir vernachlässigen hier jedoch solche Spitzfindigkeiten.

Es ist klar, dass sich Impräzision mit den Mitteln der klassischen symbolischen Logik behandeln lässt: Endliche Mengen von Alternativen kann man einfach durch Disjunktionen darstellen, in denen die Alternativen aufgezählt werden, und Intervalle können beschrieben werden, indem man Vergleichsprädikate \geq und/oder \leq einführt.

Unsicherheit. Wenn man Aussagen wie die obigen Beispiele liest, setzt man i.a. implizit voraus, dass die Aussagen *sicher* sind, d.h., dass alle Alternativen, die in den Aussagen nicht genannt sind, definitiv ausgeschlossen werden können. So nimmt man etwa an, dass der Ball bestimmt nicht rot ist und dass das Auto auf jeden Fall schneller als 40 km/h fuhr. Falls solche, in der betrachteten Aussage nicht genannten Alternativen *nicht* ausgeschlossen werden können, nennen wir die Aussage *unsicher*. Denn die Aussage könnte ja falsch sein, da es möglich ist, dass eine dieser nicht genannten Alternativen die wahre Situation beschreibt. Man beachte, dass sowohl präzise als auch impräzise Aussagen unsicher sein können. Was eine Aussage sicher oder unsicher macht, ist, ob alle möglichen Alternativen in ihr genannt sind oder nicht.

Beschränkt man sich darauf, Aussagen als sicher oder unsicher zu *kennzeichnen*, kommt man ebenfalls noch mit den Mitteln der Logik aus: In der sogenannten *Modallogik* werden zu diesem Zweck die *Modalitäten* „notwendig“ und „möglich“ eingeführt. Eine einfache Kennzeichnung ist jedoch oft nicht ausreichend, da sie keine Hilfestellung mehr bietet, wenn man sich zwischen mehreren unsicheren Aussagen zu entscheiden hat. In einem solchen Fall braucht man Kalküle, in denen man mindestens *Präferenzen* zwischen unsicheren Aussagen ausdrücken kann oder in denen sich sogar die Sicherheit einer Aussage in Form eines Sicherheits- oder Vertrauensgrades *quantifizieren* lässt.

Vagheit. Die meisten Begriffe der Alltagssprache sind *vage*: Wenn wir sagen, Peter sei *groß* oder an einem Sommertag sei es *heiß*, dann meinen wir keine bestimmte Körpergröße und keine bestimmte Temperatur, ja nicht einmal ein Intervall wie bei impräzisen Aussagen. Die Bereiche der Körpergrößen und Temperaturen, die als *groß* bzw. *heiß* bezeichnet werden können, sind nicht scharf begrenzt. Zwar gibt es Werte der Körpergröße, die sicher als *groß* bezeichnet werden können, z.B. 220 cm, und solche, die sicher nicht als *groß* bezeichnet werden können, z.B. 140 cm. Genauso gibt es Temperaturen, die sicher *heiß* sind, z.B. 35° C, und solche, die es sicher nicht sind, z.B. 5° C. Doch zwischen diesen Körpergrößen und Temperaturen, für die klar entschieden werden kann, ob der Begriff anwendbar ist oder nicht, liegt eine *Penumbra* (lat. für *Halbschatten*) von Werten, die nicht eindeutig zugeordnet werden können.

Zwar kann man stets die Penumbra beseitigen, indem man in einem Gewaltakt eine scharfe Grenze zieht, doch ist es, wie der Philosoph [58] schrieb:

Wenn Einer eine scharfe Grenze zöge, so könnte ich sie nicht als die anerkennen, die ich auch schon immer ziehen wollte, oder im Geist gezogen habe. Denn ich wollte gar keine ziehen. Man kann dann sagen: sein Begriff ist nicht der gleiche wie der meine, aber ihm verwandt. Und die Verwandtschaft ist die zweier Bilder, deren eines aus unscharf begrenzten Farbfleckken, das andere aus ähnlich geformten und verteilten, aber scharf begrenzten, besteht. Die Verwandtschaft ist dann ebenso unleugbar wie die Verschiedenheit.

Man wird daher Wege finden müssen, mit der Vagheit sprachlicher Ausdrücke umzugehen, ohne sie auf Intervalle (und damit auf impräzise Aussagen) zu reduzieren. Es ist klar, dass die klassische Logik hierzu nicht ausreicht, da sie nur scharfe Grenzen kennt.

Wir betonen hier noch einmal, dass *vage* Aussagen nicht notwendig Aussagen minderer Qualität und schon gar nicht nutzlos sind. [58] schreibt dazu:

Wenn ich Einem sage „Halte Dich ungefähr hier auf!“ – kann denn diese Erklärung nicht vollkommen funktionieren? Und kann jede andere nicht auch versagen?

„Aber ist die Erklärung nicht doch unexakt?“ – Doch; warum soll man sie nicht unexakt nennen? Verstehen wir aber nur, was „unexakt“ bedeutet! Denn es bedeutet nun nicht „unbrauchbar“. Und überlegen wir uns doch, was wir, im Gegensatz zu dieser Erklärung, eine „exakte“ Erklärung nennen! Etwa das Abgrenzen eines Bezirks durch einen Kreidestrich? Da fällt uns gleich ein, dass der Strich eine Breite hat. Exakter wäre also eine Farbgrenze. Aber hat denn diese Exaktheit hier noch eine Funktion; läuft sie nicht leer? Und wir haben ja auch noch nicht bestimmt, was als Überschreiten dieser scharfen Grenze gelten soll; wie, mit welchen Instrumenten, es festzustellen ist. Usw.

8.1.3 Schlussfolgern

Die wesentliche Eigenschaft des Schlussfolgerns ist, dass ein bestimmter Typ von Wissen – Wissen über Wahrheit, Wahrscheinlichkeit, (Grad der) Möglichkeit – von gegebenen Aussagen, Ereignissen, Zuständen etc. auf andere Aussagen, Ereignisse, Zustände etc. übertragen wird. Z.B. wird in einem (deduktiven) logischen Schluss Wissen über die Wahrheit der Prämissen auf die Konklusion übertragen; in der Wahrscheinlichkeitstheorie geht die Wahrscheinlichkeit eines Ereignisses in die Berechnung der Wahrscheinlichkeit anderer, in Beziehung stehender Ereignisse ein und wird so in gewisser Weise auf diese anderen Ereignisse übertragen usw.

Damit diese Übertragung möglich ist, bedarf es dreier Dinge: Wissen, von dem man ausgehen kann (z.B. das Wissen, dass eine bestimmte Aussage wahr ist), Wissen, das einen Weg für die Übertragung bereitstellt (z.B. eine Implikation, von der die gegebene Aussage das Antezedens ist), und ein Mechanismus, um dem Weg folgen zu können (z.B. die Schlussregel des *Modus Ponens* mit dem die Wahrheit des Konsequens einer Implikation, deren Antezedens bekannt ist, erschlossen werden kann). Nur wenn alle drei gegeben sind und zusammenpassen, kann eine Schlussfolgerung gezogen werden. Natürlich muss die Übertragung nicht immer direkt sein. In der klassischen Logik kann man z.B. Argumente verketten, indem die Konklusion eines Argumentes zur Prämisse eines anderen macht. Bisweilen sind mehrere solcher Schritte nötig, um eine gewünschte Konklusion abzuleiten.

Aus dieser Beschreibung können wir ablesen, was die wesentlichen Probleme der Modellierung von Schlussfolgerungsprozessen sind: Sie bestehen darin, die Pfade aufzufinden, auf denen Wissen übertragen werden kann, und geeignete Mechanismen bereitzustellen, mit denen diese Pfade verfolgt werden können. (Im Gegensatz dazu ist das Wissen, von dem man ausgeht, i.a. unmittelbar verfügbar, z.B. als Ergebnis von Beobachtungen.)

Sicher ist die klassische Logik [6, 19, 45, 46, 48] der bekannteste Ansatz, um diese Probleme zu lösen. Sie stellt einfache Schlussregeln bereit, durch die es möglich wird, Wissen über einen bestimmten Anwendungsbereich als Menge von Aussagen darzustellen, die nur wenige atomare Aussagen oder Prädikate enthalten und die die Pfade sind, auf denen durch Beobachtungen gewonnenes Wissens übertragen werden kann. Alle Folgerungen lassen sich (automatisch) mit Hilfe der Schlussregeln ableiten, auch wenn dies eine mitunter aufwendige Suche nach den geeigneten Argumenten erfordert [20]. Es liegt nahe, die im Bereich der Logik bereits geleistete Arbeit zu nutzen, indem man nach einer Erweiterung sucht, in der man statt mit Wahrheitswerten *wahr* und *falsch*, die mit 1 und 0 oder mit 1 und -1 identifiziert werden können, mit

Sicherheitsfaktoren arbeitet, um unsicheres Wissen zu behandeln. Dies ist der Ansatz, den wir in Abschnitt 8.2 betrachten werden.

Die Analyse des Sicherheitsfaktorenansatzes zeigt jedoch, dass er sehr starke, in der Praxis oft nicht erfüllte Unabhängigkeitsannahmen macht. Insbesondere kann die Einschätzung einer Konklusion von der Reihenfolge der Schlussfolgerungen abhängen und es kann, wenn ein beobachtetes Faktum über mehrere Implikationsketten auf die gleiche Konklusion führt, zu einer fehlerhaften Einschätzung der Sicherheit dieser Konklusion kommen. Man beachte, dass die Unabhängigkeitsannahmen in der Struktur der zweiwertigen Logik liegen. Bei einer Beschränkung auf die Sicherheitsfaktoren 0 und 1 bzw. -1 und 1 (für falsch und wahr) sind sie notwendigerweise erfüllt. Erst wenn man Zwischenwerte zulässt, können sie sich negativ auswirken. Dann jedoch können sie, wenn die implizit vorausgesetzten Unabhängigkeiten nicht gelten, zu falschen Ergebnissen führen.

Folglich müssen eventuell vorhandene Abhängigkeiten explizit erfasst und berücksichtigt werden. Da die Aussagen, die zum Schlussfolgern über einen bestimmten Anwendungsbereich benutzt werden, meist die Form von Implikationen haben (oder zu Implikationen äquivalent sind), bietet es sich an, sie dazu durch einem (gerichteten) Graphen darzustellen. Dieser Graph gibt die Argumentketten wieder, die aus den Implikationen und beobachteten Fakten gebildet werden können, indem die Terme des Antezedens einer Implikation mit dem Konsequens durch gerichtete Kanten verbunden werden. Aus einem solchen Graphen lässt sich dann leicht ablesen, welche Aussagen auf welche Schlussfolgerungen Einfluss haben und welche Abhängigkeiten folglich bestehen. Außerdem zeigt er direkt die möglichen Übertragungswege für aus Beobachtungen gewonnenes Wissen und hilft daher auch, den Schlussfolgerungsprozess effizienter zu machen.

Mit dieser Überlegung haben wir bereits den ersten Schritt in Richtung auf die *probabilistischen Schlussfolgerungsnetze* getan, die wir in Abschnitt 8.3 betrachten. Die grundlegende Idee dieser Netze, die auch *graphische Modelle* genannt werden, ist es, die Pfade, auf denen (probabilistisches) Wissen übertragen wird, in Form eines (gerichteten oder ungerichteten) Graphen darzustellen. Eine solche Darstellung kann gewöhnlich erreicht werden, wenn es gelingt, das Wissen über den betrachteten Anwendungsbereich geeignet zu zerlegen – ähnlich wie man es in der Logik in eine Menge von Implikationen zerlegt. Der Graph beschreibt dann, wie in der Logik, diese Zerlegung und gibt an, auf welchen Wegen Informationen weitergegeben werden müssen. Der Unterschied besteht im wesentlichen darin, dass nicht eine Menge von Implikationen, sondern eine Familie von (bedingten oder Rand-) Verteilungen dargestellt wird, die i.a. informationsreicher sind als einfache Implikationen.

Wenden wir uns nun dem vagen Wissen zu, also etwa vagen Implikationen oder (Handlungs-) Regeln. Hier stoßen wir auf andere Probleme als bei unsicherem Wissen. Schon wegen ihrer Vagheit, aber auch wegen der Anwendungsbereiche, in denen sie eingesetzt werden, können vage Regeln meist als sicher angenommen werden. Da sie jedoch vage formuliert sind, steht bei gegebenen Beobachtungen oft in Zweifel, ob sie angewandt werden können, nämlich dann, wenn die beobachteten Werte in den Penumbrae der im Antezedens auftretenden vagen Begriffe liegen. Dies ist insbesondere dann problematisch, wenn es mehrere Regeln gibt, die angewendet werden können, von denen jedoch keine mit Sicherheit anwendbar ist. In diesem Fall geht es dann nicht darum, durch den Schlussfolgerungsmechanismus die Sicherheit der Konklusion zu bestimmen, sondern darum, zwischen den vagen Regeln geeignet zu interpolieren, um eine passende Konklusion zu bestimmen. Wir können auch sagen, die vagen Regeln gäben nicht,

wie in der Logik, Übertragungswege an, sondern eine unscharfe Beschreibung eines Geländes, in dem wir einen Weg erst auffinden müssen.

Z.B. könnte ein Mensch die Regelung eines technischen Prozesses so beschreiben: Wenn die Temperatur mittel und der Druck hoch ist, darf das Ventil nur wenig geöffnet werden. Wenn aber sowohl Druck als auch Temperatur niedrig sind, muss das Ventil weit geöffnet werden. Was ist dann zu tun, wenn der Druck mittel und die Temperatur niedrig ist? Wie weit ist das Ventil zu öffnen? Offenbar muss hier zwischen den beiden Regeln gemittelt werden, wobei die zweite wohl stärkeres Gewicht erhalten sollte. Eine Methode, um mit solchen Situationen umzugehen, stellen die Fuzzy-Regelsysteme mit ihren speziellen Schlussfolgerungsverfahren bereit, die wir in Abschnitt 8.4 behandeln.

8.1.4 Wahrscheinlichkeit

Der bekannteste Kalkül zur Behandlung von Unsicherheit ist die Wahrscheinlichkeitstheorie. Sie beschäftigt sich mit *zufälligen Ereignissen*. Bei diesen ist zwar bekannt, welche Ereignisse überhaupt eintreten können, nicht jedoch, welches der möglichen Ereignisse tatsächlich eintreten wird (das Eintreten der Ereignisse ist also unsicher). Beispiele sind das Werfen einer Münze oder eines Würfels. In der Wahrscheinlichkeitstheorie wird einem zufälligen Ereignis ein Zahlenwert – genannt Wahrscheinlichkeit – zugeschrieben. Diese Größe soll die Chance oder die Tendenz des Eintretens des Ereignisses beschreiben.

Um eine intuitive Vorstellung zu bekommen, betrachten wir zunächst die klassische Definition der Wahrscheinlichkeit. Die klassische Wahrscheinlichkeitsdefinition ist eigentlich eine Rechnungsvorschrift für Wahrscheinlichkeiten und ging aus Überlegungen über die möglichen Ausgänge in Glücksspielen hervor.

Definition 8.1.1. (Klassische Definition der Wahrscheinlichkeit)

Als Wahrscheinlichkeit $P(A)$ eines Ereignisses A bezeichnet man das Verhältnis der Anzahl der günstigen Ereignisse n_A zur Gesamtzahl der unvereinbaren, gleichmöglichen Ereignisse n , d.h. $P(A) = \frac{n_A}{n}$.

Dass die Voraussetzung der Gleichmöglichkeit erfüllt ist, wird gewöhnlich durch Symmetrievergleichungen hergeleitet. Die unvereinbaren, gleichmöglichen Ereignisse werden *Elementarereignisse* genannt. Sie bilden den *Ereignisraum*. Andere zufällige Ereignisse lassen sich als Kombination von Elementarereignissen darstellen.

Beispiel 8.1.1. Es werde mit einem symmetrischen und aus homogenem Material gefertigten Würfel gewürfelt. Der Ereignisraum besteht aus den Elementarereignissen „Die geworfene Augenzahl ist x “, $x \in \{1, 2, 3, 4, 5, 6\}$. Das Ereignis „Die geworfene Augenzahl ist gerade“ ist aus den Elementarereignissen „Die geworfene Augenzahl ist x “, $x \in \{2, 4, 6\}$ zusammengesetzt und tritt daher in drei der sechs gleichmöglichen Fälle ein. Seine Wahrscheinlichkeit ist folglich $\frac{1}{2}$. \square

Wesentliches Hilfsmittel für die Bestimmung von Wahrscheinlichkeiten auf der Grundlage dieser Definition ist die Kombinatorik, die Methoden zum Auszählen aller und der günstigen Fälle bereitstellt. Eine Verallgemeinerung des klassischen Wahrscheinlichkeitsbegriffs wird dadurch erreicht, dass man die Forderung nach Gleichmöglichkeit der Elementarereignisse aufgibt.

Stattdessen wird ein Ereignisraum definiert und jedem Elementarereignis eine *Elementarwahrscheinlichkeit* zugeordnet. Der klassische Begriff ergibt sich dann als der Spezialfall gleicher Elementarwahrscheinlichkeiten.

Eine andere intuitive Vorstellung der Wahrscheinlichkeit von Ereignissen ist die relative Häufigkeit. Ein Experiment mit zufälligem Ausgang werde unter denselben Bedingungen N -mal durchgeführt. Ist A ein zufälliges Ereignis, so tritt bei jeder Versuchsdurchführung A entweder ein oder nicht ein. Die Zahl der Fälle $h_N(A)$, in denen A eintritt, heißt *absolute Häufigkeit*, der Quotient $r_N(A) = \frac{h_N(A)}{N}$ *relative Häufigkeit*. [40] hat versucht, die Wahrscheinlichkeit $P(A)$ eines Ereignisses als den Grenzwert zu definieren, dem sich die relativen Häufigkeiten für große n beliebig nähern:

$$P(A) = \lim_{N \rightarrow \infty} r_N(A).$$

Doch das gelingt nicht. Es können Versuchsreihen nicht ausgeschlossen werden, in denen

$$\forall N \geq N_0(\epsilon) : P(A) - \epsilon \leq r_N(A) \leq P(A) + \epsilon$$

nicht gilt, wie groß auch immer N_0 gewählt werden mag. (Es ist nicht unmöglich, wenn auch sehr unwahrscheinlich, dass beim Werfen eines Würfels nur Einsen auftreten.) Dennoch ist eine Vorstellung der Wahrscheinlichkeit als eine relative Häufigkeit oft hilfreich, insbesondere, wenn man sie als unsere Schätzung der relativen Häufigkeit eines Ereignisses (bei zukünftigen Durchführungen des entsprechenden Versuchs) interpretiert.

Beispiel 8.1.2. *Es werden etwa gleich viele Mädchen wie Jungen geboren. Die relative Häufigkeit einer Mädchengeburt ist folglich etwa gleich der relativen Häufigkeit einer Jungengeburt. Wir sagen daher, dass die Wahrscheinlichkeit, dass ein Mädchen geboren wird (genauso wie die Wahrscheinlichkeit, dass ein Junge geboren wird) $\frac{1}{2}$ beträgt. Man beachte, dass sich hier der Wert $\frac{1}{2}$ für die Wahrscheinlichkeit nicht – wie etwa bei einem Münzwurf – aus Symmetrieverlegungen ableiten lässt.* \square

Der klassische Wahrscheinlichkeitsbegriff und die Interpretation als relative Häufigkeit sind stark in unserer Intuition verwurzelt. Die moderne Mathematik hat sich jedoch die axiomatische Methode zu eigen gemacht, bei der von der Bedeutung der Objekte, über die man spricht, abstrahiert wird. Sie nimmt Objekte als gegeben an, die keine anderen Eigenschaften haben als ihre Identität, und untersucht lediglich die Struktur der Relationen zwischen diesen Objekten, die sich aus vorausgesetzten Axiomen ergibt. Auch die Wahrscheinlichkeitstheorie wird daher heute axiomatisch aufgebaut, und zwar über die Kolmogorow-Axiome. Unter einem Ereignis wird in diesen Axiomen einfach eine Menge von Elementarereignissen verstanden, die unterscheidbar sind, d.h. eine Identität haben (s.o.). Eine Wahrscheinlichkeit ist dann eine Zahl, die diesen Ereignissen zugeordnet wird. Zunächst definieren wir jedoch die Begriffe Ereignisalgebra und σ -Algebra.

Definition 8.1.2 (Ereignisalgebra und σ -Algebra). *Sei \mathcal{S} ein System von Ereignissen (Mengen) über einer Grundmenge Ω von Elementarereignissen (dem Ereignisraum). \mathcal{S} heißt Ereignisalgebra genau dann, wenn gilt:*

1. *\mathcal{S} enthält das sichere Ereignis Ω (d.h., die Vereinigung aller Elementarereignisse) und das unmögliche Ereignis \emptyset .*

2. Gehört A zu \mathcal{S} , so gehört auch das Ereignis \overline{A} zu S .
3. Gehören A und B zu \mathcal{S} , so gehören auch die Ereignisse $A \cap B$ und $A \cup B$ zu \mathcal{S} .

Es kann außerdem die folgende Bedingung erfüllt sein:

- 3'. Gehört für alle $n \in \mathbb{N}$ das Ereignis A_n zu \mathcal{S} , dann gehören auch die Ereignisse $\bigcup_{n=1}^{\infty} A_n$ und $\bigcap_{n=1}^{\infty} A_n$ zu \mathcal{S} .

In diesem Fall nennt man \mathcal{S} eine σ -Algebra.

Die Wahrscheinlichkeit wird nun über die folgenden Axiome definiert:

Definition 8.1.3 (Kolmogorow-Axiome). Gegeben sei eine Ereignisalgebra \mathcal{S} über einem endlichen Ereignisraum Ω .

1. Die Wahrscheinlichkeit $P(A)$ eines Ereignisses $A \in \mathcal{S}$ ist eine eindeutig bestimmte, nicht-negative Zahl, die höchstens gleich Eins sein kann, d.h., es gilt $0 \leq P(A) \leq 1$.
2. Das sichere Ereignis Ω besitzt die Wahrscheinlichkeit Eins, d.h. $P(\Omega) = 1$.
3. (Additionsaxiom) Für zwei unvereinbare (unverträgliche) Ereignisse A und B (also mit $A \cap B = \emptyset$) gilt $P(A \cup B) = P(A) + P(B)$.

Bei Ereignisräumen Ω , die unendlich viele Elemente enthalten, muss \mathcal{S} eine σ -Algebra sein und Axiom 3 ersetzt werden durch:

- 3'. (erweitertes Additionsaxiom) Sind A_1, A_2, \dots abzählbar unendlich viele, paarweise unvereinbare Ereignisse (also mit $\forall i, j : i \neq j \Rightarrow A_i \cap A_j = \emptyset$), so gilt

$$P\left(\bigcup_{n=1}^{\infty} A_n\right) = \sum_{n=1}^{\infty} A_n.$$

Die Wahrscheinlichkeit $P(A)$ lässt sich also als eine auf einer Ereignisalgebra definierte Funktion mit bestimmten Eigenschaften sehen. Es ergeben sich die Folgerungen:

1. Für jedes Ereignis A gilt $P(\overline{A}) = 1 - P(A)$.
2. Das unmögliche Ereignis besitzt die Wahrscheinlichkeit Null, d.h. $P(\emptyset) = 0$.
3. Aus $A \subseteq B$ folgt $P(A) \leq P(B)$.
4. Für beliebige Ereignisse A und B gilt $P(A \setminus B) = P(A \cap \overline{B}) = P(A) - P(A \cap B)$.
5. Für beliebige Ereignisse A und B gilt $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.
6. Da $P(A \cap B)$ nicht-negativ ist, folgt unmittelbar $P(A \cup B) \leq P(A) + P(B)$.
7. Durch einfache Induktion erhält man aus dem Additionsaxiom: Sind A_1, \dots, A_n paarweise unvereinbar, so gilt $P(A_1 \cup \dots \cup A_n) = \sum_{i=1}^n P(A_i)$.

Das Kolmogorowsche Axiomensystem ist widerspruchsfrei, da es Systeme gibt, die allen diesen Axiomen genügen. Die Axiomatik von Kolmogorow gestattet es, die Wahrscheinlichkeitstheorie als Teil der Maßtheorie aufzubauen und die Wahrscheinlichkeit als nichtnegative normierte additive Mengenfunktion, d.h. als Maß zu interpretieren.

Definition 8.1.4 (Wahrscheinlichkeitsraum). *Sei Ω ein Ereignisraum, \mathcal{S} eine σ -Algebra über Ω und P eine Wahrscheinlichkeit, die auf \mathcal{S} definiert ist. Dann heißt das Tripel (Ω, \mathcal{S}, P) Wahrscheinlichkeitsraum.*

Die Anwendung der Wahrscheinlichkeitstheorie erfordert – wie die Anwendung jeder abstrakten mathematischen Theorie – die Deutung der in ihr auftretenden Begriffe, d.h. ihre Abbildung auf die Welt. Obwohl wir bereits intuitive Deutungen betrachtet haben, wenden wir uns noch einmal kurz den auftretenden Problemen zu. Man unterscheidet i.a. drei Deutungen des Wahrscheinlichkeitsbegriffs [47, 56]:

- Die *logische Deutung* fasst die Wahrscheinlichkeit einer Aussage oder eines Zustandes als Funktion eines möglichen Wissens auf. Gegeben ein Wissen (als wahr bekannte Aussagen) und eine Aussage, so bestimmen die logischen Regeln der Sprache die logische Wahrscheinlichkeit dieser Aussage bezüglich dieses Wissens.
- Die *empirische (frequentistische) Deutung* fasst Wahrscheinlichkeitsaussagen als Darstellung statistischer Wahrheiten über die Welt (im Sinne physikalischer Parameter) auf, die sich in den (relativen) Häufigkeiten des Auftretens von Ereignissen niederschlagen, aus denen sie dann abgelesen werden können.
- Die *subjektive (personalistische) Deutung* fasst die Wahrscheinlichkeit als Ausdruck einer Regel auf, der eine Person in einer gegebenen Situation „in ihrer nützlichen Gewohnheit“ folgt. Ein Zahlenwert kann ihr in dieser Deutung über die Interpretation als Wettbereitschaft zugeordnet werden.

Leider sind mit allen drei Deutungen Probleme verbunden [56]:

- Die Werte der (bedingten) Wahrscheinlichkeiten sind nicht logisch begründbar.
- Die A-priori-Wahrscheinlichkeiten (speziell in der Bayes-Statistik) sind willkürlich.
- Der Erfolg oder Mißerfolg eines Wettsystems ist ein historisches Faktum, insofern also nicht subjektiv.

Den Erfolg der Wahrscheinlichkeitstheorie kann keine der drei Deutungen vollständig erklären. Die Lösung dieses (philosophischen) Problems steht noch aus.

Oft ist die Wahrscheinlichkeit eines Ereignisses A zu bestimmen, wenn bereits bekannt ist, dass ein Ereignis B eingetreten ist. Solche Wahrscheinlichkeiten nennt man *bedingte* Wahrscheinlichkeiten und bezeichnet sie mit $P(A | B)$. Streng genommen sind die „unbedingten“ Wahrscheinlichkeiten, die wir bisher betrachtet haben, auch bedingt, da wir sie stets unter bestimmten Bedingungen angegeben haben. So haben wir in Beispiel 8.1.1 etwa angenommen, dass der Würfel, mit dem wir werfen, symmetrisch und aus homogenem Material sei. Nur unter diesen, und gegebenfalls weiteren stillschweigend vorausgesetzten Bedingungen (z.B. keine elektromagnetische Beeinflussung des Würfels etc.) haben wir die Wahrscheinlichkeit jeder Augenzahl zu $\frac{1}{6}$ bestimmt.

Definition 8.1.5 (bedingte Wahrscheinlichkeit). *Seien A und B zwei beliebige Ereignisse mit $P(B) > 0$. Dann heißt*

$$P(A | B) = \frac{P(A \cap B)}{P(B)}$$

die bedingte Wahrscheinlichkeit von A unter (der Bedingung) B .

Beispiel 8.1.3. Es werde mit zwei Würfeln geworfen. Wie groß ist die Wahrscheinlichkeit, dass einer der Würfel eine fünf zeigt, wenn bekannt ist, dass die Summe der Augenzahlen acht ist? Beim Wurf mit zwei Würfeln gibt es offenbar 36 Elementarereignisse, von denen fünf die Augensumme acht haben ($4+4$, $5+3$ und $6+2$, wobei die beiden letzten wegen der zwei möglichen Verteilungen der beiden Zahlen auf die beiden Würfel doppelt gezählt werden müssen), d.h., es ist $P(B) = \frac{5}{36}$. Das Ereignis „Augensumme gleich acht und ein Würfel zeigt eine fünf“ kann durch genau zwei Elementarereignisse erreicht werden: Entweder der erste Würfel zeigt eine fünf und der zweite eine drei oder umgekehrt. Folglich ist $P(A \cap B) = \frac{2}{36}$ und damit die gesuchte bedingte Wahrscheinlichkeit $P(A | B) = \frac{2}{5}$. \square

Eine bedingte Wahrscheinlichkeit hat alle Eigenschaften einer Wahrscheinlichkeit, d.h., sie erfüllt die Kolmogorow-Axiome. Damit gilt:

Satz 8.1.1. Für ein fest gewähltes Ereignis B mit $P(B) > 0$ ist die durch $P_B(A) = P(A|B)$ erklärte Funktion P_B eine Wahrscheinlichkeit, die $P_B(\overline{B}) = 0$ erfüllt.

Mit Hilfe der bedingten Wahrscheinlichkeit wird die (stochastische) Unabhängigkeit definiert. Dieser Begriff lässt sich so motivieren: Wenn z.B. Rauchen keinen Einfluß auf das Entstehen von Lungenkrebs hat, so müßte in der Gruppe der Raucher und der Nichtraucher der Anteil (die relative Häufigkeit) der Personen mit Lungenkrebs etwa gleich sein.

Definition 8.1.6 ((stochastisch) unabhängig). Sei B ein Ereignis mit $0 < P(B) < 1$. Dann heißt das Ereignis A (stochastisch) unabhängig von B , wenn $P(A | B) = P(A | \overline{B})$.

Zu dieser Beziehung sind die folgenden, meist angenehmeren Beziehungen äquivalent:

Satz 8.1.2. Ein Ereignis A ist von einem Ereignis B mit $0 < P(B) < 1$ genau dann (stochastisch) unabhängig, wenn $P(A | B) = P(A)$ oder wenn $P(A \cap B) = P(A) \cdot P(B)$.

Man beachte, dass die Beziehung der (stochastischen) Unabhängigkeit symmetrisch ist: Ist A (stochastisch) unabhängig von B , so ist auch B (stochastisch) unabhängig von A .

Im folgenden geben wir einige wichtige Sätze der Wahrscheinlichkeitstheorie an, die wir im weiteren Verlauf dieses Kapitels brauchen werden.

Satz 8.1.3 (Produktsatz/Multiplikationssatz). Für beliebige Ereignisse A und B gilt $P(A \cap B) = P(A | B) \cdot P(B)$.

Dieser Satz folgt unmittelbar aus der Definition der bedingten Wahrscheinlichkeit unter Hinzunahme der offensichtlichen Beziehung $P(A \cap B) = 0$, wenn $P(B) = 0$. Durch einfache Induktion über die Zahl der Ereignisse erhält man die Verallgemeinerung für n Ereignisse:

$$P\left(\bigcap_{i=1}^n A_i\right) = \prod_{i=1}^n P\left(A_i \mid \bigcap_{k=1}^{i-1} A_k\right).$$

Oft hat man es mit Situationen zu tun, in denen die Wahrscheinlichkeit von disjunkten, zusammen den ganzen Ereignisraum abdeckenden Ereignissen A_i , sowie die bedingten Wahrscheinlichkeiten eines weiteren Ereignisses B unter den A_i bekannt sind. Gesucht ist die Gesamtwahrscheinlichkeit für das Ereignis B . Als Beispiel betrachte man etwa eine Fabrik, in der es eine

bestimmte Anzahl von Maschinen zur Herstellung des gleichen Produktes gibt. Der Ausstoß der Maschinen sowie ihre jeweilige Ausschussrate seien bekannt, die Gesamtausschussrate soll berechnet werden. Eine solche Berechnung ermöglicht der Satz über die vollständige Wahrscheinlichkeit. Vorher definieren wir aber noch den Begriff der vollständigen Ereignisdisjunktion.

Definition 8.1.7 (vollständige Ereignisdisjunktion). *n Ereignisse A_1, \dots, A_n bilden eine vollständige Ereignisdisjunktion, wenn alle Paare A_i, A_k , $i \neq k$ unvereinbar sind ($A_i \cap A_k = \emptyset$ für $i \neq k$) und wenn gilt $A_1 \cup \dots \cup A_n = \Omega$.*

Satz 8.1.4 (Satz über die vollständige Wahrscheinlichkeit). *Sei A_1, \dots, A_n eine vollständige Ereignisdisjunktion mit $\forall 1 \leq i \leq n : P(A_i) > 0$ (und, wie aus den Kolmogorow Axiomen folgt, $\sum_{i=1}^n P(A_i) = 1$). Dann gilt für die Wahrscheinlichkeit eines beliebigen Ereignisses B*

$$P(B) = \sum_{i=1}^n P(B | A_i) \cdot P(A_i).$$

Den Satz über die vollständige Wahrscheinlichkeit erhält man, indem man den Produktsatz auf $P(B) = P(B \cap \Omega) = P(B \cap \bigcup_{i=1}^n A_i) = P(\bigcup_{i=1}^n (B \cap A_i)) = \sum_{i=1}^n P(B \cap A_i)$ anwendet.

Mit Hilfe des Satzes über die vollständige Wahrscheinlichkeit lässt sich nun leicht der wichtige Bayessche Satz ableiten (nach Thomas Bayes, 1702–1761, obwohl die heute übliche Form erst von Pierre-Simon Laplace, 1749–1827, angegeben wurde). Dazu muss man sich nur klar machen, dass sich der Produktsatz für Wahrscheinlichkeiten auf das gleichzeitige Eintreten zweier Ereignisse A und B in zweierlei Weise anwenden lässt:

$$P(A \cap B) = P(A | B) \cdot P(B) = P(B | A) \cdot P(A).$$

Dies führt nach Division durch die Wahrscheinlichkeit $P(B)$, die dazu als positiv vorausgesetzt werden muss, zum ersten Teil des Bayesschen Satzes. Die Anwendung des Satzes über die vollständige Wahrscheinlichkeit auf den Nenner liefert dann den zweiten Teil.

Satz 8.1.5 (Bayesscher Satz/Bayessche Formel). *Für eine vollständige Ereignisdisjunktion A_1, \dots, A_n mit $\forall 1 \leq i \leq n : P(A_i) > 0$ und jedes Ereignis B mit $P(B) > 0$ gilt*

$$P(A_k | B) = \frac{P(B | A_k) \cdot P(A_k)}{P(B)} = \frac{P(B | A_k) \cdot P(A_k)}{\sum_{i=1}^n P(B | A_i) \cdot P(A_i)}.$$

Diese Formel heißt auch die Formel über die Wahrscheinlichkeit von Hypothesen.

Beispiel 8.1.4. Gegeben seien fünf Urnen folgenden Inhalts:

- zwei Urnen vom Inhalt A_1 mit je zwei weißen und drei schwarzen Kugeln,
- zwei Urnen vom Inhalt A_2 mit je einer weißen und vier schwarzen Kugeln,
- eine Urne mit dem Inhalt A_3 mit vier weißen und einer schwarzen Kugel.

Aus einer willkürlich gewählten Urne werde eine Kugel entnommen. Sie sei weiß. (Dies sei das Ereignis B.) Wie groß ist die (A-posteriori-)Wahrscheinlichkeit dafür, dass die Kugel aus der

Urne mit Inhalt A₃ stammt? Nach Voraussetzung ist:

$$\begin{aligned} P(A_1) &= \frac{2}{5}, & P(A_2) &= \frac{2}{5}, & P(A_3) &= \frac{1}{5}, \\ P(B | A_1) &= \frac{2}{5}, & P(B | A_2) &= \frac{1}{5}, & P(B | A_3) &= \frac{4}{5}. \end{aligned}$$

Mit der Bayesschen Formel erhalten wir:

$$P(A_3 | B) = \frac{\frac{4}{5} \cdot \frac{1}{5}}{\frac{2}{5} \cdot \frac{2}{5} + \frac{1}{5} \cdot \frac{2}{5} + \frac{4}{5} \cdot \frac{1}{5}} = \frac{2}{5}$$

Genauso finden wir $P(A_1 | B) = \frac{2}{5}$ und $P(A_2 | B) = \frac{1}{5}$. □

8.1.5 Fuzzy-Menge

Wie schon bemerkt, ist die Fuzzy-Mengentheorie bzw. Fuzzy-Logik [59] der vorherrschende Ansatz zur Behandlung von Vagheit. Mit diesem Ansatz versucht man einige der Probleme, die sich aus der Vagheit sprachlicher Ausdrücke ergeben, dadurch zu lösen, dass man die Begriffe der Zugehörigkeit zu einer Menge bzw. des Wahrheitswertes „fuzzifiziert“, d.h. Grade der Zugehörigkeit bzw. Wahrheit oder Möglichkeit einführt.

Die Probleme, die sich durch die Vagheit sprachlicher Begriffe ergeben, lassen sich am besten mit Hilfe des klassischen *Sorites-Paradoxons* deutlich machen. Als Sorites-Paradoxon bezeichnete man ursprünglich die Frage „Wieviel Körner machen einen Haufen?“.² Dieses Paradoxon soll von Eubulides von Milet (4. Jahrhundert v. Chr.) aufgebracht worden sein. Wir betrachten die folgende Form des Sorites-Paradoxons:

1. Wenn man von einem Sandhaufen ein Sandkorn wegnimmt,
dann bleibt ein Sandhaufen übrig.
2. Eine Ansammlung von einer Milliarde Sandkörnern ist ein Sandhaufen.

Aus diesen beiden, offenbar wahren Aussagen folgt:

3. Eine Ansammlung von 999.999.999 Sandkörnern ist ein Sandhaufen.

Diese Schlussfolgerung können wir aber zu einer Prämisse machen und erschließen so:

4. Eine Ansammlung von 999.999.998 Sandkörnern ist ein Sandhaufen.

Indem wir die Schlusskette fortsetzen, erhalten wir schließlich, dass eine Ansammlung von drei, zwei, einem, ja sogar gar keinem Sandkorn ein Sandhaufen ist. Diese Schlussfolgerung ist jedoch offenbar falsch, denn sonst lägen ja überall Sandhaufen herum.

² Der Name dieses Paradoxons leitet sich ab von griech. *σωρος* (soros), was „Haufen“ bedeutet. Das Paradoxon hieß ursprünglich einfach *σωρίτης* (sorites), d.h. „Häufelnder“. Manchmal ergänzte man noch *λόγος* (logos) oder *συλλογισμός* (syllogismos) was dann „Haufenschluss“ ergibt.

Das Problem besteht darin, dass der Ausgangspunkt, nämlich dass eine Milliarde Sandkörner ein Sandhaufen sind, sicherlich richtig ist, wir aber in der Schlusskette kaum einen *einzelnen* Schluss identifizieren können, der zu einer falschen Schlussfolgerung führt, und der erklärte, warum wir schließlich ein falsches Ergebnis erhalten. Es gibt keine scharf bestimmte Anzahl von Sandkörnern, die noch einen Sandhaufen bilden, während eine Ansammlung von Sandkörnern mit einem Sandkorn weniger kein Sandhaufen mehr ist. Man wird vielmehr sagen müssen, dass die Konklusionen „allmählich“ falsch werden, und zwar während wir uns schließend durch die Penumbra des Begriffs „Sandhaufen“ bewegen.

Es liegt nahe, dieses „allmähliche“ Falschwerden der Konklusionen dadurch zu modellieren, dass man neben den Wahrheitswerten *wahr* und *falsch* Zwischenwerte, also *Grade der Wahrheit*, einführt. In der Penumbra des Begriffs „Sandhaufen“ wird der Wahrheitsgrad dann mit jedem Schluss, d.h. mit jedem Sandkorn, das entfernt wird, ein bißchen kleiner, bis er schließlich den Wert *falsch* erreicht. Genauso können wir mit dem Begriff der Zugehörigkeit zu einer Menge verfahren, der in der klassischen Mengenlehre auch nur zweiwertig ist: Entweder ein Element gehört zu einer Menge oder es gehört nicht dazu. Da sich jedoch die Ansammlungen von Sandkörnern, die unter den Begriff des Sandhaufens fallen, nicht scharf von jenen trennen lassen, die nicht unter ihn fallen, erscheint es sinnvoll, für Ansammlungen von Sandkörnern, die in der Penumbra des Begriffs „Sandhaufen“ liegen, *Grade der Zugehörigkeit* einzuführen. Diese Zugehörigkeitsgrade sollten um so höher sein, je mehr Sandkörner die Ansammlung enthält.

Formal lässt sich der Begriff einer Fuzzy-Menge über die sogenannte *Indikatorfunktion* einer Menge einführen. Diese Funktion ordnet den Elementen einer gegebenen Grundmenge, die in der zu beschreibenden Menge enthalten sind, eine 1, allen anderen eine 0 zu und zeigt so an (lat. *indicare*: anzeigen), welche Elemente enthalten sind.

Definition 8.1.8 (Indikatorfunktion). *Sei U eine Menge und $X \subseteq U$. Die Indikatorfunktion von X bzgl. U ist die Funktion*

$$I_X : U \rightarrow \{0, 1\}, \quad u \mapsto \begin{cases} 1, & \text{falls } u \in X, \\ 0, & \text{sonst.} \end{cases}$$

Indem man außer 0 und 1 auch Zwischenwerte zulässt, erhält man eine *Fuzzy-Menge*.

Definition 8.1.9 (Fuzzy-Menge). *Sei U eine Menge. Eine Fuzzy-Menge über U ist eine Funktion $\mu : U \rightarrow [0, 1]$.*

Wie bei der Wahrscheinlichkeit stellt sich natürlich auch hier das Problem, wie denn die formale Theorie zu interpretieren ist, insbesondere welche Bedeutung ein Zugehörigkeitsgrad zwischen 0 und 1, z.B. 0.7, hat. Zwar haben wir aus den obigen Erklärungen eine Intuition, was der Zugehörigkeitsgrad ausdrücken soll, doch wissen wir noch nicht, wie man die Zahlenwerte der Zugehörigkeitsgrade festlegen und begründen kann.

In der Tat ist diese Frage nach der Semantik der Zugehörigkeitsgrade ein fundamentales Problem der Fuzzy-Mengentheorie bzw. Fuzzy-Logik, die in den meisten Büchern zu diesen Themen nicht hinreichend beantwortet wird. Zwar gibt es z.B. mit der *Possibilitätstheorie* [13] und ihren verschiedenen Interpretationen Ansätze, die angedeuteten Fragen zu beantworten, doch würde eine genaue Erörterung den Rahmen dieses Beitrags sprengen. Wir verfolgen daher hier einen anderen, für die meisten Anwendungen hinreichenden, oft sogar besser geeigneten Ansatz, der

auf der folgenden Einsicht beruht: Das Problem, die genauen Werte der Zugehörigkeitsgrade zu rechtfertigen, stellt sich eigentlich nur dann, wenn man sich darauf versteift, Fuzzy-Mengen als die *Bedeutung* oder die (mathematische) *Interpretation* eines sprachlichen Ausdrucks aufzufassen. Doch das ist oft gar nicht notwendig. Vielmehr haben wir oft folgende Situation vorliegen, die bereits im Abschnitt über Vagheit behandelt wurde (siehe Abschnitt 8.1.2): Ein (menschlicher) Experte gibt (z.B. für die Steuerung eines Prozesses) unscharfe Regeln an. Da die Regeln unscharf formuliert sind, gibt es zwar Bereiche, in denen sie sicher anwendbar sind, und solche, in denen sie sicher nicht gelten, aber auch Zwischenbereiche, in denen nicht eindeutig entschieden werden kann, welche von ggf. mehreren Regeln verwendet werden sollte. In diesen Zwischenbereichen sollte man offenbar zwischen den (nur eingeschränkt) anwendbaren Regeln interpolieren, u.U. gewichtet mit der „Nähe“ zu dem Bereich, in dem die jeweilige Regel sicher gültig ist. Mit anderen Worten: wir beschränken uns hier darauf, Fuzzy-Mengen als einen Mechanismus zur Glättung von vage beschriebenen Funktionen und zur Interpolation zwischen vagen Regeln aufzufassen.

Trotz dieser Beschränkung führen wir im folgenden – im wesentlichen der Vollständigkeit halber – die Standardoperationen auf Fuzzy-Mengen ein, auch wenn sie nicht benötigt werden, wenn man Fuzzy-Mengen nur zur Interpolation zwischen vagen Regeln verwendet, wie wir es in Abschnitt 8.4 tun.

Die Operationen auf Fuzzy-Mengen werden in Analogie zu den Mengenoperationen Schnitt, Vereinigung und Komplementbildung eingeführt. Bedingung ist stets, dass die Operationen auf Fuzzy-Mengen mit den entsprechenden Operationen auf (scharfen) Mengen übereinstimmen, wenn man sich auf Indikatorfunktionen beschränkt. Weiter wird festgelegt, dass die Operationen elementweise auszuführen sind. D.h., der Zugehörigkeitsgrad eines Elementes z.B. zum Schnitt zweier Fuzzy-Mengen muss sich aus den Zugehörigkeitsgraden des Elementes zu den beiden Mengen berechnen lassen.

Diese Bedingungen, aber auch weitere naheliegende, wie z.B. Assoziativität und Kommutativität, legen die Operationen jedoch noch nicht eindeutig fest. Daher betrachtet man Klassen von Operationen, und zwar sogenannte *t-Normen* (triangular norms) als Grundlage einer verallgemeinerten Schnittmengenbildung, sogenannte *t-Conormen* als Grundlage einer verallgemeinerten Vereinigung, und sogenannte *Negationen* als Grundlage einer verallgemeinerten Komplementbildung.

Definition 8.1.10 (*t*-Norm). Eine Funktion $\top : [0, 1]^2 \rightarrow [0, 1]$ heißt *t-Norm*, wenn für alle $a, b, c \in [0, 1]$ gilt:

- (1) $\top(a, \top(b, c)) = \top(\top(a, b)c)$ (Assoziativität)
- (2) $\top(a, b) = \top(b, a)$ (Kommutativität)
- (3) $\top(a, 1) = a$ (neutrales Element)
- (4) $a \leq b \Rightarrow \top(a, c) \leq \top(b, c)$ (Monotonie)

Die wohl bekanntesten *t*-Normen sind $\top_{\min}(a, b) = \min\{a, b\}$, $\top_{\text{prod}}(a, b) = a \cdot b$ und $\top_{\text{Łuka}}(a, b) = \max\{0, a + b - 1\}$ (nach Jan Łukasiewicz, 1878–1956) [27]. Abbildung 8.1 zeigt ihre Funktionsgraphen. Aus jeder *t*-Norm erhält man eine Schnittmengenbildung für Fuzzy-Mengen, indem man sie elementweise anwendet. D.h., für zwei Fuzzy-Mengen μ_1 und μ_2 wird der Schnitt definiert als

$$\forall u \in U : (\mu_1 \cap \mu_2)(u) = \top(\mu_1(u), \mu_2(u)).$$

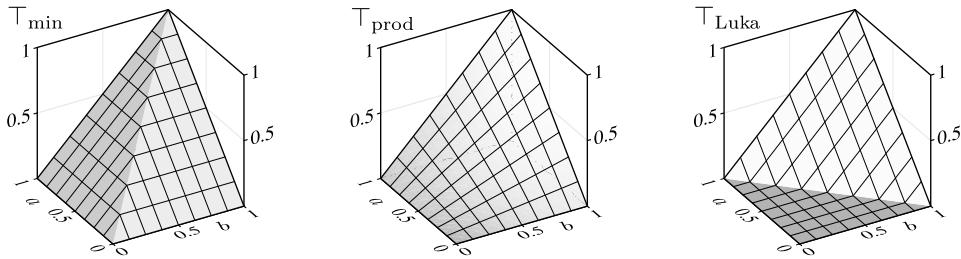


Abbildung 8.1: Die t -Normen T_{\min} , T_{prod} und T_{Luka} .

Definition 8.1.11 (t -Conorm). Eine Funktion $\perp : [0, 1]^2 \rightarrow [0, 1]$ heißt t -Conorm, wenn für alle $a, b, c \in [0, 1]$ gilt:

- (1) $\perp(a, \perp(b, c)) = \perp(\perp(a, b), c)$ (Assoziativität)
- (2) $\perp(a, b) = \perp(b, a)$ (Kommutativität)
- (3) $\perp(a, 0) = a$ (neutrales Element)
- (4) $a \leq b \Rightarrow \perp(a, c) \leq \perp(b, c)$ (Monotonie)

Die bekanntesten t -Conormen sind $\perp_{\min}(a, b) = \max\{a, b\}$, $\perp_{\text{prod}}(a, b) = a + b - ab$ und $\perp_{\text{Luka}}(a, b) = \min\{a + b, 1\}$ [27]. Abbildung 8.2 zeigt ihre Funktionsgraphen. Aus jeder t -Conorm erhält man, analog zu den t -Normen, eine Vereinigungsoperation für Fuzzy-Mengen, indem man sie elementweise anwendet. D.h., für zwei Fuzzy-Mengen μ_1 und μ_2 wird die Vereinigung definiert als

$$\forall u \in U : (\mu_1 \cup \mu_2)(u) = \perp(\mu_1(u), \mu_2(u)).$$

Definition 8.1.12 (Negation). Eine Funktion $n : [0, 1] \rightarrow [0, 1]$ heißt Negation, wenn sie die Bedingungen $n(0) = 1$, $n(1) = 0$ und $\forall a, b \in [0, 1] : a \leq b \Rightarrow n(a) \geq n(b)$ erfüllt.

Am bekanntesten ist die von Zadeh vorgeschlagene Negation $n(a) = 1 - a$ [27]. Eine Komplementbildung für Fuzzy-Mengen erhält man aus einer Negation in dem man sie, wie die t -Normen und t -Conormen, elementweise anwendet. D.h., man definiert als Komplement $\bar{\mu}$ einer Fuzzy-Menge μ

$$\forall u \in U : \bar{\mu}(u) = n(\mu(u)).$$

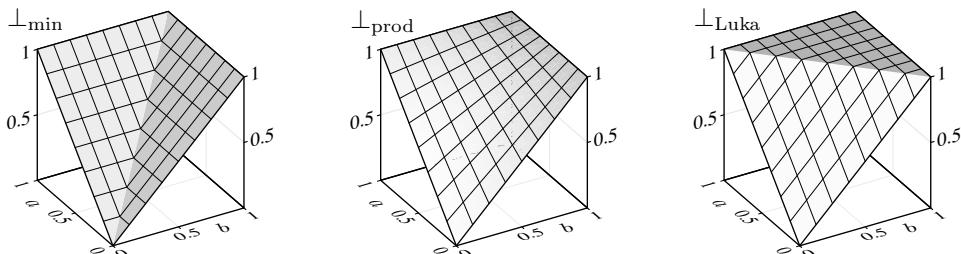


Abbildung 8.2: Die t -Conormen \perp_{\max} , \perp_{sum} und \perp_{Luka} .

Üblicherweise werden eine t -Norm und eine t -Conorm über eine Negation zu einem dualen Paar assoziiert. Mit der Negation $\overline{\mu} = 1 - \mu$ werden z.B. die oben genannten, mit dem gleichen Index versehenen t -Normen und t -Conormen miteinander assoziiert. Diese Assoziation hat den Vorteil, dass dann die De Morganschen Gesetze gelten, d.h., es gilt

$$\overline{\mu_1 \cup \mu_2} = \overline{\mu_1} \cap \overline{\mu_2} \quad \text{und} \quad \overline{\mu_1 \cap \mu_2} = \overline{\mu_1} \cup \overline{\mu_2}.$$

Am häufigsten verwendet man die Negation $n(a) = 1 - a$, die t -Norm \top_{\min} und die zu ihr bezüglich n duale t -Conorm \perp_{\min} , da dieser Operationensatz günstige algebraische Eigenschaften hat. Z.B. sind Schnitt und Vereinigung idempotent, d.h., es ist $\mu \cap \mu = \mu$ und $\mu \cup \mu = \mu$, und es gelten die Distributivgesetze, d.h., es gilt

$$\begin{aligned} \mu_1 \cap (\mu_2 \cup \mu_3) &= (\mu_1 \cap \mu_2) \cup (\mu_1 \cap \mu_3) \quad \text{und} \\ \mu_1 \cup (\mu_2 \cap \mu_3) &= (\mu_1 \cup \mu_2) \cap (\mu_1 \cup \mu_3). \end{aligned}$$

Distributivität ist jedoch in manchen Anwendungen keine wünschenswerte Eigenschaft, so dass mitunter auch andere Operationensätze verwendet werden [26, 27].

8.2 Sicherheitsfaktoren

Zur Behandlung von Unsicherheit in medizinischen Expertensystemen wurde in [51] der Sicherheitsfaktoransatz vorgeschlagen und in dem System MYCIN implementiert [5, 52]. MYCIN ist ein Expertensystem zur Beratung von Ärzten bei der Behandlung bakteriogener Infektionskrankheiten. In einem Beratungsdialog erfragt MYCIN Angaben über den Patienten, entwickelt daraus eine Diagnose und macht Therapievorschläge.

8.2.1 Grundlagen des Sicherheitsfaktoransatzes

MYCIN ist ein regelbasiertes System, d.h., das Wissen über die Zusammenhänge zwischen möglichen Beobachtungen und Hypothesen wird in Form von Regeln (Implikationen) dargestellt. Gewöhnlich bilden Symptome oder allgemein Beobachtungen das Antezedens, eine Diagnose das Konsequens einer Regel. Geschlossen wird mit diesen Regeln wie in der Logik: Ist das Antezedens einer Regel wahr, kann man ihr Konsequens erschließen. Aber im Gegensatz zur klassischen Logik sind auch solche Regeln zugelassen, die nicht notwendig wahr sind, also auch nicht völlig verlässliche Schlüsse zulassen.

Beispiel 8.2.1 (MYCIN Regel).

Falls S_1 : Die Färbung des Organismus ist grampositiv
 und S_2 : Das Erscheinungsbild des Organismus ist kokkenförmig
 und S_3 : Die Wachstumsform des Organismus ist kettenförmig
 dann gibt es Anzeichen dafür (0.7),
 dass H : Der Typ des Organismus ist Streptokokke.

Eine solche Regel wird kurz als $S_1 \wedge S_2 \wedge S_3 \xrightarrow{0.7} H$ geschrieben. □

Die obige Regel enthält die Formulierung „gibt es Anzeichen dafür“. Aus dem Antezedens der Regel kann man ihr Konsequens also nicht mit Sicherheit erschließen. Stattdessen wird der

Regel der Zahlenwert 0.7 zugeordnet, der den Sicherheitsgrad der Regel beschreiben soll. Dieser Zahlenwert wird daher als *Sicherheitsfaktor* (certainty factor) bezeichnet.

Es ist offenbar naheliegend, einen Sicherheitsfaktor als bedingte Wahrscheinlichkeit des Konsequens gegeben das Antezedens der Regel aufzufassen, für das obige Beispiel also $P(H | S_1 \cap S_2 \cap S_3) = 0.7$. Dann folgt aber $P(\overline{H} | S_1 \cap S_2 \cap S_3) = 0.3$. Die Übersetzung dieser Beziehung in Regelform, d.h. $S_1 \wedge S_2 \wedge S_3 \xrightarrow{0.3} \overline{H}$, lehnte der Experte, der die obige Regel formulierte hatte, jedoch ab. Seine Begründung: Das Vorliegen von S_1 , S_2 und S_3 sagt nur etwas über die Gültigkeit der Hypothese H , aber keinesfalls etwas über die Negation von H . Für die Entwickler von MYCIN war dies ein Grund, nach einer anderen Interpretation für Sicherheitsfaktoren zu suchen.

Die grundlegende Idee der schließlich gewählten Deutung ist, den Sicherheitsgrad nicht als absolutes Maß für das Vertrauen in eine Hypothese H aufzufassen, wenn das Antezedens der zugehörigen Regel erfüllt ist, sondern als Änderung dieses Vertrauens, wenn die im Antezedens genannten Sachverhalte bekannt werden. Das Vertrauen in eine Hypothese H wird dabei als subjektive Wahrscheinlichkeit $P(H)$ gesehen, so dass dem Sicherheitsfaktor, wenn auch auf andere Weise als oben, eine wahrscheinlichkeitstheoretische Interpretation gegeben werden kann: Sei $P(H)$ die A-priori-Wahrscheinlichkeit der Hypothese H und $P(H|E)$ die A-posteriori-Wahrscheinlichkeit der Hypothese H nach Bekanntwerden der Evidenz E . Dann ist die Änderung des Vertrauensgrades in H die Differenz $P(H|E) - P(H)$ dieser Wahrscheinlichkeiten. Indem man diese Änderung relativ zur maximal möglichen Änderung beschreibt, lässt sie sich durch eine Zahl zwischen -1 und 1 ausdrücken. Wir unterscheiden dazu zwei Fälle:

$P(H|E) > P(H)$ (Das Vertrauen in H nimmt durch E zu.)

Da die Wahrscheinlichkeit $P(H|E)$ höchstens den Wert 1 annimmt, kann sich das Vertrauen in H höchstens um $1 - P(H)$ ändern.

$$P(H|E) - P(H) = c^+ \cdot (1 - P(H)) \quad \text{mit } c^+ \in (0, 1].$$

$P(H|E) < P(H)$ (Das Vertrauen in H nimmt durch E ab.)

Da die Wahrscheinlichkeit $P(H|E)$ mindestens den Wert 0 annimmt, kann sich das Vertrauen in H höchstens um $-P(H)$ ändern.

$$P(H|E) - P(H) = c^- \cdot P(H) \quad \text{mit } c^- \in [-1, 0).$$

Indem man die Faktoren c^+ und c^- auch für die jeweils ausgeschlossenen Fälle sinnvoll definiert, erhält man die folgenden beiden Maße [51].

Definition 8.2.1 (Maß des Vertrauens/Misstrauens). *Seien Ω ein Ereignisraum und $H \subseteq \Omega$ (Hypothese) und $E \subseteq \Omega$ (Evidenz) Ereignisse. Dann heißt*

$$MB(H|E) = \begin{cases} \frac{P(H|E) - P(H)}{1 - P(H)}, & \text{falls } P(H|E) > P(H), \\ 0, & \text{sonst,} \end{cases}$$

Maß des Vertrauens (measure of belief) in die Hypothese H gegeben die Evidenz E und

$$MD(H|E) = \begin{cases} \frac{P(H) - P(H|E)}{P(H)}, & \text{falls } P(H|E) < P(H), \\ 0, & \text{sonst,} \end{cases}$$

Maß des Misstrauens (measure of disbelief) in die Hypothese H gegeben die Evidenz E .

Anschaulich bedeutet das: Wenn E gilt, steigert sich die Überzeugung des Experten, dass H gilt, um $MB(H|E)$. Und analog: Wenn E gilt, steigert sich seine Überzeugung, dass H falsch ist, um $MD(H|E)$. Man sieht, dass beide Maße Werte aus dem Intervall $[0, 1]$ annehmen und höchstens eines der beiden Maße ungleich 0 sein kann. D.h., wenn ein Anzeichen für die Wahrheit einer Hypothese spricht, dann sagt es nichts über die Negation der Hypothese, und umgekehrt. Damit ist der Intuition des Experten (siehe oben) Rechnung getragen. Als Sicherheitsfaktor definiert man die Differenz der Maße.

Definition 8.2.2 (Sicherheitsfaktor). *Seien Ω ein Ereignisraum und $H \subseteq \Omega$ und $E \subseteq \Omega$ Ereignisse. Dann heißt*

$$CF(H|E) = MB(H|E) - MD(H|E)$$

Sicherheitsfaktor (certainty factor) der Hypothese H bezüglich der Evidenz E .

Da sowohl $MB(H|E) \in [0, 1]$ als auch $MD(H|E) \in [0, 1]$ gilt und höchstens eines der beiden Maße ungleich 0 ist, folgt $CF(H|E) \in [-1, 1]$. Es sei noch einmal betont, dass der Sicherheitsfaktor $CF(H|E)$ (im Gegensatz zur bedingten Wahrscheinlichkeit $P(H|E)$) *kein* absoluter Vertrauensgrad ist, sondern eine Vertrauensgradänderung.

Die Sicherheitsfaktordefinition wird in einem regelbasierten System wie MYCIN folgendermaßen angewandt: Zu jeder Regel $E \rightarrow H$ gibt es eine Angabe $CF(H|E)$. Wird E bekannt, so wird $MB(H|E) = CF(H|E)$ und $MD(H|E) = 0$ gesetzt, falls $CF(H|E) > 0$. Ist $CF(H|E) < 0$, so wird $MB(H|E) = 0$ und $MD(H|E) = -CF(H|E)$ gesetzt.

Wir müssen nun noch berücksichtigen, dass wir während des Schließens mit Sicherheitsfaktoren nicht immer von der gleichen A-priori-Wahrscheinlichkeit $P(H)$ ausgehen können. Denn wir erhalten ja schrittweise mehr Evidenz, etwa eine Folge E_1, E_2, \dots , und die bereits einbezogene Evidenz, die wir mit e bezeichnen, hat natürlich das Vertrauen in die Hypothese H geändert. Dieses veränderte Vertrauen $P(H|e)$ ist der Ausgangspunkt für das Einbeziehen der nächsten Evidenz. Für den ersten Schluß ist zwar $e = \Omega$, da noch keine einschränkende Information vorliegt, und folglich $P(H|e) = P(H)$, danach (im n -ten Schritt) besteht e aber aus dem Schnitt der bereits einbezogenen Evidenzen E_1, \dots, E_{n-1} . Streng genommen benötigen wir daher die folgende relativierte Definition

$$CF_e(H|E) = \begin{cases} \frac{P(H|E \cap e) - P(H|e)}{1 - P(H|e)}, & \text{falls } P(H|E \cap e) > P(H|e), \\ \frac{P(H|E \cap e) - P(H|e)}{P(H|e)}, & \text{falls } P(H|E \cap e) < P(H|e), \\ 0, & \text{falls } P(H|E \cap e) = P(H|e). \end{cases}$$

Um nicht alle Sicherheitsfaktoren für alle möglichen Evidenzen e betrachten (und vor allem speichern) zu müssen, werden (gewöhnlich implizit) Unabhängigkeitsannahmen gemacht, d.h., es wird angenommen, dass für alle $H, E, e_1, e_2 \subseteq \Omega$ gilt:

$$CF_{e_1}(H|E) = CF_{e_2}(H|E) = CF(H|E).$$

Anschaulich bedeutet das, dass die Vertrauensänderung in die Hypothese H , die durch E hervorgerufen wird, nicht von anderer Evidenz e beeinflußt werden soll: Unabhängig davon, was sonst noch bekannt ist, führt die Evidenz E immer zur gleichen Änderung des Vertrauensgrades einer Hypothese H .

8.2.2 Rechenregeln für Sicherheitsfaktoren

Außer dem schlichten, oben angegebenen Setzen des Vertrauens- und des Misstrauensmaßes bei Bekanntwerden des Antezedens einer Regel benötigt man zur Bewertung von Hypothesen weitere Rechenregeln. Bei der Aufstellung dieser Regeln wurde allerdings die ursprüngliche wahrscheinlichkeitstheoretische Interpretation des Vertrauens- und des Misstrauensmaßes weitgehend vernachlässigt, wie man insbesondere an der Regel für die parallele Kombination sehen kann (siehe unten).

Die einfachsten Rechenregeln betreffen die Konjunktion und die Disjunktion von Hypothesen. In ihnen wird die Intuition zum Ausdruck gebracht, dass die Konjunktion zweier Hypothesen nicht besser gestützt sein kann als die schlechter gestützte der beiden (eine Kette ist nur so stark wie ihr schwächstes Glied), und eine Disjunktion mindestens so gut gestützt ist wie die besser gestützte der beiden.

Definition 8.2.3 (Konjunktion von Hypothesen). *Seien H_1 und H_2 zwei Hypothesen und E eine Evidenz. Dann gilt*

$$\begin{aligned} MB(H_1 \cap H_2 | E) &= \min \{MB(H_1 | E), MB(H_2 | E)\} \\ MD(H_1 \cap H_2 | E) &= \max \{MD(H_1 | E), MD(H_2 | E)\} \end{aligned}$$

Definition 8.2.4 (Disjunktion von Hypothesen). *Seien H_1 und H_2 zwei Hypothesen und E eine Evidenz. Dann gilt*

$$\begin{aligned} MB(H_1 \cup H_2 | E) &= \max \{MB(H_1 | E), MB(H_2 | E)\} \\ MD(H_1 \cup H_2 | E) &= \min \{MD(H_1 | E), MD(H_2 | E)\} \end{aligned}$$

Die sequentielle Kombination beschreibt das Schließen bei Unsicherheit bzgl. des Erfülltseins des Antezedens einer Regel.

Definition 8.2.5 (sequentielle Kombination). *Ist die Regel $S \xrightarrow{CF(H|S)} H$ durch die Werte $MB(H|S)$ und $MD(H|S)$ gekennzeichnet, und ist das Antezedens nicht sicher wahr, d.h. ist $CF(S|E) < 1$, so setzt man*

$$\begin{aligned} MB(H|E) &= MB(H|S) \cdot \max \{0, CF(S|E)\} = MB(H|S) \cdot MB(S|E) \\ MD(H|E) &= MD(H|S) \cdot \max \{0, CF(S|E)\} = MD(H|S) \cdot MB(S|E) \end{aligned}$$

Man beachte, dass diese Berechnungsvorschrift für sicher wahres Antezedens (d.h. für $CF(S|E) = 1$) in die auf Seite 252 angegebene Vorschrift des Setzens des Vertrauens- und des Misstrauensmaßes übergeht, so dass sie auf beliebige Werte $CF(S|E)$ verallgemeinert werden kann.

Schließlich benötigt man noch eine Regel für die parallele Kombination von Evidenz, d.h. für den Fall, dass zwei Evidenzen eine Hypothese unabhängig voneinander stützen.

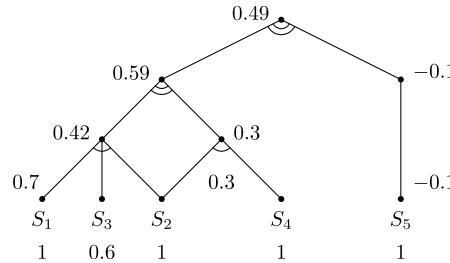


Abbildung 8.3: Darstellung der Regelauswertung aus Beispiel 8.2.2 als Netz. Einfache Bögen entsprechen einer Konjunktion von Hypothesen und der Anwendung der zugehörigen Regel, doppelte Bögen der parallelen Kombination der Resultate über die Hypothese H .

Definition 8.2.6 (parallele Kombination). Sind E_1 und E_2 zwei Evidenzen, die eine Hypothese H unabhängig voneinander stützen, und sind $MB(H|E_1)$, $MB(H|E_2)$, $MD(H|E_1)$ und $MD(H|E_2)$ bekannt, so setzt man

$$MB(H|E_1 \cap E_2) = \begin{cases} MB(H|E_1) + MB(H|E_2) & \text{falls } MD(H|E_1) < 1 \\ - MB(H|E_1) \cdot MB(H|E_2), \text{ und } MD(H|E_2) < 1 \\ 0, & \text{sonst} \end{cases}$$

$$MD(H|E_1 \cap E_2) = \begin{cases} MD(H|E_1) + MD(H|E_2) & \text{falls } MB(H|E_1) < 1 \\ - MD(H|E_1) \cdot MD(H|E_2), \text{ und } MB(H|E_2) < 1 \\ 0, & \text{sonst} \end{cases}$$

Man beachte, dass diese Kombinationsregel nicht mit der wahrscheinlichkeitstheoretischen Deutung des Vertrauens- und des Misstrauensmaßes konsistent ist, da sie zu einer Situation führen kann, in der sowohl das Vertrauensmaß als auch das Misstrauensmaß größer als 0 sind (siehe etwa das folgende Beispiel). Dies ist aber nach Definition 8.2.1 unmöglich. Im Gegensatz dazu sind die Definitionen 8.2.3 bis 8.2.5 zwar nicht aus der Wahrscheinlichkeitsdeutung herzuleiten, widersprechen ihr aber auch nicht.

Beispiel 8.2.2 (Regelauswertung in MYCIN). Gegeben sei der folgende Inhalt einer Wissensbasis zum Zeitpunkt t

($E(t)$ ist die Gesamtheit des Hintergrundwissens zum Zeitpunkt t):

Fakten: $S_1: MB(S_1|E(t)) = 1, MD(S_1|E(t)) = 0$
 $S_2: MB(S_2|E(t)) = 1, MD(S_2|E(t)) = 0$
 $S_3: MB(S_3|E(t)) = 0.6, MD(S_3|E(t)) = 0$
 $S_4: MB(S_4|E(t)) = 1, MD(S_4|E(t)) = 0$
 $S_5: MB(S_5|E(t)) = 1, MD(S_5|E(t)) = 0$

Regeln: $R_1: S_1 \cap S_2 \cap S_3 \xrightarrow{0.7} H$
 $R_2: S_2 \cap S_4 \xrightarrow{0.3} H$
 $R_3: S_5 \xrightarrow{-0.1} H$

Die Berechnung des Sicherheitsfaktors der Hypothese H ist in Abbildung 8.3 dargestellt. Wir greifen beispielhaft die abschließende Berechnung an der Spitze dieses Netzes heraus, wobei wir abkürzend S_{1-4} für $S_1 \cap S_2 \cap S_3 \cap S_4$ schreiben, S_{1-5} analog. Aus dem linken Teilbaum wissen wir, dass $CF(H|S_{1-4}) = 0.59$, also $MB(H|S_{1-4}) = 0.59$ und $MD(H|S_{1-4}) = 0$, aus dem rechten, dass $CF(H|S_5) = -0.1$, also $MB(H|S_5) = 0$ und $MD(H|S_5) = -0.1$. Die parallele Kombination ergibt:

$$\begin{aligned} MB(H|S_{1-5}) &= MB(H|S_{1-4}) + MB(H|S_5) \\ &\quad - MB(H|S_{1-4}) \cdot MB(H|S_5) \\ &= 0.59 + 0 - 0.59 \cdot 0 = 0.59 \\ MD(H|S_{1-5}) &= MD(H|S_{1-4}) + MD(H|S_5) \\ &\quad - MD(H|S_{1-4}) \cdot MD(H|S_5) \\ &= 0 + 0.1 - 0 \cdot 0.1 = 0.1 \\ \Rightarrow CF(H|S_{1-5}) &= 0.59 - 0.1 = 0.49 \end{aligned}$$

Man beachte, dass dieses Ergebnis, bei dem sowohl Vertrauens- als auch Misstrauensmaß ungleich 0 sind, nicht mit Definition 8.2.1 konsistent ist. \square

In EMYCIN [38,39] wird nur noch ein Zahlenwert, der Sicherheitsfaktor, zur Beschreibung des Vertrauens in eine Aussage verwendet. So wird die Inkonsistenz versteckt, da Vertrauens- und Misstrauensmaß nicht mehr explizit auftreten. Man benutzt bei der Kombination die folgende Regel:

Definition 8.2.7 (parallele Kombination in EMYCIN). Seien E_1 und E_2 zwei Evidenzen, die die Hypothese H unabhängig voneinander stützen, und sind $CF(H|E_1)$ und $CF(H|E_2)$ bekannt, so setzt man

$$CF(H|E_1 \cap E_2) = \begin{cases} CF(H|E_1) + CF(H|E_2) & \text{falls } CF(H|E_1) > 0 \\ -CF(H|E_1) \cdot CF(H|E_2), & \text{und } CF(H|E_2) > 0, \\ CF(H|E_1) + CF(H|E_2) & \text{falls } CF(H|E_1) < 0 \\ + CF(H|E_1) \cdot CF(H|E_2), & \text{und } CF(H|E_2) < 0, \\ \frac{CF(H|E_1) + CF(H|E_2)}{1 - \min\{|CF(H|E_1)|, |CF(H|E_2)|\}}, & \text{falls } CF(H|E_1) \\ & \quad \cdot CF(H|E_2) \in (-1, 0], \\ \text{undefiniert,} & \text{sonst.} \end{cases}$$

Für $CF(H|E_1) \cdot CF(H|E_2) = -1$ ist $CF(H|E_1 \cap E_2)$ undefiniert, da dann offenbar ein Widerspruch vorliegt (eine der beiden Evidenzen zeigt an, dass die Hypothese sicher gilt, die andere, dass sie sicher nicht gilt). In diesem Fall wird der Schlußfolgerungsvorgang abgebrochen. Man beachte außerdem, dass sich die Definition für den Fall $CF(H|E_1) \cdot CF(H|E_2) \in (-1, 0]$ von der Definition über das Vertrauens- und Misstrauensmaß unterscheidet. Die Berechnungsformel für diesen Fall wurde nach empirischen Studien festgelegt [38].

8.2.3 Inkonsistenz der Originaldefinition

Wir hatten bereits bemerkt, dass die Regel für die parallele Kombination nicht mit der Definition 8.2.1 konsistent ist. Die Frage, die sich damit stellt, ist, ob die Kombinationsregel oder die

Definition der Maße (und damit des Sicherheitsfaktors) aufgegeben werden sollte, um zu einem konsistenten Kalkül zu gelangen. Es zeigt sich, dass die Definition der Maße aufgegeben werden muß. Die Begründung liefert das folgende Beispiel, in dem wir die Evidenzkombination auf der Grundlage der Definitionen 8.2.1 und 8.2.2 und der Unabhängigkeitsannahmen untersuchen.

Beispiel 8.2.3 (Nichtkommutativität der Evidenzkombination). *Gegeben seien die Regeln $E_1 \xrightarrow{0.9} H$ und $E_2 \xrightarrow{-0.9} H$. Das Vertrauen in die Hypothese H vor dem Bekanntwerden der Evidenzen E_1 und E_2 , beschrieben als subjektive Wahrscheinlichkeit $P(H)$, sei 0.5. Wir untersuchen, wie sich das Vertrauen in H ändert, wenn 1. erst E_1 und dann E_2 und 2. erst E_2 und dann E_1 bekannt werden. Dabei stützen wir uns auf Definition 8.2.2 und die Unabhängigkeitsannahmen (siehe Seite 252).*

1. Erst E_1 , dann E_2 bekannt: Es ist $P(H) = 0.5$ und $CF(H|E_1) = 0.9$ und folglich

$$CF(H|E_1) = \frac{P(H|E_1) - P(H)}{1 - P(H)}, \quad \text{also } 0.9 = \frac{P(H|E_1) - 0.5}{1 - 0.5},$$

und damit $P(H|E_1) = 0.95$. Weiter ist $CF_{E_1}(H|E_2) = CF(H|E_2) = -0.9$, folglich

$$CF_{E_1}(H|E_2) = \frac{P(H|E_1 \cap E_2) - P(H|E_1)}{P(H|E_1)},$$

$$\text{also } -0.9 = \frac{P(H|E_1 \cap E_2) - 0.95}{0.95},$$

und damit $P(H|E_1 \cap E_2) = 0.095$.

2. Erst E_2 , dann E_1 bekannt: Es ist $P(H) = 0.5$ und $CF(H|E_2) = -0.9$ und folglich

$$CF(H|E_2) = \frac{P(H|E_2) - P(H)}{P(H)}, \quad \text{also } -0.9 = \frac{P(H|E_2) - 0.5}{0.5},$$

und damit $P(H|E_2) = 0.05$. Weiter ist $CF_{E_2}(H|E_1) = CF(H|E_1) = 0.9$, folglich

$$CF_{E_2}(H|E_1) = \frac{P(H|E_1 \cap E_2) - P(H|E_2)}{1 - P(H|E_2)},$$

$$\text{also } 0.9 = \frac{P(H|E_1 \cap E_2) - 0.05}{1 - 0.05},$$

und damit $P(H|E_1 \cap E_2) = 0.905$.

Die unterschiedliche Reihenfolge des Bekanntwerdens der Evidenzen E_1 und E_2 führt also zu stark unterschiedlichem A-posteriori-Vertrauen in die Hypothese H . \square

Wie das obige Beispiel zeigt, schreibt die Definition eines Sicherheitsfaktors nach Definition 8.2.2 eine nichtkommutative Kombination von Evidenz vor. Dies ist mit unserer intuitiven Vorstellung der Kombination von Evidenz nicht vereinbar; die Originaldefinition muß daher abgelehnt werden. Die Kombinationsregeln aus Definition 8.2.6 bzw. Definition 8.2.7 sind dagegen offensichtlich kommutativ (denn die Regel bleibt unverändert, wenn man E_1 und E_2 vertauscht). Wir werden im nächsten Abschnitt sehen, dass man eine probabilistische Interpretation eines Sicherheitsfaktors finden kann, so dass man i.w. zu den in Definition 8.2.7 definierten Kombinationsregeln gelangt.

8.2.4 Korrekte probabilistische Interpretation

Eine korrekte probabilistische Interpretation eines Sicherheitsfaktors hat [17] angegeben. In dieser Interpretation behält man die ursprüngliche Idee bei, dass ein Sicherheitsfaktor die Änderung des Vertrauens in eine Hypothese durch eine Zahl im Intervall $[-1, 1]$ beschreiben soll. Formal stellen wir folgende Axiome auf:

Definition 8.2.8 (Sicherheitsfaktoraxiome).

1. Ein Sicherheitsfaktor ist eine reelle Zahl aus dem Intervall $[-1, 1]$, d.h. $\forall H, E \subseteq \Omega : -1 \leq CF(H|E) \leq 1$.
2. (Unabhängigkeitsaxiom)
Die durch einen Sicherheitsfaktor beschriebene Vertrauensänderung ist unabhängig von anderer Evidenz: $\forall E, H, e_1, e_2 \subseteq \Omega : CF_{e_1}(H|E) = CF_{e_2}(H|E) = CF(H|E)$.
3. (Existenz einer Kombinationsfunktion)
Es existiert eine in jedem Argument streng monoton wachsende, kommutative und assoziative Funktion g , so dass $CF(H|E_1 \cap E_2) = g(CF(H|E_1), CF(H|E_2))$.

Dass ein Sicherheitsfaktor eine Vertrauensänderung beschreiben soll, stellen wir so dar: Sei $Bel(H)$ (belief) das A-priori-, $Bel(H|E)$ das A-posteriori-Vertrauen (nach Bekanntwerden der Evidenz E) in die Hypothese H , beschrieben durch jeweils eine reelle Zahl aus dem Intervall $[0, 1]$. Dann existiert eine Funktion f ,

$$f : [0, 1] \times [-1, 1] \rightarrow [0, 1], \quad Bel(H|E) = f(Bel(H), CF(H|E)),$$

die die durch $CF(H|E)$ beschriebene Vertrauensänderung in das Vertrauensmaß Bel einrechnet. In einer probabilistischen Interpretation wird man Bel als subjektive Wahrscheinlichkeit deuten. Man erhält also

$$P(H|E) = f(P(H), CF(H|E)).$$

Jedes $CF(H|E)$, das diese Beziehung und die Sicherheitsfaktoraxiome 8.2.8 erfüllt, heißt eine *probabilistische Interpretation von Sicherheitsfaktoren*. Um eine solche probabilistische Interpretation zu erhalten, definieren wir zunächst

Definition 8.2.9 (Chancen). Seien E und H Ereignisse. Dann heißen

$$O(H) = \frac{P(H)}{P(\bar{H})} = \frac{P(H)}{1 - P(H)}$$

die A-priori-Chancen (prior odds) des Eintretens von H und

$$O(H|E) = \frac{P(H|E)}{P(\bar{H}|E)} = \frac{P(H|E)}{1 - P(H|E)}$$

die A-posteriori-Chancen (posterior odds) des Eintretens von H gegeben E .

Indem wir auf die A-posteriori-Chancen den Bayesschen Satz anwenden, und zwar auf Zähler und Nenner, erhalten wir die Beziehung

$$O(H|E) = \frac{P(H|E)}{P(\bar{H}|E)} = \frac{P(E|H)}{P(\bar{E}|H)} \frac{P(H)}{P(\bar{H})} = \lambda(H|E) O(H).$$

Den Ausdruck $\lambda(H|E) = \frac{P(E|H)}{P(\bar{E}|H)}$ bezeichnen wir als *Wahrscheinlichkeitsverhältnis* (likelihood ratio)³. Die Form der obigen Gleichung legt es nahe, aus dem Wahrscheinlichkeitsverhältnis eine probabilistische Interpretation eines Sicherheitsfaktors abzuleiten.

Das einzige Hindernis besteht in den Werten, die $\lambda(H|E)$ annehmen kann, nämlich jeden Wert aus dem Intervall $[0, \infty]$, während ein Sicherheitsfaktor aus dem Intervall $[-1, 1]$ sein soll (siehe Definition 8.2.8). Es ist daher eine Transformation notwendig, d.h., wir benötigen eine Funktion $F : [0, \infty] \rightarrow [-1, 1]$, die ein Wahrscheinlichkeitsverhältnis (likelihood ratio) in einen Sicherheitsfaktor umrechnet. Eine Möglichkeit für eine solche Funktion F ist

$$F_1(x) = \begin{cases} 1 - \frac{1}{x}, & \text{falls } x \geq 1, \\ x - 1, & \text{falls } x < 1, \end{cases} \quad \text{also}$$

$$CF_1(H|E) = \begin{cases} 1 - \frac{1}{\lambda(H|E)}, & \text{falls } \lambda(H|E) \geq 1, \\ \lambda(H|E) - 1, & \text{falls } \lambda(H|E) < 1. \end{cases}$$

Die Beseitigung der Wahrscheinlichkeitsverhältnisse (likelihood ratios) durch Einsetzen der Definition führt schließlich auf

$$CF_1(H|E) = \begin{cases} \frac{P(H|E) - P(H)}{P(H|E)(1 - P(H))}, & \text{falls } P(H|E) > P(H), \\ \frac{P(H|E) - P(H)}{P(H)(1 - P(H|E))}, & \text{falls } P(H|E) < P(H), \\ 0 & \text{falls } P(H|E) = P(H). \end{cases}$$

Damit ist eine probabilistische Interpretation eines Sicherheitsfaktors gefunden. Sie unterscheidet sich von der Originaldefinition eines Sicherheitsfaktors nur durch die zusätzlichen Nennerfaktoren $P(H|E)$ im ersten und $1 - P(H|E)$ im zweiten Fall. Man kann auch leicht zeigen, dass die parallele Kombination bezüglich CF_1 i.w. der Kombinationsregel aus EMYCIN entspricht, denn für disjunkte Evidenzen E_1 und E_2 gilt:

$$\lambda(H|E_1 \cap E_2) = \frac{P(E_1 \cap E_2|H)}{P(E_1 \cap E_2|\bar{H})} = \frac{P(E_1|H)}{P(E_1|\bar{H})} \frac{P(E_2|H)}{P(E_2|\bar{H})} = \lambda(H|E_1) \cdot \lambda(H|E_2)$$

und folglich z.B. für den ersten Fall

$$\begin{aligned} CF(H|E_1 \cap E_2) &= CF(H|E_1) + CF(H|E_2) - CF(H|E_1) \cdot CF(H|E_2) \\ &= 1 - \frac{1}{\lambda(H|E_1)} + 1 - \frac{1}{\lambda(H|E_2)} \\ &\quad - \left(1 - \frac{1}{\lambda(H|E_1)}\right) \cdot \left(1 - \frac{1}{\lambda(H|E_2)}\right) \\ &= 1 - \frac{1}{\lambda(H|E_1) \cdot \lambda(H|E_2)} = 1 - \frac{1}{\lambda(H|E_1 \cap E_2)} \end{aligned}$$

³ Das Englische kennt zwei Worte – *probability* und *likelihood* – während wir im Deutschen nur das Wort *Wahrscheinlichkeit* haben. Es ist daher in diesem Fall u.U. angemessener, den englischen Ausdruck *likelihood ratio* zu verwenden, um Verwechslungen zu vermeiden.

Wir prüfen nun, was die Unabhängigkeitsaxiome in Begriffen der Wahrscheinlichkeitsrechnung bedeuten. Offenbar gilt $CF_{e_1}(H|E) = CF_{e_2}(H|E)$ für alle $E, H, e_1, e_2 \in \Omega$, wenn

$$\lambda_{e_1}(H|E) = \frac{P(E|H \cap e_1)}{P(E|\overline{H} \cap e_1)} = \frac{P(E|H \cap e_2)}{P(E|\overline{H} \cap e_2)} = \lambda_{e_2}(H|E)$$

gilt, und zwar unabhängig davon, ob $\lambda_{e_i} \geq 1$ oder $\lambda_{e_i} < 1$. Dies ist dann erfüllt, wenn für beliebige e_1 sowohl $P(E|H \cap e_1) = P(E|H)$ als auch $P(E|\overline{H} \cap e_1) = P(E|\overline{H})$ gilt, d.h. wenn E durch H und durch \overline{H} unabhängig von beliebiger anderer Evidenz ist.

Abschließend bemerken wir noch, dass die Funktion F natürlich nicht notwendigerweise so gewählt werden muß, wie hier angegeben. Vielmehr kann jede beliebige monoton wachsende Funktion $F : [0, \infty] \rightarrow [-1, 1]$ verwendet werden, die $F(\frac{1}{x}) = -F(x)$ für alle $x \neq 0$, und $F(\infty) = 1$ erfüllt. Jede solche Funktion liefert eine gültige Interpretation $CF(H|E) = F(\lambda(H|E))$ eines Sicherheitsfaktors. Umgekehrt lässt sich jede probabilistische Interpretation eines Sicherheitsfaktors auf die angegebene Art als Transformierte des Wahrscheinlichkeitsverhältnisses (likelihood ratio) darstellen.

Man kann zeigen, dass Schlussfolgerungsnetze (vgl. die Darstellung in Abbildung 8.3 auf Seite 254, in der die Schlussfolgerungen in Form eines Netzes dargestellt sind), die den Sicherheitsfaktorenansatz verwenden, folgende Bedingungen erfüllen müssen:

- Die Wertebereiche aller auftretenden Variablen enthalten genau zwei Werte,
- bedingte Unabhängigkeit von Evidenzen bezüglich Hypothesen und ihrer Negation (vgl. die obigen Betrachtungen),
- Baumstruktur des Schlussfolgerungsnetzes (d.h., eine Evidenz darf nicht auf mehr als einem Weg in die Berechnung des Sicherheitsfaktors einer Hypothese eingehen).

Für Anwendungen sind diese Bedingungen i.a. zu restriktiv. Speziell die Forderung der bedingten Unabhängigkeit ist oft nicht erfüllt. (E)MYCIN funktionierte jedoch trotzdem. Eine genauere Analyse des verwendeten Expertenwissens der medizinischen Diagnose enthüllte, dass dieses Wissen „gutartig“ ist, d.h., eine geringfügige Änderung der Sicherheitsfaktoren sich nicht signifikant auf das Systemverhalten auswirkt.

8.3 Probabilistische Schlussfolgerungsnetze

Versucht man, sich so kurz wie möglich zu fassen, so lässt sich die Idee der Schlussfolgerungsnetze wie folgt beschreiben: Unter bestimmten Bedingungen kann eine Verteilung δ (z.B. eine Wahrscheinlichkeitsverteilung) auf einem mehrdimensionalen Raum, die *A-priori-Wissen* oder *generisches Wissen* über einen bestimmten Anwendungsbereich darstellt, zerlegt werden in eine Menge $\{\delta_1, \dots, \delta_s\}$ von (ggf. überlappenden) Verteilungen auf niedrigdimensionalen Unterräumen. Wenn eine solche Zerlegung möglich ist, dann reicht es aus, die Verteilungen auf den Unterräumen zu kennen, um alle Schlussfolgerungen ziehen zu können, die man mit der ursprünglichen Verteilung δ ziehen kann. Da eine solche Zerlegung oft durch ein Netz dargestellt wird und da sie zum Ziehen von Schlussfolgerungen dient, nennen wir sie ein *Schlussfolgerungsnetz*. Ein anderer populärer Name ist *graphisches Model*, wobei „graphisch“ andeutet, dass dieses

Modell auf einem Graphen (im Sinne der Graphentheorie) beruht. Obwohl diese Beschreibung alle wesentlichen Dinge nennt, ist sie natürlich zu komprimiert, um leicht verständlich zu sein. Im folgenden erklären wir daher zunächst etwas genauer die in dieser Beschreibung benutzten Begriffe und illustrieren anschließend die Idee an einem einfachen Beispiel.

Mit *mehrdimensionalem Raum* meinen wir, dass jeder Zustand eines gegebenen, zu modellierenden Weltabschnitts beschrieben werden kann, indem man die Werte einer Menge von Attributen angibt. Wenn man z.B. Fahrzeuge beschreiben will, so könnte man den Hersteller, das Modell, die Farbe, ob bestimmte Sonderausstattungen vorhanden sind oder nicht u.ä. angeben. Jedes Attribut – oder genauer, die Menge der Werte, die es annehmen kann – bildet eine Dimension des Raumes. Damit dies möglich ist, müssen die Werte des Attributes natürlich erschöpfend sein und sich gegenseitig ausschließen. In unserem Fahrzeugbeispiel heißt das, dass die Menge der Werte des Attributes „Hersteller“ alle möglichen Hersteller umfassen muß und kein Fahrzeug das gemeinsame Produkt mehrerer Hersteller sein darf. D.h., es muß möglich sein, für jedes Fahrzeug genau einen Hersteller anzugeben. Mit diesen Einschränkungen (für alle Attribute) entspricht jedes Fahrzeug einem Punkt des mehrdimensionalen Raumes.

Natürlich kann es mehrere Fahrzeuge geben, die dem gleichen Punkt entsprechen – einfach weil diese Fahrzeuge in allen Attributen übereinstimmen (gleicher Hersteller, gleiche Farbe etc.). Andererseits kann es Punkte des Raumes geben, denen kein existierendes Fahrzeug entspricht – z.B. weil bestimmte Sonderausstattungen für ein Modell nicht erhältlich sind und es folglich keine Fahrzeuge dieses Typs mit diesen Sonderausstattungen gibt. Derartige Information ist in der Verteilung auf dem betrachteten Raum enthalten. Eine *Verteilung* δ ordnet jedem Punkt des Raumes eine Zahl aus dem Intervall $[0, 1]$ zu, die die Möglichkeit oder die (A-priori-) Wahrscheinlichkeit angibt, dass der entsprechende Zustand des modellierten Weltabschnitts vorliegt. Diese Zahlen werden gewöhnlich von Experten geschätzt oder durch statistische Analysen aus Erfahrungsdaten gewonnen. In unserem Fahrzeugbeispiel könnten diese Zahlen einfach die relative Häufigkeit angeben, mit der Fahrzeuge eines bestimmten Typs und einer bestimmten Ausstattung verkauft wurden.

Mit *Zerlegung* meinen wir, dass die Verteilung δ auf dem betrachteten Raum (wenigstens näherungsweise) aus den Verteilungen $\{\delta_1, \dots, \delta_s\}$ rekonstruiert werden kann. Eine solche Zerlegung hat verschiedene Vorteile, z.B., dass die Verteilungen $\delta_1, \dots, \delta_s$ mit weniger Redundanz gespeichert werden können als die ursprüngliche Verteilung. Diese Vorteile sind der wesentliche Grund, aus dem Zerlegungen in der Datenbanktheorie [11, 36, 54] untersucht werden. Es ist daher nicht überraschend, dass die Datenbanktheorie eng mit der Theorie der Schlussfolgerungsnetze verbunden ist. Der Unterschied ist, dass man sich bei Schlussfolgerungsnetzen auf, wie der Name ja schon sagt, das Ziehen von Schlussfolgerungen konzentriert, während die Datenbanktheorie sich stärker mit dem Speichern, Warten und Abrufen von Daten beschäftigt.

Nur in der Lage zu sein, eine Verteilung effizienter zu speichern, wäre zum Zwecke des Schlussfolgerns allerdings wenig nützlich, gäbe es nicht die Möglichkeit, die Schlussfolgerungen zu ziehen, ohne die ursprüngliche Verteilung vorher rekonstruieren zu müssen. Die Kernidee ist, Informationen lokal von Unterraum zu Unterraum weiterzugeben, bis alle Verteilungen aktualisiert sind. Diesen Vorgang nennt man gewöhnlich *Evidenzpropagation*. Wie er abläuft, lässt sich am besten an einem einfachen Beispiel zeigen. In diesem Beispiel vernachlässigen wir zunächst die Wahrscheinlichkeiten, die leicht die sehr einfache Idee verschleiern können, und betrachten nur, ob bestimmte Zustände eines gegebenen Weltabschnitts möglich sind oder nicht. D.h., wir betrachten zunächst Netze, die man *relationale Netze* nennen könnte, und erst anschließend

echte probabilistische Netze (die Möglichkeit eines Ereignisses durch seine Wahrscheinlichkeit ersetzend).

8.3.1 Ein einfaches Beispiel

Der Weltausschnitt unseres einfachen Beispiels ist eine Menge geometrischer Objekte, die in Abbildung 8.4 gezeigt sind. Diese Objekte sind durch drei Eigenschaften gekennzeichnet: Farbe, Form und Größe. Daher kann man die Menge der Objekte auch als Relation durch die in Abbildung 8.4 gezeigte Tabelle darstellen, in der jede Zeile ein Objekt beschreibt, indem Farbe, Form und Größe angegeben werden.

Es werde nun zufällig ein Objekt aus dieser Menge ausgewählt. Wir nehmen jedoch an, dass nicht alle kennzeichnenden Eigenschaften des Objektes beobachtet werden können. Man kann sich etwa vorstellen, dass jemand in einiger Entfernung ein Objekt aus einer Kiste zieht, so dass nur die Farbe, nicht aber die Form oder die Größe des Objektes erkennbar ist. Nun wissen wir aber, dass es nur zehn Objekte mit bestimmten Kombinationen der drei Eigenschaften gibt. Wie kann man diese Information ausnutzen, um etwas über die nicht beobachteten bzw. beobachtbaren Eigenschaften des zufällig gewählten Objektes zu erschließen?

Farbe	Form	Größe
■	○	klein
■	○	mittel
▨	○	klein
▨	○	mittel
▨	△	mittel
▨	△	groß
□	□	mittel
□	□	mittel
□	△	mittel
□	△	groß

Abbildung 8.4: Eine Menge geometrischer Objekte und ihre Darstellung als Tabelle.

Probleme dieser Art treten in vielen Anwendungen auf, z.B. in der medizinischen Diagnose: Ein Arzt verfügt über Lehrbuch- und Erfahrungswissen über die Abhängigkeiten zwischen Symptomen, physiologischen Zuständen und Krankheiten, ggf. im Kontext anderer Eigenschaften eines Patienten, z.B. Alter und Geschlecht. Beobachten bzw. erfragen kann er aber nur die Symptome sowie Alter, Geschlecht etc. Welche Krankheit(en) (wahrscheinlich) vorliegen, muss er mit Hilfe seines Wissens erschließen.

In unserem einfachen Beispiel geometrischer Objekte ist die Schlussfolgerung natürlich trivial: Man könnte etwa in der Tabelle alle Objekte mit einer anderen Farbe als der beobachteten aussortieren und für die restlichen die möglichen Formen und Größen zusammenstellen. Dies ist aber nur möglich, weil wir lediglich zehn Objekte mit nur drei Eigenschaften haben. In der medizinischen Diagnose können wir kaum so vorgehen, da die Tabelle, die wir hier aufstellen müssten, viel zu groß ist, um auf diese Weise bearbeitet werden zu können. Stattdessen müssen wir das medizinische Wissen des Arztes geeignet strukturieren, z.B. durch Zerlegung in Abhängigkeiten zwischen wenigen Eigenschaften.

Wie (tabellarisches) Wissen zerlegt werden kann, so dass es handhabbar wird, lässt sich bereits am Beispiel der Menge geometrischer Objekte zeigen: Die Eigenschaftstabelle dieser Objekte lässt sich ohne Verlust auf zwei kleinere Tabellen aufteilen. Um dies verdeutlichen, stellen wir den Schlussfolgerungsraum graphisch dar, wie in Abbildung 8.5 links gezeigt: wir ordnen jeder Eigenschaft eine Raumrichtung zu, so dass sich jede mögliche Kombination von Eigenschaftswerten durch einen Würfel in diesem Raum darstellen lässt. Tragen wir die zu den Objekten unseres Beispiels gehörenden Würfel in diesen Schlussfolgerungsraum ein, so ergibt sich das in Abbildung 8.5 rechts gezeigte Bild.

Nehmen wir an, dass das zufällig gewählte Objekt hellgrau ist. In der Darstellung von Abbildung 8.5 entspricht das einfache Schlussfolgern durch Einschränken auf hellgraue Objekte dem Herausschneiden der zur Farbe Hellgrau gehörenden „Scheibe“, wie es in Abbildung 8.6 gezeigt ist. Wir erhalten so, dass das Objekt kein Kreis sein kann, sondern ein Quadrat oder ein Dreieck sein muß, und dass es nicht klein sein kann, sondern mittelgroß oder groß sein muß.

Diese Schlussfolgerung lässt sich jedoch auch anders ziehen, da, wie bereits angedeutet, das Wissen über die Eigenschaftskombinationen der Objekte zerlegt werden kann, und zwar in sogenannte *Projektionen* auf zweidimensionale Unterräume des Schlussfolgerungsraums. Alle möglichen derartigen Projektionen sind in Abbildung 8.7 gezeigt. Sie ergeben sich als Schattenwürfe, wenn man sich Lichtquellen in genügender Entfernung vor, über und rechts von der Darstellung der Relation vorstellt.

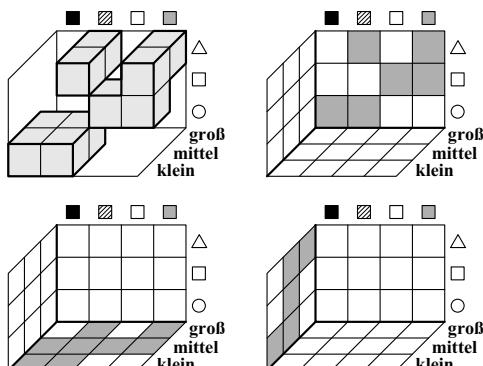


Abbildung 8.7: Alle drei zweidimensionalen Projektionen der Relation

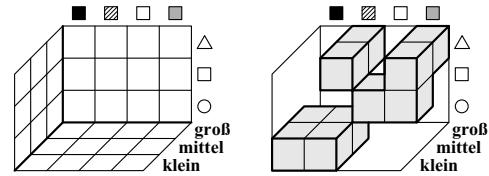


Abbildung 8.5: Schlussfolgerungsraum und graphische Darstellung der Relation.

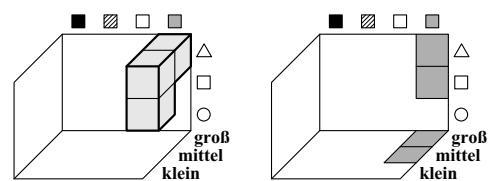


Abbildung 8.6: Direktes Schlussfolgern

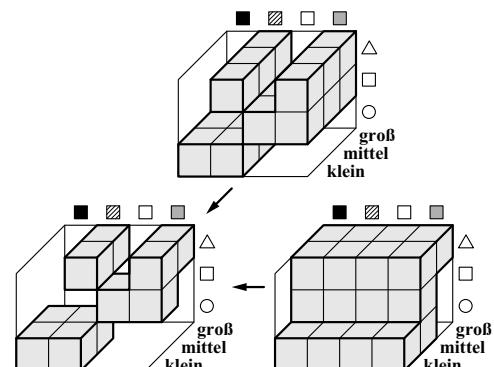


Abbildung 8.8: Zwei zylindrische Erweiterungen von Projektionen und ihr Schnitt

Wir können die Relation in die Projektionen auf die hintere und die linke Seitenfläche des Schlussfolgerungsraums zerlegen, denn aus diesen lässt sie sich rekonstruieren. Dies ist in Abbildung 8.8 gezeigt. Wir gehen folgendermaßen vor: Zunächst werden die sogenannten *zylindrischen Erweiterungen* der beiden Projektionen gebildet. D.h., wir fügen in der jeweils fehlenden Dimension alle möglichen Werte hinzu. Dies ist in Abbildung 8.8 oben und rechts gezeigt. (Der Name „zylindrische Erweiterung“ für diese Operation leitet sich übrigens von der üblichen Praxis ab, in Skizzen Mengen durch Kreisscheiben darzustellen: Fügt man zu einer Kreisscheibe eine Dimension hinzu, so erhält man einen Zylinder.) Die so erhaltenen Würfelanordnungen werden anschließend miteinander geschnitten, d.h., es bleiben nur die Würfel übrig, die in beiden zylindrischen Erweiterungen enthalten sind. Das Ergebnis ist in Abbildung 8.8 gezeigt. Wie man leicht sieht, stimmt es mit der graphischen Darstellung der Relation aus Abbildung 8.5 überein.

Der Vorteil einer Zerlegung der Relation ist, dass sie sich zum Schlussfolgern nutzen lässt, ohne dass die Darstellung im dreidimensionalen Raum vorher wiederhergestellt werden muß. Abbildung 8.9 zeigt das Vorgehen. Zuerst wird die Beobachtung, dass das Objekt hellgrau ist, auf die Projektion auf den Unterraum Farbe \times Form zylindrisch erweitert (schraffierte Spalte) und mit der Projektion der Relation auf diesen Raum (grau unterlegte Felder) geschnitten. Das Ergebnis wird auf die Dimension Form projiziert. Man erhält so, wie oben, dass das Objekt kein Kreis sein kann, sondern ein Quadrat oder ein Dreieck sein muß. Dieses Zwischenergebnis wird in analoger Weise auf den Unterraum Form \times Größe zylindrisch erweitert (schraffierte Zeilen), mit der Projektion der Relation auf diesen Raum (grau Felder) geschnitten und schließlich auf die Dimension Größe projiziert. Man erhält, wieder wie oben, dass das Objekt nicht klein sein kann, sondern mittelgroß oder groß sein muß.

Dieses Verfahren, Schlussfolgerungen zu ziehen, rechtfertigt die Darstellung durch ein Netz, wie es in Abbildung 8.10 gezeigt ist. Jeder Eigenschaft entspricht ein Knoten und die Kanten geben an, welche Projektionen benötigt werden. Zu beachten ist, dass natürlich nicht immer auf zweidimensionale Unterräume projiziert wird. Dass wir nur solche Projektionen betrachten, liegt an der Einfachheit des Beispiels. In Anwendungen können die Unterräume, auf die projiziert wird, drei, vier oder noch mehr Dimensionen haben. Entsprechend verbinden die Kanten des zugehörigen Netzes dann mehr als zwei Knoten. Eine Darstellung der Attributmengen, die so verbunden werden, durch z.B. einen Verbundbaum kann hier hilfreich sein. Diese Möglichkeit werden wir später noch etwas genauer betrachten.

Man beachte außerdem, dass die Projektionen sorgfältig gewählt werden müssen. Die Relation aus Abbildung 8.4 lässt sich nicht in zwei beliebige zweidimensionale Projektionen zerlegen. Dies zeigt Abbildung 8.11. Statt der Projektion auf die hintere Seitenfläche wird hier die Projektion auf die untere Fläche verwendet. Der Schnitt der zylindrischen Erweiterungen der beiden Projektionen (unten links gezeigt) ist jedoch deutlich verschieden von der Originalrelation, die oben rechts noch einmal dargestellt ist: Er enthält sechs zusätzliche Objekte.

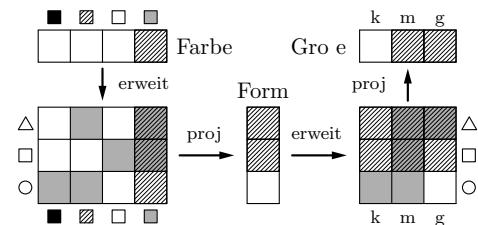


Abbildung 8.9: Evidenzpropagation



Abbildung 8.10: Netzwerkdarstellung

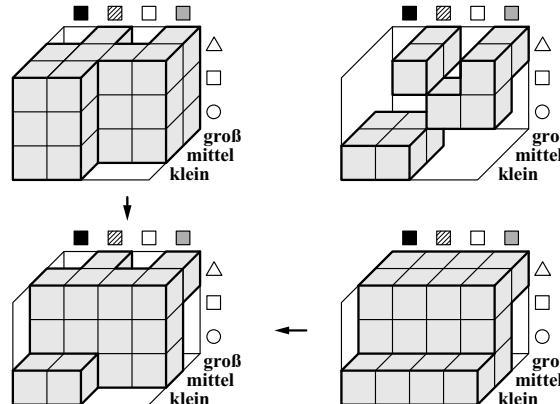


Abbildung 8.11: *Andere Projektionen*

Nicht nur müssen die Projektionen sorgfältig gewählt werden, man kann sogar nicht immer eine Zerlegung finden. Dazu betrachten wir Abbildung 8.12, in der zwei Würfel markiert sind. Nehmen wir zunächst an, das zu dem mit einer 1 markierten Würfel gehörende Objekt werde entfernt. Dann kann die Relation nicht mehr in zwei Projektionen auf zweidimensionale Unterräume zerlegt werden, wie man leicht ausprobieren kann. In diesem Fall kann man sich jedoch noch behelfen, indem man alle drei Projektionen verwendet. Da das Entfernen des Würfels 1 die Projektion auf die untere Fläche ändert, ist der Würfel 1 in der zylindrischen Erweiterung dieser Projektion und damit im Schnitt aller drei Projektionen nicht mehr enthalten. Wird dagegen das Objekt entfernt, das zu dem mit einer 2 markierten Würfel gehört, lässt sich die Relation gar nicht mehr zerlegen. Denn das Entfernen dieses Würfels ändert keine der Projektionen: In jeder Raumrichtung gibt es noch einen anderen Würfel, der den Schatten wirft.

Allerdings wird man, da das Schlussfolgern im Gesamtraum in praktischen Anwendungen wegen der großen Zahl zu betrachtender Eigenschaften i.a. unmöglich wird, versuchen, nicht (exakt) zerlegbares Wissen wenigstens näherungsweise zu zerlegen. Ziel ist es dann, eine Zerlegung zu finden, die möglichst wenige zusätzliche Eigenschaftskombinationen für möglich erklärt (eine näherungsweise Zerlegung führt offenbar höchstens zu zusätzlichen Kombinationen im Schnitt der zylindrischen Erweiterungen).

Die bisher erläuterten Ideen zur Zerlegung einer Relation lassen sich leicht auf Wahrscheinlichkeitsverteilungen übertragen. D.h., statt der Möglichkeit oder Unmöglichkeit von bestimmten Eigenschaftskombinationen betrachten wir nun ihre Wahrscheinlichkeit. Wir erweitern dazu unser Beispiel, wie in Abbildung 8.13 gezeigt, indem wir jeder Kombination der drei Eigenschaften der geometrischen Objekte eine Wahrscheinlichkeit zuordnen. Das Beispiel ist so gewählt, dass Eigenschaftskombinationen, die vorher möglich waren, nun eine hohe, solche, die unmöglich waren, eine niedrige Wahrscheinlichkeit haben. Zur Illustration: die Wahrscheinlichkeiten könnten z.B. die relative Häufigkeit der verschiedenen Objekte in einer Kiste angeben.

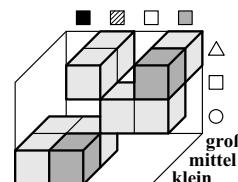


Abbildung 8.12: *Ist eine Zerlegung immer möglich?*

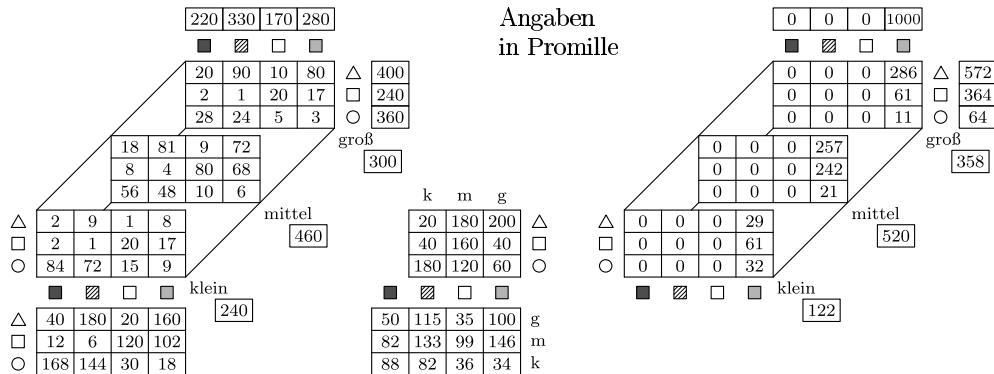


Abbildung 8.13: Eine dreidimensionale Wahrscheinlichkeitsverteilung und direktes Schlussfolgern in dieser Verteilung (d.h., Berechnung einer bedingten Verteilung).

Zu dieser Verteilung sind außerdem die *Randverteilungen* (dieser Name sagt offenbar, wo sie zu finden sind) angegeben. Sie werden durch Summieren über die herausfallende Dimension bzw. die herausfallenden Dimensionen berechnet. (Man vergleiche dazu die Herleitung des Satzes 8.1.4 über die vollständige Wahrscheinlichkeit auf Seite 245.) So ergibt sich etwa der Wert von 40 Promille für schwarze Dreiecke in der unten links in Abbildung 8.13 gezeigten Randverteilung auf dem Unterraum Farbe \times Form als Summe der Wahrscheinlichkeiten von 2, 18 und 20 Promille für kleine, mittlere und große schwarze Dreiecke, der Wert von 460 Promille für mittelgroße Objekte als Summe der Wahrscheinlichkeiten der mittleren Tafel.

Betrachten wir hier zunächst wieder ein direktes Schlussfolgern durch Einschränken auf die „Scheibe“, die dem Wert des beobachteten Attributes entspricht. Wir nehmen wieder an, dass wir durch Beobachtung feststellen, dass das Objekt hellgrau ist. Dann können wir die Wahrscheinlichkeit dieser Farbe auf den Wert 1 (denn diese Farbe ist jetzt sicher) und die Wahrscheinlichkeit aller anderen Farben auf den Wert 0 setzen (denn sie sind ja jetzt unmöglich). Entsprechend passen wir die Werte in den Tafeln des Schemas an, indem wir die zu den Farben gehörenden „Scheiben“ mit dem Quotienten der Wahrscheinlichkeiten vor und nach der Beobachtung multiplizieren (siehe Abbildung 8.13 rechts). Formal berechnen wir dadurch die bedingten Wahrscheinlichkeiten der verschiedenen Wertkombinationen gegeben die Beobachtung, dass das Objekt hellgrau ist. (Man vergleiche die Definition der bedingten Wahrscheinlichkeit in Definition 8.1.5 auf Seite 243). Durch Summation über die Zeilen bzw. Spalten der „Scheibe“ erhalten wir die bedingten (Rand-)Wahrscheinlichkeiten der Formen und Größen.

Wie die Relation aus Abbildung 8.5, so lässt sich auch die betrachtete Wahrscheinlichkeitsverteilung in die Projektionen (Randverteilungen) auf die Unterräume Farbe \times Form und Form \times Größe zerlegen, denn aus diesen lässt sie sich folgendermaßen rekonstruieren: Um z.B. die Wahrscheinlichkeit kleiner schwarzer Dreiecke zu bestimmen, multipliziert man die Wahrscheinlichkeit schwarzer Dreiecke (40 Promille) mit der Wahrscheinlichkeit kleiner Dreiecke (20 Promille) und teilt durch die (doppelt berücksichtigte) Wahrscheinlichkeit von Dreiecken (400 Promille). Man erhält 2 Promille, wie auch in der dreidimensionalen Verteilung ausgewiesen. Man prüft leicht nach, dass sich alle Werte der dreidimensionalen Verteilung auf analoge Weise berechnen lassen.

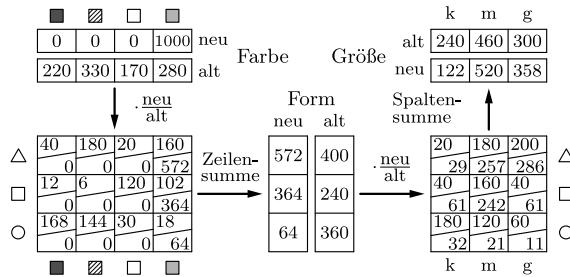


Abbildung 8.14: Propagation der Evidenz, dass das Objekt hellgrau ist (alle Angaben in Promille).

Diese Zerlegbarkeit der Verteilung lässt sich, wie im relationalen Fall, zum Schlussfolgern ausnutzen. Das Prinzip ist in Abbildung 8.14 gezeigt. Wir nehmen wieder an, dass das zufällig ausgewählte Objekt hellgrau ist. Die neue Wahrscheinlichkeit hellgrauer Objekte ist folglich 1. Wir bilden den Quotienten der neuen Wahrscheinlichkeiten mit den alten (d.h. vor der Beobachtung) und multiplizieren die Randverteilung auf dem Unterraum Farbe \times Form spaltenweise mit diesem Quotienten. Man erhält so aus den jeweils im oberen Teil eines Feldes angegebenen alten Wahrscheinlichkeiten die im unteren Teil angegebenen neuen. Das Ergebnis wird durch zeilenweise Summation auf die Dimension Form projiziert, wodurch wir die neuen Wahrscheinlichkeiten der verschiedenen Objektformen erhalten. Im zweiten Schritt gehen wir genau analog vor. Die Randverteilung auf dem Unterraum Form \times Größe wird zeilenweise mit dem Quotienten aus alter und neuer Formwahrscheinlichkeit multipliziert und anschließend durch spaltenweise Summation auf die Dimension Größe projiziert, wodurch wir die neuen Wahrscheinlichkeiten der verschiedenen Objektgrößen erhalten.

Diese Art, Schlussfolgerungen zu ziehen, rechtfertigt wieder eine Netzdarstellung, wie sie in Abbildung 8.10 gezeigt ist. Diese spezielle Art probabilistischer Netze, die mit Randverteilungen arbeitet, nennt man auch Markow-Netze (nach Andrej A. Markow, 1856–1922). Eine Alternative sind die sogenannten Bayes-Netze (nach Thomas Bayes, 1702–1761), die mit bedingten Verteilungen arbeiten und in denen die Kanten des Netzwerks gerichtet sind. Auf diese Netzwerktypen gehen wir unten noch genauer ein.

Man beachte, dass natürlich auch bei der Zerlegung von Wahrscheinlichkeitsverteilungen die Projektionen sorgfältig gewählt werden müssen, da nicht jede Wahl einer Menge von Randverteilungen eine Zerlegung darstellt, und auch Wahrscheinlichkeitsverteilungen nicht immer zerlegbar sind. Bei nicht zerlegbaren Verteilungen wird man wieder versuchen, eine möglichst gute Näherung zu finden, bei der die aus den Randverteilungen berechneten Wahrscheinlichkeiten möglichst wenig von den tatsächlichen abweichen.

8.3.2 Bedingte Unabhängigkeit

Warum lässt sich die oben betrachtete Wahrscheinlichkeitsverteilung in die Randverteilungen auf den Unterräumen Farbe \times Form und Form \times Größe zerlegen? Um diese Frage zu beantworten, stellen wir die oben nur natürlichsprachlich angegebene Berechnungsvorschrift zur Rekonstruktion der Gesamtverteilung aus den Randverteilungen zunächst formal dar. Sei das Attribut A die Farbe des Objektes, das Attribut B seine Form und das Attribut C seine Größe und seien $\text{dom}(A)$, $\text{dom}(B)$, bzw. $\text{dom}(C)$ die zugehörigen Wertebereiche. Dann gilt in der Wahrschein-

lichkeitsverteilung aus Abbildung 8.13:

$$\begin{aligned} \forall a \in \text{dom}(A) : \forall b \in \text{dom}(B) : \forall c \in \text{dom}(C) : \\ P(A = a, B = b, C = c) &= \frac{P(A = a, B = b) \cdot P(B = b, C = c)}{P(B = b)} \\ &= P(A = a, B = b) \cdot P(C = c | B = b) \\ &= P(B = b, C = c) \cdot P(A = a | B = b). \end{aligned}$$

Diese Gleichung gilt sicherlich nicht allgemein. Im allgemeinen Fall gilt lediglich (gemäß dem Produktsatz der Wahrscheinlichkeitsrechnung, siehe Satz 8.1.3 auf Seite 244)

$$\begin{aligned} \forall a \in \text{dom}(A) : \forall b \in \text{dom}(B) : \forall c \in \text{dom}(C) : \\ P(A = a, B = b, C = c) &= P(A = a, B = b) \cdot P(C = c | A = a, B = b) \\ &= P(B = b, C = c) \cdot P(A = a | B = b, C = c). \end{aligned}$$

Durch Vergleich der beiden obigen Gleichungen finden wir, dass

$$\begin{aligned} \forall a \in \text{dom}(A) : \forall b \in \text{dom}(B) : \forall c \in \text{dom}(C) : \\ P(A = a | C = c, B = b) &= P(A = a | B = b) \quad \text{und} \\ P(C = c | A = a, B = b) &= P(C = c | B = b) \end{aligned}$$

gilt. D.h., für alle Werte a, b und c müssen die Ereignisse $A = a$ und $C = c$ bedingt unabhängig sein gegeben das Ereignis $B = b$. (Vergleiche die Definition der stochastischen Unabhängigkeit von Ereignissen in Definition 8.1.6 sowie Satz 8.1.2 auf Seite 244.) Um die Quantifizierung mit allen möglichen Attributwerten zu sparen, sagt man abkürzend, dass die Attribute A und C bedingt unabhängig seien gegeben das Attribut B .

Diese Betrachtung legt nahe, dass die bedingte Unabhängigkeit von Ereignissen eine wichtige Rolle spielt. In der Tat beruht die Möglichkeit, Zerlegungen durch Graphen darzustellen, darauf, dass die Begriffe der bedingten Unabhängigkeit von Attributen und der *Trennung* von Knoten in einem Graphen eng verwandt sind. Sogenannte *bedingte Unabhängigkeitsgraphen*, die über den Begriff der Trennung von Knoten eine Menge von bedingten Unabhängigkeitsaussagen darstellen, beschreiben eine Zerlegung einer Verteilung. Als anschauliches Beispiel betrachte man den Graphen aus Abbildung 8.10. In diesem ist offenbar der einzige Pfad vom Attribut A (Farbe) zum Attribut C (Größe) durch das Attribut B (Form) blockiert und dies drückt gerade die bedingte Unabhängigkeit von A und C gegeben B aus. Außerdem beschreibt er, wie oben angegeben, eine Zerlegung der Verteilung in Randverteilungen. Diese Beziehung zwischen Zerlegungen von Verteilungen und bedingten Unabhängigkeiten sowie ihrer Darstellung durch Graphen betrachten wir, ausgehend vom Begriff der bedingten Unabhängigkeit, etwas genauer.

Für die Untersuchung bedingter Unabhängigkeiten ist eine axiomatische Fassung günstig. Zwar könnte man auch stets auf die wahrscheinlichkeitstheoretische Unabhängigkeitsdefinition zurückgreifen, doch erweist es sich als einfacher, wenn man mit Hilfe geeigneter Axiome aus bereits bekannten bedingten Unabhängigkeiten andere erschließen kann und nicht immer wieder die Definition prüfen muss. Die Axiomatisierung des Unabhängigkeitsbegriffs hat außerdem

den Vorteil, dass man nicht an die Wahrscheinlichkeitstheorie gebunden ist, sondern auch andere Kalküle, z.B. die Possibilitätstheorie [13], betrachten und die Ergebnisse übertragen kann.

Axiome zur Beschreibung bedingter Unabhängigkeit wurden schon von [12], aber später auch unabhängig von [42] angegeben. Die folgende Definition führt außerdem eine Schreibweise für bedingte Unabhängigkeiten ein.

Definition 8.3.1. *Semi-Graphoid- und Graphoid-Axiome]* Sei U eine Menge mathematischer Objekte, $(\cdot \perp\!\!\!\perp \cdot | \cdot)$ eine dreistellige Relation auf U und W, X, Y und Z vier disjunkte Teilmengen von U . Dann heißen die Aussagen

$$\text{Symmetrie: } (X \perp\!\!\!\perp Y | Z) \Rightarrow (Y \perp\!\!\!\perp X | Z)$$

$$\text{Zerlegung: } (W \cup X \perp\!\!\!\perp Y | Z) \Rightarrow (W \perp\!\!\!\perp Y | Z) \wedge (X \perp\!\!\!\perp Y | Z)$$

$$\text{Schwache Vereinigung: } (W \cup X \perp\!\!\!\perp Y | Z) \Rightarrow (X \perp\!\!\!\perp Y | Z \cup W)$$

$$\text{Zusammenziehung: } (X \perp\!\!\!\perp Y | Z \cup W) \wedge (W \perp\!\!\!\perp Y | Z) \Rightarrow (W \cup X \perp\!\!\!\perp Y | Z)$$

die *Semi-Graphoid-Axiome*. Eine dreistellige Relation $(\cdot \perp\!\!\!\perp \cdot | \cdot)$, die die *Semi-Graphoid-Axiome* für alle W, X, Y und Z erfüllt, heißt *Semi-Graphoid*. Die obigen Aussagen und

$$\text{Schnitt: } (W \perp\!\!\!\perp Y | Z \cup X) \wedge (X \perp\!\!\!\perp Y | Z \cup W) \Rightarrow (W \cup X \perp\!\!\!\perp Y | Z)$$

heißen die *Graphoid-Axiome*. Eine dreistellige Relation $(\cdot \perp\!\!\!\perp \cdot | \cdot)$, die die *Graphoid-Axiome* für alle W, X, Y und Z erfüllt, heißt *Graphoid*.

Mit der in dieser Definition genannten Menge U ist natürlich die Menge der Attribute gemeint, die zur Beschreibung des zu modellierenden Weltrausschnitts verwendet werden. Die dreistellige Relation $(\cdot \perp\!\!\!\perp \cdot | \cdot)$ stellt einen Begriff bedingter Unabhängigkeit von Attributen bzw. Attributmengen (bezüglich eines gegebenen Unsicherheitskalküls) dar, und zwar bedeutet $X \perp\!\!\!\perp Y | Z$, dass X bedingt unabhängig ist von Y gegeben Z . Mit dieser Interpretation kann man die obigen Axiome wie folgt lesen [43]:

Das *Symmetriaxiom* sagt: Wenn bei einem Kenntnisstand Z (d.h. bei Kenntnis der Werte der Attribute in Z) gilt, dass wir dadurch, dass wir die Werte der Attribute in X herausfinden (oder kurz: die Information X erhalten), nichts Neues über die Werte der Attribute in Y erfahren, dann erfahren wir auch nichts Neues über die Werte der Attribute in X , wenn wir die Werte der Attribute in Y herausfinden. Das *Zerlegungsaxiom* behauptet: Wenn die Kombination zweier Informationen irrelevant für unser Wissen über die Werte der Attribute in X ist, dann ist auch jede einzelne irrelevant. Das Axiom der *schwachen Vereinigung* sagt, dass das Erfahren irrelevanter Information W nicht dazu führen kann, dass die (vorher) irrelevante Information Y nun relevant für unser Wissen über die Werte der Attribute in X wird. Das *Zusammenziehungsaxiom* fordert, dass, vorausgesetzt die Kenntnis von X ist irrelevant für unser Wissen über die Werte der Attribute in Y , wenn wir eine irrelevante Information W erhalten haben, die Information X auch schon vorher (vor Erhalten von W) irrelevant sein muss. Zusammen bedeuten die Axiome der schwachen Vereinigung und der Zusammenziehung, dass irrelevante Information nicht die Relevanz anderer Aussagen des Systems ändert: was relevant war, bleibt relevant, und was irrelevant war, bleibt irrelevant. Es ist plausibel, dass eine bedingte Unabhängigkeit diese Axiome erfüllen sollte.

Das *Schnittaxiom* fordert: Wenn weder die Information W , bei konstantem X , unser Wissen über Y beeinflusst, noch die Information X , bei konstantem W , unser Wissen über Y beeinflusst,

dann kann weder die Information W noch die Information X noch die Kombination dieser Informationen unser Wissen über Y beeinflussen (dass sie auch einzeln irrelevant sind, folgt mit Hilfe des Zerlegungssaxioms). Dieses Axiom ist offenbar weniger plausibel als die anderen. Zwei Informationen über die Werte zweier Attribute können durchaus zusammen oder jede für sich relevant sein für unser Wissen über den Wert eines dritten, obwohl sie beide irrelevant sind, wenn die jeweils andere konstant gehalten wird. Eine solche Situation tritt etwa dann auf, wenn es eine starke Abhängigkeit zwischen den Attributen gibt. Z.B. könnte eine 1-zu-1-Beziehung zwischen den Werten der beiden Attribute bestehen, sodass mit dem Wert des einen der Wert des anderen (implizit) festgelegt wird. Dann liefert es sicherlich keine Information über den Wert eines dritten Attributes, wenn man bei Kenntnis des Wertes des einen den Wert des anderen erfährt. Wenn man aber vorher den Wert keines der beiden Attribute kennt, kann es durchaus für unser Wissen über den Wert eines dritten Attributes relevant sein, den Wert eines der beiden (oder gleich die Werte beider) zu erfahren. Es ist daher nicht überraschend, dass die bedingte stochastische Unabhängigkeit dieses Axiom nicht für beliebige Wahrscheinlichkeitsverteilungen erfüllt. Vielmehr gilt der folgende Satz:

Satz 8.3.1. *Bedingte stochastische Unabhängigkeit erfüllt die Semi-Graphoid-Axiome. Für strikt positive Wahrscheinlichkeitsverteilungen erfüllt sie die Graphoid-Axiome.*

Der Beweis ist leicht zu führen und findet sich z.B. in [7].

Die Einführung der Semi-Graphoid- und Graphoid-Axiome ähnelt offenbar der Einführung von Systemen syntaktischer Schlussregeln (z.B. der Resolutionsregel) in der formalen Logik. In der Logik erlauben es solche Schlussregelsysteme, mit rein syntaktischen Mitteln semantische Folgerungen einer Menge von Aussagen abzuleiten. Im Falle der bedingten Unabhängigkeit sind die Semi-Graphoid- bzw. die Graphoid-Axiome (zusammen mit dem Modus Ponens) die syntaktischen Schlussregeln. Eine bedingte Unabhängigkeit I folgt dagegen *semantisch* aus einer Menge \mathcal{J} von bedingten Unabhängigkeitsaussagen, wenn sie in allen Verteilungen gilt, die alle Aussagen in \mathcal{J} erfüllen. Damit stellt sich, analog zur Logik, die Frage, ob die obigen Axiome *korrekt* und *vollständig* sind, d.h., ob sie nur semantisch korrekte Folgerungen liefern und ob auch alle semantischen Folgerungen abgeleitet werden können. Die Korrektheit wird offenbar durch den obigen Satz sichergestellt. Die von [43] aufgestellte Vermutung, die Semi-Graphoid-Axiome seien für allgemeine (also nicht nur strikt positive) Wahrscheinlichkeitsverteilungen auch vollständig, hat sich jedoch als falsch erwiesen [53].

8.3.3 Darstellung durch Graphen

Wie bereits angedeutet, ist ein Begriff von bedingter Unabhängigkeit, jedenfalls wenn er die oben genannten Axiome erfüllt, erstaunlich ähnlich zur Trennung von Knoten bzw. Knotenmengen in Graphen. Dies ermöglicht die Darstellung von (Mengen von) bedingten Unabhängigkeitsaussagen durch (Knotentrennung in) Graphen. Wir gehen daher hier zunächst auf einige Grundbegriffe der Graphentheorie ein.

Ein *Graph* ist ein Tupel $G = (V, E)$, bestehend aus einer (endlichen) Menge $V = \{A_1, \dots, A_n\}$ von n *Knoten* (*vertices*) und einer Menge $E \subseteq (V \times V) \setminus \{(A, A) \mid A \in V\}$ von *Kanten* (*edges*). Mit dieser Definition sind die Graphen, die wir betrachten, *einfach* (es gibt keine mehrfachen Kanten zwischen zwei Knoten) und *schleifenfrei* (es gibt keine Kanten von einem Knoten zu sich selbst). Wir nennen eine Kante $(A, B) \in E$ *gerichtet* (von A nach B), wenn $(B, A) \notin E$,

und *ungerichtet*, wenn auch $(B, A) \in E$. D.h. bei ungerichteten Kanten sind beide möglichen Richtungen vorhanden.

Sind zwei Knoten A und B durch eine gerichtete Kante (A, B) verbunden, so sagen wir, dass A der *Elternknoten* (oder kurz *Elter*) von B und B der *Kindknoten* (oder kurz das *Kind*) von A ist. Die *Familie* eines Knotens A besteht aus dem Knoten A selbst und seinen Elternknoten. Ein Knoten A heißt *benachbart* oder *adjazent* zu einem Knoten B , wenn $(A, B) \in E$ oder $(B, A) \in E$. Der Knoten B heißt dann auch *Nachbar* von A (und umgekehrt). In einem gerichteten Graphen sind die Nachbarn eines Knotens gerade seine Kinder und seine Eltern. Wir unterscheiden nun *gerichtete* und *ungerichtete* Graphen danach, ob sie ausschließlich gerichtete oder ausschließlich ungerichtete Kanten enthalten. Wenn nötig, werden wir bei gerichteten Graphen zur Unterscheidung \vec{G} für den Graphen und \vec{E} für die Kantenmenge schreiben. Gemischte Graphen (die sowohl gerichtete als auch ungerichtete Kanten enthalten) werden wir nicht betrachten.

Zwei Knoten A und B heißen *verbunden* in einem Graphen $G = (V, E)$, geschrieben $A \sim_G B$, wenn es eine Folge $\langle C_1, \dots, C_k \rangle$ von paarweise verschiedenen Knoten, *Pfad* genannt, gibt mit $C_1 = A, C_k = B$ und $\forall i, 1 \leq i < k : (C_i, C_{i+1}) \in E \vee (C_{i+1}, C_i) \in E$. Ein Graph $G = (V, E)$ ist ein *Baum*, wenn für jedes Knotenpaar $A, B \in V$ nur genau ein Pfad existiert, der sie verbindet. In einem Baum heißen Knoten mit nur einem Nachbarn *Blätter*, alle anderen Knoten heißen *innere Knoten*. Ein Pfad in einem gerichteten Graphen heißt *gerichtet*, wenn $\forall i, 1 \leq i < k : (C_i, C_{i+1}) \in E$, d.h. ein gerichteter Pfad folgt stets der Richtung der Kanten. Wir schreiben dann $A \nearrow_G B$ und sagen, das A ein *Vorfahre* von B und B ein *Nachfahre* von A ist. Ein gerichteter Graph heißt *kreisfrei* oder *azyklisch*, wenn $\forall A, B \in E : A \nearrow_G B \Rightarrow (B, A) \notin E$, d.h. man kann nicht zu einem Knoten zurückkehren, wenn man stets der Richtung der Kanten folgt.

Was unter „Trennung“ von Knoten und Knotenmengen in Graphen zu verstehen ist, hängt von der Art des Graphen ab. Für ungerichtete Graphen wird sie so definiert:

Definition 8.3.2 (*u-Trennung*). Sei $G = (V, E)$ ein ungerichteter Graph und X, Y und Z drei disjunkte Knotenmengen. Z *u-trennt* X und Y in G , geschrieben $\langle X \mid Z \mid Y \rangle_G$, genau dann, wenn alle Pfade von einem Knoten aus X zu einem Knoten aus Y einen Knoten aus Z enthalten. Enthält ein Pfad einen Knoten aus Z , so heißt dieser Pfad (durch Z) *blockiert*, sonst *aktiv*.

Alternativ können wir sagen: Z *u-trennt* X und Y in G genau dann, wenn es nach dem Entfernen der Knoten aus Z und der zugehörigen Kanten aus G keinen Pfad mehr von einem Knoten aus X zu einem Knoten aus Y gibt.

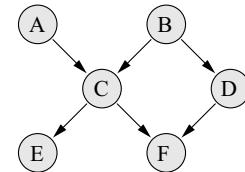
Für gerichtete Graphen wird ein etwas komplizierteres Kriterium verwendet [14, 43, 55], das weniger natürlich ist als die *u-Trennung*. Es ist ihm anzusehen, dass mit ihm die bedingten Unabhängigkeiten erfasst werden sollen, die man bei einer Zerlegung mit Hilfe des Produktsatzes ausnutzt (siehe unten).

Definition 8.3.3 (*d-Trennung*). Sei $\vec{G} = (V, \vec{E})$ ein gerichteter azyklischer Graph (d.h., ohne gerichtete Kreise) und seien X, Y und Z drei disjunkte Teilmengen von Knoten. Z *d-trennt* X und Y in \vec{G} , geschrieben $\langle X \mid Z \mid Y \rangle_{\vec{G}}$, genau dann, wenn es keinen Pfad von einem Knoten aus X zu einem Knoten aus Y gibt, auf dem die folgenden Bedingungen gelten:

1. Jeder Knoten, an dem Kanten zusammenlaufen (d.h., an dem die beiden diesen Knoten berührenden Kanten des Pfades auf diesen Knoten gerichtet sind), ist selbst aus Z oder hat einen (direkten oder indirekten) Nachfolger in Z .
2. Jeder andere Knoten ist nicht in Z .

Wenn entlang eines Pfades die beiden obigen Bedingungen gelten, so heißt dieser Pfad aktiv, anderenfalls heißt er (durch Z) blockiert.

Beispiel 8.3.1. Gegeben sei der rechts gezeigte gerichtete azyklische Graph. In diesem Graphen gilt $\langle A \mid \emptyset \mid B \rangle$, da der Pfad $A \rightarrow C \leftarrow B$ blockiert ist, wenn keine Attribute gegeben sind, aber es gilt nicht $\langle A \mid F \mid B \rangle$, da der Pfad $A \rightarrow C \leftarrow B$ durch den Nachfolger F von C aktiviert wird. Es gilt $\langle A \mid C \mid E \rangle$, da der Pfad $A \rightarrow C \rightarrow E$ durch C blockiert ist, aber nicht $\langle B \mid C \mid F \rangle$, da zwar der Pfad $B \rightarrow C \rightarrow F$ blockiert ist, nicht jedoch der Pfad $B \rightarrow D \rightarrow F$. Es gilt $\langle C \mid B \mid D \rangle$, da der Pfad $C \leftarrow B \rightarrow D$ durch B und der Pfad $C \rightarrow F \leftarrow D$ dadurch blockiert ist, dass weder F noch ein Nachfolger von F (den es hier auch gar nicht gibt) gegeben sind. \square



Sowohl u - als auch d -Trennung erfüllen die Graphoid-Axiome. Für die u -Trennung ist dies aus der in Abbildung 8.15 gezeigten Illustration der Graphoid-Axiome [43] ersichtlich. (Die Symmetrie ist trivial und wird daher vernachlässigt.) Auch für den Nachweis, dass die d -Trennung die Graphoid-Axiome erfüllt, ist diese Illustration hilfreich, doch sparen wir uns hier eine detaillierte Untersuchung.

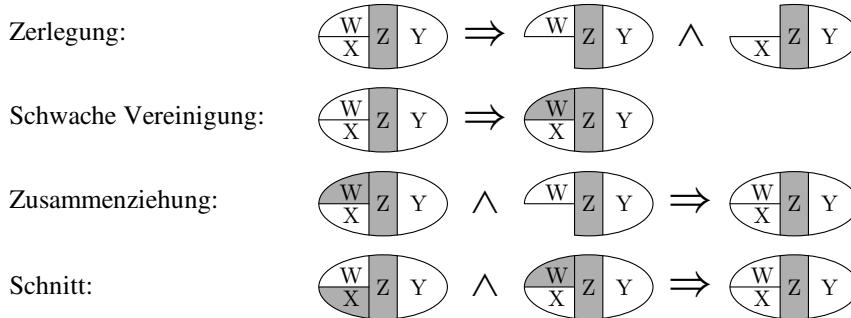


Abbildung 8.15: Illustration der Graphoid-Axiome und der Trennung in Graphen.

Die Tatsache, dass die Trennung von Knoten in Graphen die gleichen Axiome erfüllt, die auch ein Begriff bedingter Unabhängigkeit erfüllen sollte, legt es nun nahe, eine Menge bedingter Unabhängigkeitsaussagen durch einen Graphen darzustellen, z.B. alle bedingten Unabhängigkeitsaussagen, die in einer gegebenen Wahrscheinlichkeitsverteilung gelten. Im Idealfall können wir dann an dem Graphen ablesen, ob zwei Attributmengen bedingt unabhängig sind gegeben eine dritte, indem wir feststellen, ob sie durch die dritte Attributmenge in diesem Graphen getrennt werden. Leider lässt sich dieses Optimum, also die völlige Isomorphie der bedingten Unabhängigkeit von Attributen und ihrer Trennung in einem Graphen, nicht im allgemeinen Fall erreichen. Dies liegt erstens daran, dass speziell die u -Trennung Axiome erfüllt, die deutlich stärker

sind als die Graphoid-Axiome [42], so dass aus Aussagen über die Trennung von Knoten andere folgen, deren Analoga aus den entsprechenden bedingten Unabhängigkeitsaussagen nicht gefolgert werden können. Zweitens erfüllt die bedingte stochastische Unabhängigkeit im allgemeinen Fall nur die Semi-Graphoid-Axiome, so dass bereits Ableitungen mit dem Schnittaxiom ungültig sein können. Drittens gibt es Fälle, in denen bestimmte bedingte Unabhängigkeitsaussagen, die gleichzeitig in einer Verteilung gelten können, aber keine logischen Folgerungen von einander sind, nicht gleichzeitig durch einen Graphen dargestellt werden können.

Folglich müssen wir eine schwächere Definition benutzen, als die Isomorphie von bedingter Unabhängigkeit und Trennung zu fordern [43]. Diese Definition ist jedoch ausreichend, da zur Beschreibung von Zerlegungen, die ja unser eigentliches Ziel ist, diese Isomorphie zwar wünschenswert, aber nicht unerlässlich ist.

Definition 8.3.4 (bedingter (Un)Abhängigkeitsgraph). *Sei $(\cdot \perp\!\!\!\perp_{\delta} \cdot | \cdot)$ eine dreistellige Relation, die die bedingten Unabhängigkeiten darstellt, die in einer gegebenen Verteilung δ über einer Menge U von Attributen gelten. Ein ungerichteter Graph $G = (U, E)$ heißt bedingter Abhängigkeitsgraph (conditional dependence graph) oder Abhängigkeitskarte (dependence map) für δ genau dann, wenn für alle disjunkten Teilmengen $X, Y, Z \subseteq U$ gilt*

$$X \perp\!\!\!\perp_{\delta} Y | Z \Rightarrow \langle X | Z | Y \rangle_G,$$

d.h., wenn G durch u -Trennung alle (bedingten) Unabhängigkeiten erfasst, die in δ gelten. Analog heißt G bedingter Unabhängigkeitsgraph (conditional independence graph) oder Unabhängigkeitskarte (independence map) für δ genau dann, wenn für alle disjunkten Teilmengen $X, Y, Z \subseteq U$ gilt

$$\langle X | Z | Y \rangle_G \Rightarrow X \perp\!\!\!\perp_{\delta} Y | Z,$$

d.h., wenn G durch u -Trennung nur (bedingte) Unabhängigkeiten darstellt, die in δ gelten. Falls G sowohl eine Abhängigkeitskarte als auch eine Unabhängigkeitskarte für eine gegebene Verteilung δ ist, so heißt G perfekte Karte (perfect map) für δ .

Man beachte, dass ein leerer Graph eine triviale Abhängigkeitskarte, ein vollständiger Graph eine triviale Unabhängigkeitskarte ist, und dass nicht jede Verteilung eine perfekte Karte besitzt. Für gerichtete azyklische Graphen werden die Begriffe natürlich analog definiert. Allerdings ist zu bemerken, dass gerichtete und ungerichtete Graphen in Bezug auf die Darstellung bedingter Unabhängigkeiten unterschiedlich ausdrucksfähig sind. Z.B. gibt es keinen gerichteten azyklischen Graphen, der die gleichen bedingten Unabhängigkeiten darstellt wie der in Abbildung 8.16 gezeigte ungerichtete Graph, und keinen ungerichteten Graphen, der die gleichen bedingten Unabhängigkeiten darstellt wie der ebenfalls in Abbildung 8.16 gezeigte gerichtete azyklische Graph.

Im folgenden beschränken wir uns auf bedingte Unabhängigkeitsgraphen, da es ja gerade (siehe oben) die bedingten Unabhängigkeiten von Attributen sind, die es ermöglichen, eine Verteilung zu zerlegen. Der Graph, den wir benutzen, sollte uns daher keine bedingten Unabhängigkeiten ablesen lassen, die in der betrachteten Verteilung nicht gelten, denn sonst erhalten wir möglicherweise keine korrekte Zerlegung der Verteilung. Ehe wir jedoch diesen Zusammenhang genauer untersuchen können, müssen wir definieren, was es bedeutet, dass eine Verteilung bezüglich eines Graphen zerlegbar ist.

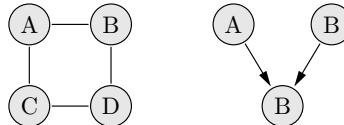
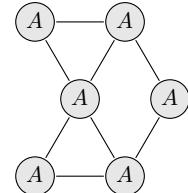


Abbildung 8.16: Ein ungerichteter Graph, dem kein gerichteter, und ein gerichteter Graph, dem kein ungerichteter Graph entspricht.

Definition 8.3.5 (faktorisierbar bzgl. eines ungerichteten Graphen). Eine Wahrscheinlichkeitsverteilung p_U über einer Menge $U = \{A_1, \dots, A_n\}$ von Attributen heißt zerlegbar oder faktorisierbar bezüglich eines ungerichteten Graphen G genau dann, wenn sie als Produkt von nichtnegativen Funktionen auf den maximalen Cliques von G geschrieben werden kann. Genauer: Sei \mathcal{M} eine Familie von Teilmengen von U , so dass die durch die Mengen $M \in \mathcal{M}$ induzierten Teilgraphen die maximalen Cliques von G sind. Sei außerdem \mathcal{E}_M die Menge der Ereignisse, die sich durch Zuweisung von Werten an alle Attribute in M beschreiben lassen. Dann heißt p_U zerlegbar oder faktorisierbar bzgl. G , wenn es Funktionen $\phi_M : \mathcal{E}_M \rightarrow \mathbb{R}_0^+$, $M \in \mathcal{M}$, gibt, so dass

$$\forall a_1 \in \text{dom}(A_1) : \dots \forall a_n \in \text{dom}(A_n) : \\ p_U \left(\bigwedge_{A_i \in U} A_i = a_i \right) = \prod_{M \in \mathcal{M}} \phi_M \left(\bigwedge_{A_i \in M} A_i = a_i \right).$$

Beispiel 8.3.2. Gegeben sei der rechts gezeigte ungerichtete Graph. Dieser Graph besitzt vier maximale Cliques, nämlich die Teilgraphen, die von den Mengen $\{A_1, A_2, A_3\}$, $\{A_3, A_5, A_6\}$, $\{A_2, A_4\}$, und $\{A_4, A_6\}$ gebildet werden. Er stellt eine Faktorisierung einer Wahrscheinlichkeitsverteilung über dem gemeinsamen Wertebereich der Attribute A_1 bis A_6 gemäß der folgenden Formel dar:



$$\forall a_1 \in \text{dom}(A_1) : \dots \forall a_6 \in \text{dom}(A_6) : \\ p_U(A_1 = a_1, \dots, A_6 = a_6) = \phi_{A_1 A_2 A_3}(A_1 = a_1, A_2 = a_2, A_3 = a_3) \\ \cdot \phi_{A_3 A_5 A_6}(A_3 = a_3, A_5 = a_5, A_6 = a_6) \\ \cdot \phi_{A_2 A_4}(A_2 = a_2, A_4 = a_4) \\ \cdot \phi_{A_4 A_6}(A_4 = a_4, A_6 = a_6). \quad \square$$

Für solche Zerlegungen gilt der folgende Satz, der üblicherweise [15] zugeschrieben wird, die ihn für den diskreten Fall bewiesen haben (auf den wir uns hier auch beschränken), obwohl, laut [35], dieses Ergebnis von verschiedenen Autoren in unterschiedlichen Formen entdeckt worden zu sein scheint.

Satz 8.3.2. Sei p_U eine strikt positive Wahrscheinlichkeitsverteilung über einer Menge U von (diskreten) Attributen. p_U ist genau dann faktorisierbar bzgl. eines ungerichteten Graphen $G = (U, E)$, wenn G ein bedingter Unabhängigkeitsgraph für p_U ist.

Einen Beweis dieses Satzes findet man z.B. in [35]. Er kann auf allgemeinere Verteilungen erweitert werden, z.B. auf Verteilungen über reellwertigen Attributen, vorausgesetzt sie haben eine positive und stetige Dichtefunktion [35].

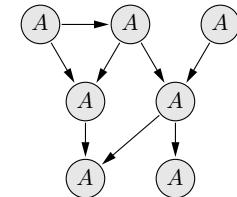
Für gerichtete Graphen wird der Begriff der Zerlegbarkeit bzw. Faktorisierbarkeit in analoger Weise eingeführt. Der einzige Unterschied besteht darin, dass man, entsprechend der Richtungen der Kanten, bedingte Wahrscheinlichkeitsverteilungen benutzt.

Definition 8.3.6 (faktorisierbar bzgl. eines gerichteten azyklischen Graphen). Eine Wahrscheinlichkeitsverteilung p_U über einer Menge $U = \{A_1, \dots, A_n\}$ von Attributen heißt zerlegbar oder faktorisierbar bezüglich eines gerichteten azyklischen Graphen \vec{G} genau dann, wenn sie geschrieben werden kann als Produkt der bedingten Wahrscheinlichkeiten der Attribute gegeben ihre Elternknoten in \vec{G} , d.h., wenn gilt

$$\forall a_1 \in \text{dom}(A_1) : \dots \forall a_n \in \text{dom}(A_n) : \\ p_U\left(\bigwedge_{A_i \in U} A_i = a_i\right) = \prod_{A_i \in U} P\left(A_i = a_i \mid \bigwedge_{A_j \in \text{parents}[\vec{G}]A_i} A_j = a_j\right),$$

wobei $\text{parents}[\vec{G}]A_i$ die Menge der Elternattribute des Attributes A_i in \vec{G} ist.

Beispiel 8.3.3. Gegeben sei der rechts gezeigte gerichtete azyklische Graph. Die durch diesen Graphen dargestellte Faktorisierung können wir ablesen, indem wir ein Produkt bilden, das einen Faktor für jedes Attribut enthält. Dieser Faktor gibt die bedingte Wahrscheinlichkeit eines Wertes dieses Attributes an, gegeben eine Belegung der Elternattribute. Wir erhalten so:



$$\forall a_1 \in \text{dom}(A_1) : \dots \forall a_7 \in \text{dom}(A_7) :$$

$$\begin{aligned} p_U(A_1 = a_1, \dots, A_7 = a_7) &= P(A_1 = a_1) \cdot P(A_2 = a_2 \mid A_1 = a_1) \cdot P(A_3 = a_3) \\ &\quad \cdot P(A_4 = a_4 \mid A_1 = a_1, A_2 = a_2) \cdot P(A_5 = a_5 \mid A_2 = a_2, A_3 = a_3) \\ &\quad \cdot P(A_6 = a_6 \mid A_4 = a_4, A_5 = a_5) \cdot P(A_7 = a_7 \mid A_5 = a_5). \end{aligned}$$

□

Für gerichtete azyklische Graphen gilt analog zu ungerichteten Graphen der Satz:

Satz 8.3.3. Sei p_U eine Wahrscheinlichkeitsverteilung über einer Menge U von (diskreten) Attributen. Die Verteilung p_U ist genau dann faktorisierbar bzgl. eines gerichteten azyklischen Graphen $\vec{G} = (U, \vec{E})$, wenn G ein bedingter Unabhängigkeitsgraph für p_U ist.

Die Faktorisierbarkeit lässt sich offenbar unmittelbar dadurch rechtfertigen, dass man den (verallgemeinerten) Produktsatz der Wahrscheinlichkeitsrechnung (siehe Satz 8.1.3 auf Seite 244) auf die Verteilung p_U anwendet und dann die durch den Graphen dargestellten bedingten Unabhängigkeiten ausnutzt, um einige der bedingenden Attribute zu streichen. Ein vollständiger Beweis findet sich z.B. in [35].

Mit dem Begriff der Faktorisierbarkeit bezüglich eines Graphen können wir schließlich die Begriffe des Markov-Netzes und des Bayes-Netzes einführen. Ein *Markov-Netz* ist ein ungerichteter bedingter Unabhängigkeitsgraph zusammen mit den Funktionen, die in der Zerlegung bezüglich dieses Graphen auftreten. Analog ist ein *Bayes-Netz* ein gerichteter azyklischer bedingter Unabhängigkeitsgraph zusammen mit den bedingten Wahrscheinlichkeitsverteilungen, die in der Zerlegung bezüglich dieses Graphen auftreten. Beide nennen wir *probabilistische (Schlussfolgerungs-)Netze*.

8.3.4 Evidenzpropagation

Um deutlich zu machen, wie man von einer Zerlegung einer mehrdimensionalen Verteilung zu einem Verfahren zur Propagation von Evidenz in dem zugehörigen Unabhängigkeitskarte gelangt, geben wir eine formale Begründung der Evidenzpropagation aus Abbildung 8.14 (Seite 266) für das Beispiel aus Abbildung 8.13 (Seite 265) an. Wie in den Erläuterungen zu der in Abbildung 8.13 rechts dargestellten Schlussfolgerung im Gesamtraum angegeben, besteht eine wahrscheinlichkeitstheoretische Schlussfolgerung in der Bestimmung der bedingten Wahrscheinlichkeiten der Werte eines Attributes gegeben die Werte der beobachteten Attribute. In diesem Beispiel haben wir, wenn wieder das Attribut A für die Farbe, das Attribut B für die Form und das Attribut C für die Größe eines Objektes stehen, zunächst die bedingten Wahrscheinlichkeiten der Werte des Attributes B zu bestimmen, gegeben dass das Attribut A den Wert a_{obs} hat. D.h., wir müssen $\forall b \in \text{dom}(B)$ berechnen:

$$\begin{aligned}
 & P(B = b \mid A = a_{\text{obs}}) \\
 &= P\left(\bigvee_{a \in \text{dom}(A)} A = a, B = b, \bigvee_{c \in \text{dom}(C)} C = c \mid A = a_{\text{obs}}\right) \\
 &\stackrel{(1)}{=} \sum_{a \in \text{dom}(A)} \sum_{c \in \text{dom}(C)} P(A = a, B = b, C = c \mid A = a_{\text{obs}}) \\
 &\stackrel{(2)}{=} \sum_{a \in \text{dom}(A)} \sum_{c \in \text{dom}(C)} P(A = a, B = b, C = c) \cdot \frac{P(A = a \mid A = a_{\text{obs}})}{P(A = a)} \\
 &\stackrel{(3)}{=} \sum_{a \in \text{dom}(A)} \sum_{c \in \text{dom}(C)} \frac{P(A = a, B = b) \cdot P(B = b, C = c)}{P(B = b)} \cdot \frac{P(A = a \mid A = a_{\text{obs}})}{P(A = a)} \\
 &= \sum_{a \in \text{dom}(A)} P(A = a, B = b) \cdot \frac{P(A = a \mid A = a_{\text{obs}})}{P(A = a)} \cdot \underbrace{\sum_{c \in \text{dom}(C)} P(C = c \mid B = b)}_{=1} \\
 &= \sum_{a \in \text{dom}(A)} P(A = a, B = b) \cdot \frac{P(A = a \mid A = a_{\text{obs}})}{P(A = a)}.
 \end{aligned}$$

In dieser Ableitung gilt (1) wegen der Kolmogorow-Axiome (siehe Definition 8.1.3 auf Seite 242) und (3) wegen der bedingten Unabhängigkeit von A und C gegeben B (siehe Seite 267), die es erlaubt, die Wahrscheinlichkeitsverteilung zu zerlegen. (2) gilt, da erstens

$$\begin{aligned}
 & P(A = a, B = b, C = c \mid A = a_{\text{obs}}) \\
 &= \begin{cases} \frac{P(A = a, B = b, C = c)}{P(A = a_{\text{obs}})}, & \text{falls } a = a_{\text{obs}}, \\ 0, & \text{sonst,} \end{cases}
 \end{aligned}$$

und zweitens

$$\frac{P(A = a, A = a_{\text{obs}})}{P(A = a)} = \begin{cases} 1, & \text{falls } a = a_{\text{obs}}, \\ 0, & \text{sonst,} \end{cases}$$

und folglich, unabhängig davon, ob $a = a_{\text{obs}}$ oder nicht,

$$\begin{aligned} & P(A = a, B = b, C = c \mid A = a_{\text{obs}}) \\ &= P(A = a, B = b, C = c) \cdot \frac{P(A = a \mid A = a_{\text{obs}})}{P(A = a)}. \end{aligned}$$

Es ist klar, dass der linke Teil von Abbildung 8.14 nur eine graphische Darstellung dieser Berechnungsformel für alle möglichen Werte des Attributes B ist.

Im zweiten Schritt der Evidenzpropagation haben wir die bedingten Wahrscheinlichkeiten der Werte des Attributes C zu berechnen, wieder gegeben, dass das Attribut A den Wert a_{obs} hat. D.h., wir müssen $\forall c \in \text{dom}(C)$ berechnen:

$$\begin{aligned} & P(C = c \mid A = a_{\text{obs}}) \\ &= P\left(\bigvee_{a \in \text{dom}(A)} A = a, \bigvee_{b \in \text{dom}(B)} B = b, C = c \mid A = a_{\text{obs}}\right) \\ &\stackrel{(1)}{=} \sum_{a \in \text{dom}(A)} \sum_{b \in \text{dom}(B)} P(A = a, B = b, C = c \mid A = a_{\text{obs}}) \\ &\stackrel{(2)}{=} \sum_{a \in \text{dom}(A)} \sum_{b \in \text{dom}(B)} P(A = a, B = b, C = c) \cdot \frac{P(A = a \mid A = a_{\text{obs}})}{P(A = a)} \\ &\stackrel{(3)}{=} \sum_{a \in \text{dom}(A)} \sum_{b \in \text{dom}(B)} \frac{P(A = a, B = b)P(B = b, C = c)}{P(B = b)} \cdot \frac{P(A = a \mid A = a_{\text{obs}})}{P(A = a)} \\ &= \sum_{b \in \text{dom}(B)} \frac{P(B = b, C = c)}{P(B = b)} \underbrace{\sum_{a \in \text{dom}(A)} P(A = a, B = b) \cdot \frac{P(A = a \mid A = a_{\text{obs}})}{P(A = a)}}_{= P(B=b \mid A=a_{\text{obs}})} \\ &= \sum_{b \in \text{dom}(B)} P(B = b, C = c) \cdot \frac{P(B = b \mid A = a_{\text{obs}})}{P(B = b)}. \end{aligned}$$

(1), (2) und (3) gelten hier aus den gleichen Gründen wie oben. Es ist klar, dass der rechte Teil von Abbildung 8.14 nur eine graphische Darstellung dieser Berechnungsformel für alle möglichen Werte des Attributes C ist.

Aus diesen beiden Berechnungen lässt sich bereits das sehr einfache Prinzip der Ableitung von Evidenzpropagationsformeln ablesen. Bei der Berechnung der bedingten Wahrscheinlichkeiten treten offenbar immer Summen über die Werte der anderen, nicht beobachteten Attribute auf. Die Zerlegung der Verteilung ermöglicht es, aus diesen Summen Faktoren, die nicht von der Summationsvariable abhängen, herauszuziehen. Die so umgeschriebenen Summen lassen sich dann oft vereinfachen. Entweder summieren sie lediglich alle bedingten Wahrscheinlichkeiten der Werte eines Attributes und sind folglich 1, oder sie sind bereits in einem früheren Schritt der Propagation berechnet worden.

Man beachte, dass diese Ableitung nicht auf ungerichtete Graphen beschränkt ist. Denn z.B. das Ergebnis der ersten Berechnung lässt sich ja auch schreiben als

$$P(B = b \mid A = a_{\text{obs}}) = P(B = b \mid A = a) \cdot P(A = a \mid A = a_{\text{obs}}),$$

was offenbar, wegen der bedingten Wahrscheinlichkeit $P(B = b | A = a)$, einer Propagation entlang einer gerichteten Kante $A \rightarrow B$ entspricht. Diese Betrachtung erklärt auch den Namen *Bayes-Netz*, denn wäre die Kante $B \rightarrow A$ gerichtet, so müßten wir

$$P(B = b | A = a_{\text{obs}}) = \frac{P(A = a | B = b) \cdot P(B = b)}{P(A = a)} \cdot P(A = a | A = a_{\text{obs}})$$

berechnen, also die Bayessche Formel (siehe Satz 8.1.5 auf Seite 245) anwenden, um die bedingte Wahrscheinlichkeit und damit gewissermaßen die Kante „umzudrehen“.

Man beachte weiter das typische Auftreten von Faktoren wie $\frac{P(A=a|A=a_{\text{obs}})}{P(A=a)}$ (für ungerichtete Graphen) oder nur $P(A = a | A = a_{\text{obs}})$ (für gerichtete Graphen), die man anschaulich *Evidenzfaktoren* nennen könnte, da sie die Evidenz wiedergeben, dass das Attribut A den Wert a_{obs} hat. Im Prinzip kann man auch bei mehr als einem beobachteten Attribut mit einem solchen einfachen Evidenzfaktor für jedes beobachtete Attribut rechnen, wenn man berücksichtigt, dass man dann zur Normalisierung der Wahrscheinlichkeiten den Faktor

$$n = \frac{\prod_{A_i \in O} P(A_i = a_{i,\text{obs}})}{P\left(\bigwedge_{A_i \in O} A_i = a_{i,\text{obs}}\right)}$$

hinzufügen muss, wobei O die Menge der beobachteten Attribute ist. (Denn i.a. wird das Produkt der Wahrscheinlichkeiten der beobachteten Werte nicht mit ihrer gemeinsamen Wahrscheinlichkeit übereinstimmen.) Dieser Faktor kann jedoch in den Rechnungen vernachlässigt werden, da er für gegebene Beobachtungen konstant ist und folglich durch eine Normalisierung ersetzt werden kann: Die Summe der (bedingten) Wahrscheinlichkeiten aller Werte eines Attributes muss ja, mit oder ohne Evidenz, stets 1 ergeben.

Mit der gleichen Vorgehensweise wie in den oben betrachteten Beispielen lassen sich leicht allgemeine Propagationsformeln für einfach zusammenhängende Unabhängigkeitskarten, also Bäume, ableiten. Auch wenn wir hier nicht auf alle Details eingehen können (ein interessanter Leser sei auf die ausführlicheren Darstellungen in z.B. [43], [23] oder [7] verwiesen), wollen wir doch versuchen, die wesentlichen Prinzipien deutlich zu machen. Wir beschränken uns dazu (zunächst) auf einfache, ungerichtete Bäume. Wie wir schon in den Beispielen gesehen haben, laufen die Rechnungen i.W. in den Verbundverteilungen ab, die den Kanten entsprechen. Es ist daher hilfreich, diese Verteilungen in den Vordergrund zu setzen, was dadurch geschieht, dass man die Attribute in einem sogenannten *Verbundbaum* darstellt. In diesem bilden die Kanten, die die Verbundverteilungen repräsentieren, die Knoten. Diese Knoten werden so verbunden, dass wieder ein Baum entsteht (d.h., u.U. wird nicht jedes Paar von Verbundverteilungen, das ein Attribut gemeinsam hat, durch eine Kante verbunden).

Den Verbundbaum für das einfache Beispiel aus Abschnitt 8.3.1 (hier gibt es nur eine Möglichkeit, während im allgemeinen Fall Wahlmöglichkeiten bestehen können) zeigt Abbildung 8.17. Man beachte, dass die einzige Kante in diesem Verbundbaum mit dem Schnitt der Attributmengen der Knoten, der auch als *Separatormenge* bezeichnet wird, belegt ist. Die Evidenzpropagation in einem Verbundbaum, der aus einem ungerichteten, azyklischen Unabhängigkeitsgraphen (also einem ungerichteten Baum) erzeugt wurde, ist sehr



Abbildung 8.17: Verbundbaum.

einfach, da die Separatormengen immer nur ein Attribut enthalten, auf das marginalisiert werden muss und von dem aus, mit Hilfe des Faktors aus neuer und alter Wahrscheinlichkeit der Werte des Attributs, die verbundene Verbundverteilung aktualisiert wird.

Betrachten wir dazu die in Abbildung 8.13 dargestellte Verteilung in Verbindung mit dem Verbundbaum aus Abbildung 8.17. Angenommen wir erhalten die Evidenz, dass die Farbe des Objektes *weiß* ist, so müssen wir die Verteilungen der anderen Attribute entsprechend anpassen. Gemäß dem Verbundbaum aktualisieren wir zuerst die Verteilung von „Farbe, Form“. Die 17% weiße Objekte stellen nun 100% der möglichen Objekte dar, was wiederum die Wahrscheinlichkeiten für die Form des Objektes beeinflusst (die nun 11.8%, 70.6% bzw. 17.6% für dreieckige, viereckige bzw. runde Objekte sind).

Die Veränderung der Verteilung des Attributes *Form* bewirkt nun wiederum eine Änderung der Verteilung des Attributes *Größe*, die in der rechts gezeigten Tabelle dargestellt ist. Damit ist die Evidenzpropagation für dieses Beispiel auch schon abgeschlossen. Diese Berechnungen entsprechen im Wesentlichen den Berechnungen, die wir schon in Abbildung 8.14 gesehen haben (für eine andere Beobachtung). Allerdings wurden die Berechnungen hier auf einem Verbundbaum ausgeführt, der die Weitergabe der Information steuert.

k	m	g	
6	53	59	△
118	470	118	□
88	59	29	○

Mit dem Verbundbaumverfahren werden nach und nach alle Verbundverteilungen und Separatowahrscheinlichkeiten gemäß der erhaltenen Evidenz angepasst. Indem jeweils innerhalb eines Verbundbaumknotens die Verteilung neu berechnet wird, werden die daraus resultierenden Verteilungsänderungen über die Separatormengen (hier: die Menge {Form} in Abbildung 8.17) an andere Verbundbaumknoten weitergereicht. In diesen werden auch wieder die zugehörigen Verteilungen neu berechnet usf.

Auf einer Baumstruktur lässt sich auch die Verbundverteilung über alle Attribute sehr einfach angeben (vgl. Abschnitt 8.3.2). Hat man einen zyklifenfreien, ungerichteten Unabhängigkeitsgraphen $G = (V, E)$, mit $V = \{A_1, \dots, A_n\}$, so gilt:

$$\forall a^{(1)} \in \text{dom}(A_1) : \dots \forall a^{(n)} \in \text{dom}(A_n) : \\ P(A_1 = a^{(1)}, \dots, A_n = a^{(n)}) = \left(\prod_{A \in V} P(A) \right) \cdot \left(\prod_{\{A, B\} \in E} \frac{P(A, B)}{P(A) \cdot P(B)} \right).$$

Mit Hilfe dieser Formel lassen sich die allgemeinen Propagationsformeln, die wir oben anschaulich anhand eines Verbundbaums beschrieben haben, recht leicht ableiten (auch wenn wir dies hier nicht im Detail tun wollen). Im Grunde geht man auf die gleiche Weise vor wie in dem oben betrachteten einfachen Beispiel (geometrische Objekte): für ein beobachtetes Attribut wird ein Evidenzfaktor hinzugefügt. Für ein Zielattribut, für dessen Werte wir die neuen Wahrscheinlichkeiten berechnen wollen, summieren wir die obige Zerlegungsformel über alle anderen Attribute. Dann können wir Faktoren, die nicht von bestimmten Summationsvariablen abhängen, aus den zugehörigen Summen herausziehen und erhalten so am Ende eine Art Kettenrechnung für die Evidenzpropagation, die die Information von Kante zu Kante (also von Knoten zu Knoten des Verbundbaums) weitergibt. Die Form der einzelnen Rechenschritte ist stets die gleiche wie schon in dem mehrfach betrachteten Beispiel: es werden Quotienten aus neuer

und alter Wahrscheinlichkeit der verschiedenen Attributwerte gebildet, mit denen eine zugehörige Verbundverteilung multipliziert wird. Diese wird dann auf das jeweils andere Attribut marginalisiert (durch Summation über die Werte des ersten Attributes).

Bei der Verallgemeinerung dieses Verfahrens treten i.W. nur zwei Probleme auf, die sich jedoch recht leicht lösen lassen, ohne dass das Verfahren an sich deutlich komplizierter wird. Erstens treten u.U. gewisse technische Schwierigkeiten auf, wenn Evidenz für mehr als ein Attribut vorliegt. Das ist natürlich im Grunde kein Problem, da man ja Evidenz über mehrere Attribute Schritt für Schritt einbeziehen kann: erst propagiert man die Evidenz eines ersten Attributes und aktualisiert alle Verteilungen des Verbundbaums. Dann propagiert man die Evidenz eines zweiten Attributes auf dem aktualisierten Verbundbaum. Der Nachteil dieses Verfahrens ist, dass dann für jedes Attribut, für das eine Evidenz vorliegt, der komplette Graph durchlaufen werden muss. Verfahren, wie sie z.B. in [7] oder [23] vorgestellt werden, benutzen stattdessen ein Nachrichtensystem, in dem die Knoten Nachrichten über Änderungen ihrer Verteilungen an ihre Nachbarn verschicken. Knoten aktualisieren ihre eigenen Verteilungen dann anhand aller eingehenden Nachrichten und versenden ihrerseits wieder Nachrichten, nachdem ihre Aktualisierung abgeschlossen ist. Eine genaue Herleitung dieses Verfahrens würde jedoch den Rahmen dieses Kapitels sprengen. Es möge hier genügen, wenn wir erwähnen, dass mit diesem Verfahren beliebig viel Evidenz (d.h. für beliebig viele Attribute) mit nur zwei Durchläufen des Graphens propagiert werden kann, nämlich einer „Sammelphase“ und einer „Verteilungsphase“.

Ein zweites Problem besteht darin, dass sich Abhängigkeiten nur selten durch eine einfache Baumstruktur darstellen lassen. Das gerade vorgestellte Verfahren lässt sich aber nur auf Baumstrukturen anwenden. In der Praxis liegen dagegen oft Strukturen vor, wie die in Abbildung 8.18 links gezeigte. Da es hier mehrere Wege von einem Knoten zu einem anderen gibt, kann es mit einem auf Bäumen basierenden Verfahren zu Berechnungsfehlern kommen. Der Grund für diese Berechnungsfehler ist, dass sich nur schwer verhindern lässt, dass eine Evidenz doppelt berücksichtigt wird, nämlich dann, wenn sie auf zwei verschiedenen Wegen zu einem Knoten gelangt. Eine mehrfache Einrechnung der gleichen Evidenz kann jedoch zu falschen Ergebnissen führen, da die Einrechnung i.W. aus der Multiplikation mit einem Faktor besteht. In einer formalen Ableitung der Formeln für die Evidenzpropagation zeigt sich dies dadurch, dass man eben nicht zu einer einfachen Kettenrechnung kommt, wie sie für das oben besprochene Verfahren notwendig wäre.

Um mit diesem Problem umzugehen, sind verschiedene Verfahren vorgeschlagen worden. Zu den bekanntesten gehört die Cliquenbaum-Propagation (clique tree propagation) [33], dessen Idee darin besteht, einen mehrfach zusammenhängenden Graphen durch Zusammenfassen von Knoten (siehe z.B. Abbildung 8.18 rechts) in eine einfach zusammenhängende Struktur, eben einen Cliquenbaum, zu überführen. Ohne auf Details eingehen zu können (siehe z.B. [7], [3]

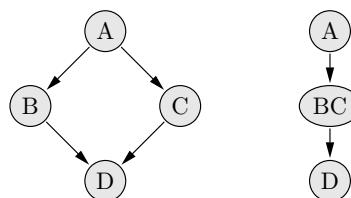


Abbildung 8.18: Durch Verschmelzen von Attributen können mehrfache Wege eliminiert werden.

oder [28]), bemerken wir hier, dass diese Umformung i.W. aus zwei Schritten besteht. Im ersten wird der Unabhängigkeitsgraph *trianguliert*: es wird durch Einfügen von Kanten sichergestellt, dass der (ungerichtete) Graph keine Kreise mit mehr als drei Knoten enthält, in denen nur die Nachbarn im Kreis durch eine Kante verbunden sind. Anschaulich: es wird sichergestellt, dass der Graph aus Dreiecken aufgebaut ist (daher auch der Name *Triangulierung*).

Triangulierte Graphen sind in einem gewissen Sinne azyklisch, nämlich wenn man die Struktur ihrer Cliques betrachtet, wobei eine Clique ein vollständiger Teilgraph ist (alle möglichen Kanten sind vorhanden). Man kann dann aus dem triangulierten Graphen einen neuen Graphen bilden, in der alle maximalen Cliques des Ursprungsgraphen Knoten sind. (Eine Clique ist maximal, wenn sich nicht Teil einer größeren Clique ist.) Aus dieser Struktur lässt sich immer ein Verbundbaum ableiten, ähnlich dem oben betrachteten, nur dass die Knoten mehr als zwei Attribute enthalten können. Außerdem kann man stets sicherstellen, dass ein Attribut, das in zwei Knoten des Verbundbaums enthalten ist, auch in allen Knoten auf dem diese Knoten verbindenden Pfad enthalten ist. Dadurch wird eine sonst mögliche inkonsistente Propagation vermieden.

Der sich ergebende Verbundbaum hat nun zwar Knoten mit mehr als zwei Attributen und ggf. auch Separatormengen mit mehr als einem Attribut, aber er kann immer noch in gleicher Weise behandelt werden wie die oben betrachteten Verbundbäume. Die auftretenden Attributmengen (speziell der Separatoren) können nämlich einfach als komplexe (Pseudo-)Attribute aufgefasst werden, die als Wertebereich das Kreuzprodukt der Wertebereiche der enthaltenen Attribute haben. Damit lassen sich alle Verfahren zur Berechnung der Verteilungsänderungen recht direkt übertragen.

8.3.5 Lernen aus Daten

Bisher haben wir weitgehend die Frage unberücksichtigt gelassen, wie man denn eine geeignete Unabhängigkeitskarte für einen gegebenen Weltausschnitt findet. Das klassische Vorgehen ist, einen menschlichen Experten zu bitten, einen solchen Graphen anzugeben. Oft wird dieser dann von einem kausalen Modell des betrachteten Weltausschnitts ausgehen und einfach die (vermuteten) kausalen Einflüsse mit den gerichteten Kanten eines Bayes-Netzes identifizieren. Ein solches Vorgehen macht zwar starke Annahmen über die Art und Weise, in der sich kausale Beziehungen statistisch zeigen (insbesondere setzt es voraus, dass ein Netz kausaler Beziehungen die durch die d -Trennung ablesbaren stochastischen Unabhängigkeiten zeigt) doch ist ein solches Vorgehen oft erfolgreich. Die bedingten Wahrscheinlichkeiten des Bayes-Netzes werden anschließend von dem Experten geschätzt oder mit statistischen Verfahren aus Erfahrungsdaten bestimmt.

Man kann jedoch auch versuchen, eine Zerlegung automatisch zu finden, indem man eine Datenbank von Beispieldaten analysiert. Das Prinzip eines solchen Lernens aus Daten wollen wir hier auch kurz anhand des oben betrachteten relationalen Beispiels erläutern. Nehmen wir an, uns sei die Tabelle aus Abbildung 8.4 auf Seite 261 gegeben und wir suchten eine (möglichst gute) Zerlegung in Projektionen auf zweidimensionale Unterräume. Wir könnten natürlich alle Möglichkeiten ausprobieren, jeweils die zusätzlichen Würfel zählen und schließlich die Möglichkeit mit der geringsten Zahl zusätzlicher Würfel wählen. Ein solches Verfahren ist allerdings für praktische Probleme wegen der sehr großen Zahl von Möglichkeiten nicht brauchbar. Angenehm wäre es, wenn man an einer Projektion schon ablesen könnte, ob man sie in der Zerlegung

Tabelle 8.1: Auswahlkriterien für relationale Projektionen.

Unterraum	relative Anzahl möglicher Wertkombinationen	Gewinn an Hartley-Information
Farbe \times Form	$\frac{6}{12} = \frac{1}{2} = 50\%$	$\log_2 \frac{12}{6} = 1$
Farbe \times Größe	$\frac{8}{12} = \frac{2}{3} \approx 67\%$	$\log_2 \frac{12}{8} \approx 0.58$
Form \times Größe	$\frac{5}{9} = \frac{5}{9} \approx 56\%$	$\log_2 \frac{9}{5} \approx 0.85$

benötigt oder nicht. In der Tat lässt sich eine Heuristik angeben, die solche Einzelbewertungen durchführt und gute Chancen verspricht, eine geeignete Zerlegung zu finden.

Die diesem Verfahren zugrundeliegende Idee ist sehr einfach: Der Schnitt der zylindrischen Erweiterungen der Projektionen soll möglichst wenige (zusätzliche) Wertkombinationen enthalten, um die gegebene Relation gut anzunähern. Nun ist es aber plausibel, dass der Schnitt wenige Wertkombinationen enthält, wenn dies schon für die zylindrischen Erweiterungen gilt. Die Zahl der Kombinationen in einer zylindrischen Erweiterung wird aber direkt von der Zahl der Wertkombinationen in der Projektion bestimmt (die zylindrische Erweiterung fügt ja nur alle Werte der fehlenden Dimension(en) hinzu). Wir sollten daher Projektionen wählen, in denen möglichst wenige Kombinationen vorkommen.

Allerdings sollte man die möglicherweise unterschiedliche Größe des Unterraums berücksichtigen – denn in großen Unterräumen wird es i.a. mehr mögliche Kombinationen geben, ohne dass dies in Bezug auf die Zerlegbarkeit etwas zu besagen hätte – und deshalb nicht die absolute, sondern die relative Zahl der Wertkombinationen bestimmen. Für unser Beispiel sind die Werte dieses Kriteriums in der zweiten Spalte der Tabelle 8.1 angegeben. Offenbar führt eine Auswahl der beiden Projektionen mit den kleinsten Werten zu der richtigen Zerlegung. In der dritten Spalte ist der Logarithmus zur Basis 2 dieser Anzahlen berechnet, den man auch als Hartley-Informationsgewinn bezeichnet [16]. Für die Auswahl von Randverteilungen einer mehrdimensionalen Wahrscheinlichkeitsverteilung lassen sich ähnliche Heuristiken angeben, wie z.B. der Shannon-Informationsgewinn [9, 30, 50]. Ein Verfahren zum Lernen eines Bayesschen Netzes, das auf einem Bayesschen Ansatz (im Sinne des Bayesschen Satzes) beruht, wurde in [10] und [18] vorgeschlagen und untersucht. Einen Überblick über Lernverfahren findet man z.B. in [3, 31].

8.4 Fuzzy-Regelsysteme

8.4.1 Einführung

Wir haben in Abschnitt 8.1.3 gesehen, dass menschliches Wissen häufig in Form von Regeln modelliert wird. Menschliches Wissen, wie z.B. „Wenn die Nachfrage nach einem Produkt steigt, kann der Preis erhöht werden.“ weist jedoch oft Unsicherheit, Vagheit oder Impräzision auf. Eine Möglichkeit, dieses Wissen geeignet zu modellieren, ist die Verwendung von *linguistischen Regeln* bzw. von *Fuzzy-Regeln* [1, 13, 27, 32, 37]. Der Unterschied zwischen „normalen“ Regeln und Fuzzy-Regeln ist die Verwendung des Konzepts der Fuzzy-Menge, vgl. Abschnitt 8.1.5. Eine Fuzzy-Regel besteht damit im allgemeinen aus einem Antezedens, das eine bestimmte Situation in Form einer unscharfen Spezifikation der Werte der Messgrößen bzw. In-

formationen beschreibt, und einem Konsequens, das eine geeignete Folgerung, Reaktion bzw. Stellgröße für diese Situation (meist unscharf) angibt. Ein typisches Beispiel für so eine Regel ist

Wenn die Temperatur sehr hoch ist **und** der Druck niedrig ist,
dann sollte der Zufluß groß sein.

Um diese Regeln auswerten zu können, sind die verwendeten sprachlichen Ausdrücke wie z.B. *ungefähr Null* oder *negativ* geeignet zu modellieren. Hierfür bietet sich die Verwendung der in Abschnitt 8.1.5 vorgestellten Fuzzy-Mengen an [60].

Ein *Fuzzy-Regelsystem* ist eine Menge von Fuzzy-Regeln, die durch einen Schlußfolgerungsmechanismus ausgewertet werden. Schon seit einigen Jahren werden Fuzzy-Regelsysteme erfolgreich im Bereich der Regelungstechnik eingesetzt. In neuerer Zeit erobern solche Systeme auch andere Bereiche wie zum Beispiel das Gebiet der Präzisionslandwirtschaft [41], der Regelung von Klimaanlagen [8] oder des Supply Chain Management [2]. Im folgenden wird daher auch von Messwerten und Stellgrößen anstelle von Eingabe- und Ausgabeinformationen gesprochen. Abbildung 8.19 zeigt das allgemeine Schema eines Fuzzy-Reglers. Ein Fuzzy-Regler besteht normalerweise aus folgenden Komponenten:

- Die *Fuzzifizierungsschnittstelle* nimmt den aktuellen Messwert auf und transformiert ihn gegebenenfalls in einen geeigneten Wertebereich durch. Danach wird der Messwert in einen linguistischen Term oder eine Fuzzy-Menge umgewandelt. Im allgemeinen wird der scharfe Messwert x_0 in die Fuzzy-Menge

$$I_{\{x_0\}} = \begin{cases} 1, & \text{falls } x = x_0 \\ 0, & \text{sonst} \end{cases}$$

transformiert.

- Die *Wissensbasis* beinhaltet Informationen über die Wertebereiche der Mess- und Stellgrößen, eventuelle Normierungen und die den linguistischen Termen zugeordneten Fuzzy-Mengen. Außerdem enthält die Wissensbasis linguistische Regeln.
- Die *Entscheidungslogik* stellt das Rechenwerk des Fuzzy-Reglers dar, in dem aus den Messgrößen mit Hilfe der Wissensbasis Informationen über die Stellgröße gewonnen werden.
- Die *Defuzzifizierungsschnittstelle* hat die Aufgabe, aus den von der Entscheidungslogik gegebenen Informationen über die Stellgröße einen scharfen Stellwert bzw. einen Ausgabewert zu bestimmen.

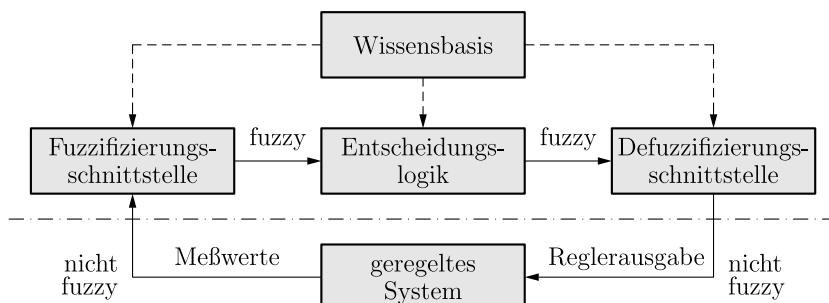


Abbildung 8.19: Architektur eines Fuzzy-Reglers.

Fuzzy-Regelsysteme werden meistens dazu verwendet, eine nicht lineare Funktion zu definieren. Sie können aber auch problemlos in anderen Anwendungsbereichen, in denen Expertenwissen vorliegt, eingesetzt werden.

8.4.2 Fuzzy-Regelsysteme nach Mamdani

Bei dem *Ansatz von Mamdani* formuliert ein Experte sein Wissen in Form von linguistischen Regeln [27, 32, 37]. Zunächst werden die linguistischen Terme festgelegt, die in den linguistischen Regeln auftreten können. Hierfür werden für jede der Wertemengen X_1, \dots, X_n (für die Eingabewerte) und Y (für den Ausgabewert) geeignete linguistische Terme wie *ungefähr Null*, *positiv klein* usw. bestimmt. Dabei kann *ungefähr Null* bezogen auf den Eingabewert x_1 durchaus etwas anderes bedeuten als für x_2 . Abbildung 8.20 zeigt eine Partitionierung, in der die linguistischen Terme *negativ*, *ungefähr Null* und *positiv* verwendet werden. Jeder dieser drei linguistischen Terme wird durch eine Fuzzy-Menge modelliert.

Formal stellen *linguistische Terme* nur Namen für die Fuzzy-Mengen bzw. für die durch die Fuzzy-Mengen repräsentierten Konzepte dar. Daher werden im folgenden zuerst die Fuzzy-Mengen festgelegt und anschließend mit passenden linguistischen Termen versehen. Hierzu wird jede der Mengen X_1, \dots, X_n und Y in Fuzzy-Mengen $\mu_1^{(i)}, \dots, \mu_{p_i}^{(i)}$ partitioniert. Häufig werden Dreiecksfunktionen der Form

$$\begin{aligned}\mu_{x_0,z} : [a, b] &\rightarrow [0, 1], \\ x &\mapsto 1 - \min\{1, z \cdot |x - x_0|\}\end{aligned}$$

verwendet. Der Wert $x_0 \in [a, b]$ gibt die Spitze des Dreiecks an. Der Parameter z bestimmt, wie spitzwinklig bzw. stumpfwinklig das Dreieck ist. Für $a < x_1 < \dots < x_{p_i} < b$ werden meist nur die Fuzzy-Mengen $\mu_2^{(i)}, \dots, \mu_{p_i}^{(i)}$ als Dreiecksmengen definiert. An den „Rändern“ des Intervalls verwendet man dagegen oft

$$\begin{aligned}\mu_1^{(i)} : [a, b] &\rightarrow [0, 1], \\ x &\mapsto \begin{cases} 1, & \text{falls } x \leq x_1 \\ 1 - \min\{1, z \cdot (x - x_1)\}, & \text{sonst} \end{cases}\end{aligned}$$

für die linke und

$$\begin{aligned}\mu_{p_i}^{(i)} : [a, b] &\rightarrow [0, 1], \\ x &\mapsto \begin{cases} 1, & \text{falls } x_{p_i} \geq x \\ 1 - \min\{1, z \cdot (x_{p_i} - x)\}, & \text{sonst} \end{cases}\end{aligned}$$

für die rechte Grenze. Abbildung 8.20 zeigt eine grobe Partitionierung einer Menge in drei Fuzzy-Mengen, denen linguistische Terme *negativ*, *ungefähr Null* und *positiv* zugeordnet sind. Daneben ist eine feinere Partitionierung dargestellt.

Dreiecksfunktionen werden vor allem deswegen häufig verwendet, weil die durch den Fuzzy-Regler auszuführenden Berechnungen bei stückweise linearen Funktionen sehr einfach sind. Prinzipiell kann jedoch jede beliebige Funktion verwendet werden.

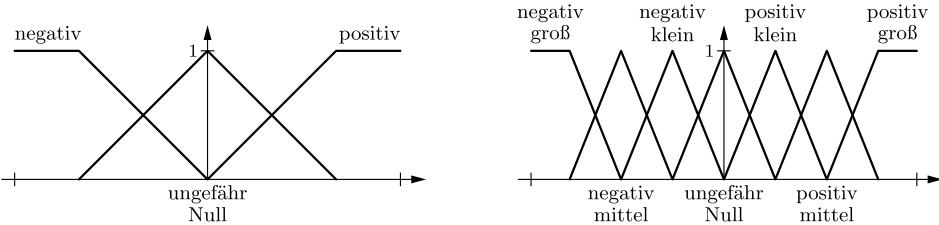


Abbildung 8.20: Eine grobe Partitionierung (links) und eine feinere (rechts).

Bei dem Ansatz von Mamdani wird ein Tupel von Eingabewerten $(x_1, \dots, x_n) \in X_1 \times \dots \times X_n$ von der Fuzzifizierungsschnittstelle direkt an die Entscheidungslogik weitergegeben. Die Entscheidungslogik wertet zunächst jede der k Regeln R_r

$$R_r : \text{if } x_1 \text{ is } A_{1,r} \text{ and } \dots \text{ und } x_n \text{ ist } A_{n,r} \text{ then } y \text{ is } B_r, \quad r = 1, \dots, k,$$

einzelne aus. Zur Auswertung wird der *Erfüllungs-* oder *Akzeptanzgrad* bestimmt, zu dem das Antezedenz bei den vorliegenden Eingabewerten erfüllt ist. Dazu wird für $v = 1, \dots, n$ der Wert $\mu(x_v)$ berechnet, der angibt, inwieweit x_v dem zu der Fuzzy-Regel gehörenden linguistischen Term entspricht. Die Werte $\mu(x_v)$ sind für die Auswertung einer Prämisse geeignet zu verknüpfen. Da eine Schlussfolgerung nicht stärker sein kann als ihre Voraussetzungen, wählt man üblicherweise als Erfüllungsgrad einer Regel das Minimum der Erfüllungsgrade der Voraussetzungen:

$$\alpha_r = \min\{\mu_{1r}(x_1), \dots, \mu_{nr}(x_n)\}.$$

Als Ausgabe der Regel R_r ergibt sich die Fuzzy-Menge von Stellwerten, die man durch „Abschneiden“ der Ausgabe-Fuzzy-Menge μ_r der Regel R_r bei dem gegebenen Zugehörigkeitsgrad erhält. In formaler Beschreibung induziert die Regel R_r bei gegebenen Messwerten (x_1, \dots, x_n) die Fuzzy-Menge

$$\begin{aligned} \mu_{\text{output}(R_r)}(x_1, \dots, x_n) : Y &\rightarrow [0, 1], \\ y &\mapsto \min\{\mu_{i_{1,r}}^{(1)}(x_1), \dots, \mu_{i_{n,r}}^{(n)}(x_n), \mu_{ir}(y)\}. \end{aligned}$$

Wenn die Antezedenzen voll erfüllt sind, liefert die Regel ihre Konsequenz-Fuzzy-Menge. Ist der Erfüllungsgrad 0, liefert die Regel eine Fuzzy-Menge, die identisch 0 ist.

Abbildung 8.21 veranschaulicht die Auswertung zweier Fuzzy-Regeln R_1 und R_2 . Die waagerecht gestrichelten Linien geben den Akzeptanzgrad für das Antezedenz der jeweiligen Regel an, so dass die Fuzzy-Menge für die Stellgröße auf dieser Höhe abzuschneiden ist. Die Ausgabe der Regel R_1 bzw. R_2 ist die schraffierte Fläche.

Nachdem die Entscheidungslogik jede Regel einzeln ausgewertet hat, werden die erhaltenen Fuzzy-Mengen mittels Maximumbildung zu einer Fuzzy-Menge μ_{output} vereinigt.

$$\begin{aligned} \mu_{\text{output}}(x_1, \dots, x_n) : Y &\rightarrow [0, 1], \\ y &\mapsto \max_{r \in \{1, \dots, k\}} \left\{ \min\{\mu_{i_{1,r}}^{(1)}(x_1), \dots, \mu_{i_{n,r}}^{(n)}(x_n), \mu_{ir}(y)\} \right\}. \end{aligned}$$

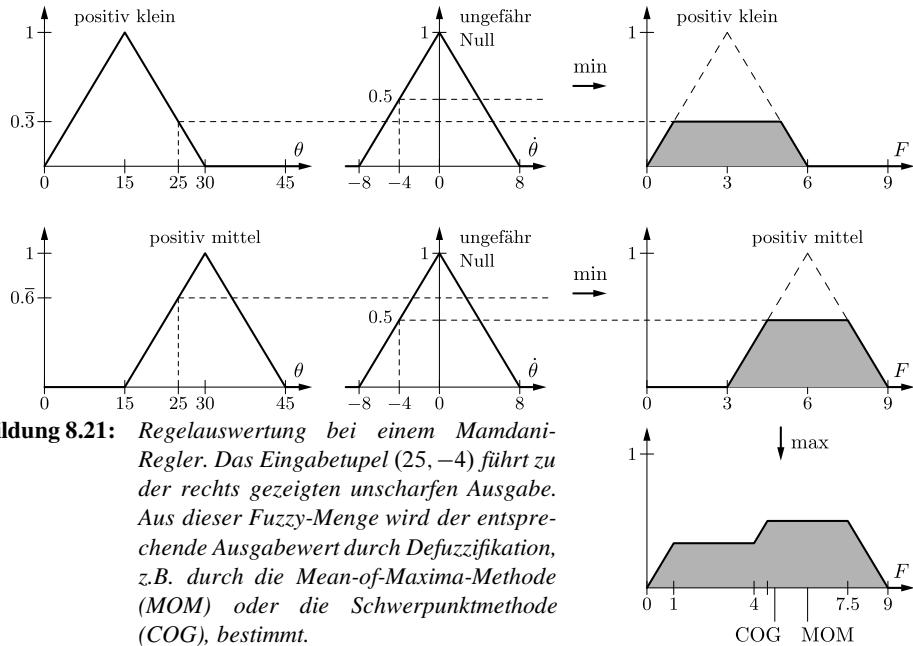


Abbildung 8.21: Regelauswertung bei einem Mamdani-Regler. Das Eingabetupel $(25, -4)$ führt zu der rechts gezeigten unscharfen Ausgabe. Aus dieser Fuzzy-Menge wird der entsprechende Ausgabewert durch Defuzzifikation, z.B. durch die Mean-of-Maxima-Methode (MOM) oder die Schwerpunktmetode (COG), bestimmt.

Die Entscheidungslogik liefert somit eine Abbildung, die jedem Tupel $(x_1, \dots, x_n) \in X_1 \times \dots \times X_n$ von Eingabewerten eine Fuzzy-Menge μ_{output} von Y zuordnet. Aus dieser Fuzzy-Menge μ_{output} wird durch die Defuzzifizierungsschnittstelle ein scharfer Ausgabewert bestimmt.

8.4.3 Defuzzifizierung

Es gibt eine Vielzahl von Verfahren zur Defuzzifizierung. Im folgenden werden exemplarisch drei gebräuchliche Methoden vorgestellt.

Die Max-Kriterium-Methode. Bei der *Max-Kriterium-Methode* ist die Ausgabe ein beliebiger Wert $y \in Y$, für den die Fuzzy-Menge $\mu(x_1, \dots, x_n)$ ihren maximalen Zugehörigkeitsgrad annimmt. Da nicht vorgegeben ist, welcher Wert zu wählen ist, repräsentiert dieses Kriterium eigentlich eine Klasse von Defuzzifizierungsstrategien. Falls der Wert zufällig gewählt wird, ist das Verhalten des Fuzzy-Reglers nicht deterministisch. Eine mit der Max-Kriterium-Methode definierte Funktion weist häufig Sprünge auf.

Die Mean-of-Maxima-Methode. Bei der *Mean-of-Maxima-Methode* (MOM) ist der defuzzifizierte Wert der Mittelwert der Menge

$$\begin{aligned} & \text{Max}(\mu(x_1, \dots, x_n)) \\ &= \{y \in Y \mid \forall y' \in Y : \mu(x_1, \dots, x_n)(y') \leq \mu(x_1, \dots, x_n)(y)\}. \end{aligned}$$

Falls die Menge $\text{Max}(\mu(x_1, \dots, x_n))$ endlich ist, ergibt sich der Stellwert zu

$$y = \frac{1}{|\text{Max}(\mu(x_1, \dots, x_n))|} \sum_{y \in \text{Max}(\mu(x_1, \dots, x_n))} y,$$

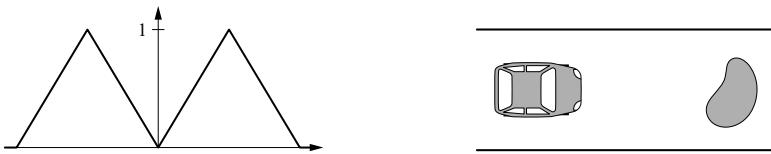


Abbildung 8.22: Eine Fuzzy-Menge, bei der die Defuzzifizierungsstrategien MOM und COG zu einem unerwünschten Ergebnis führen können. Das Auto kann bei beiden Strategien dem Hindernis nicht ausweichen.

andernfalls zu

$$y = \frac{1}{\int_{y \in \text{Max}(\mu(x_1, \dots, x_n))} dy} \cdot \int_{y \in \text{Max}(\mu(x_1, \dots, x_n))} y dy.$$

Es ist anzumerken, dass der defuzzifizierte Wert nicht notwendigerweise ein Element aus $\text{Max}(\mu(x_1, \dots, x_n))$ ist.

Die Mean-of-Maxima-Methode führt bei der Verwendung symmetrischer Dreiecksfunktionen häufig zu einem unstetigen Kurvenverlauf, da ihr Wert von der Fuzzy-Menge mit dem größten Erfüllungsgrad bestimmt wird. Solange diese Regel den größten Erfüllungsgrad aufweist, ist der defuzzifizierte Wert konstant. Sobald jedoch das Antezedenz einer anderen Regel einen größeren Erfüllungsgrad aufweist, ist der defuzzifizierte Wert der Mittelwert der dieser Regel zugehörigen Ausgabemenge.

Die Schwerpunktmetode. Bei der *Schwerpunktmetode* (center of gravity, COG) ist der Ausgabewert y der Wert, der unter dem Schwerpunkt der durch die Funktion $\mu(x_1, \dots, x_n)$ und der y -Achse begrenzten Fläche liegt. y wird daher berechnet durch:

$$y = \frac{\int_{y \in Y} y \cdot \mu(x_1, \dots, x_n)(y) dy}{\int_{y \in Y} \mu(x_1, \dots, x_n)(y) dy}.$$

Im Gegensatz zu den beiden vorhergehenden Verfahren erzeugt die Schwerpunktmetode meist einen relativ glatten Kurvenverlauf. Dies wird dadurch erreicht, dass die Regeln entsprechend ihres Erfüllungsgrads bei der Defuzzifizierung berücksichtigt werden. Es ist anzumerken, dass auch bei dieser Methode der defuzzifizierte Wert nicht notwendigerweise ein Element aus $\text{Max}(\mu(x_1, \dots, x_n))$ ist.

Beispiel für die Defuzzifizierung. Obwohl die Schwerpunkt-Methode und die Mean-of-Maxima-Methode plausibel sind und häufig der Max-Kriterium-Methode vorgezogen werden, ist für einige Anwendungen die Max-Kriterium-Methode zu bevorzugen. Wenn z.B. eine Entscheidung zwischen zwei konträren Alternativen zu treffen ist und die Antezedenzen der Regeln zum gleichen Grad erfüllt sind, liefern die Schwerpunkt- und die Mean-of-Maxima-Methode „Kompromisse“ zwischen den beiden Alternativen während die Max-Kriterium eine (willkürliche) Entscheidung herbeiführt. Dies kann durch eine Defuzzifizierung der in Abbildung 8.22 dargestellten Fuzzy-Menge gezeigt werden.

Ein einfaches Beispiel für diese Problematik ist das Ausweichen vor einem Hindernis. Hierfür kann z.B. eine Regelbasis bestehend aus den Regeln „Wenn das Hindernis links vorne ist, lenke nach rechts.“ und „Wenn das Hindernis rechts vorne ist, lenke nach links.“ verwendet werden. Wenn man genau auf das Hindernis zufährt, sind die Antezedenzen beider Regeln zum gleichen Grad erfüllt. Es ist also egal, in welche Richtung man ausweicht, man sollte jedoch auf jeden Fall ausweichen. Dieses Ergebnis erhält man nur bei der Max-Kriterium-Methode, während die Schwerpunkt-Methode und die Mean-of-Maxima-Methode dazu führen, dass man geradeaus weiterfährt.

8.4.4 Fuzzy-Regelung auf der Basis von Gleichheitsrelationen

Die im vorhergehenden Abschnitt vorgestellte Methode der Funktionsdefinition nach Mamdani kann auf der Basis von Gleichheitsrelationen interpretiert werden [25, 27]. Die Idee besteht darin, eine Funktion durch die Angabe von typischen Ein- und Ausgabetupeln sowie der Verwendung von Gleichheitsrelationen zur Modellierung von Ähnlichkeiten zu beschreiben, vgl. Abbildung 8.23.

Definition 8.4.1 (Gleichheitsrelation). Eine Abbildung $E : X \times X \rightarrow [0, 1]$ heißt Gleichheitsrelation, wenn sie die Eigenschaften

- (i) $E(x, x) = 1$,
- (ii) $E(x, y) = E(y, x)$ und
- (iii) $\max(E(x, y) + E(y, z) - 1, 0) \leq E(x, z)$

für alle $x, y, z \in X$ besitzt.

Gleichheitsrelationen modellieren das Konzept der Ähnlichkeit. Der Wert $E(x, y)$ kann als Akzeptanzgrad interpretiert werden, mit dem x und y als gleich bzw. nicht unterscheidbar angesehen werden. Häufig wird eine Gleichheitsrelation E auch als Toleranzrelation oder Ähnlichkeitsrelation bezeichnet. Fuzzy-Mengen, die eine Gleichheitsrelation in dem Sinne respektieren, dass zwei ähnliche Objekte nicht erheblich voneinander differierende Zugehörigkeitsgrade aufweisen, heißen extensional.

Fuzzy-Mengen können auf der Basis von Gleichheitsrelationen interpretiert werden. Sie beschreiben dann eine „lokale“ Ähnlichkeit. So kann man die durch die Gleichheitsrelation $E(x, y) = 1 - \min\{|x - y|/a, 1\}$ induzierte Ähnlichkeit in der Umgebung von x_0 durch die Fuzzy-Menge μ_{x_0} mit $\mu_{x_0}(x) := 1 - \min\{|x - x_0|/a, 1\}$ gut charakterisieren. Diese Fuzzy-Menge kann man als Formalisierung des Konzeptes *ungefähr x_0* ansehen. Analog kann jeder

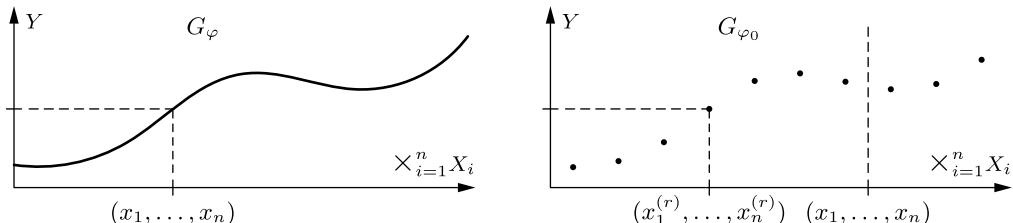


Abbildung 8.23: Die punktweise Angabe eines Graphen.

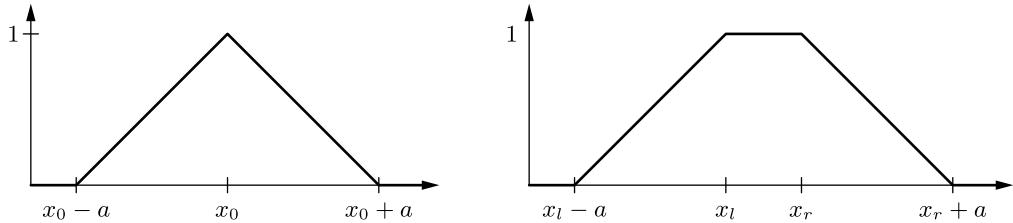


Abbildung 8.24: Zwei Fuzzy-Mengen, die durch den Wert x_0 bzw. die Menge $[x_l, x_r]$ und die Gleichheitsrelation $E(x, y) = 1 - \min\{|x - y|/a, 1\}$ spezifiziert sind.

Teilmenge von X für eine Gleichheitsrelation $E : X \times X \rightarrow [0, 1]$ die kleinste extensionale Fuzzy-Menge zugeordnet werden, vgl. Abbildung 8.24.

Definition 8.4.2 (Extensionalität). *Es sei E eine Gleichheitsrelation auf X . Eine Fuzzy-Menge μ von X , die die Bedingung*

$$\max(0, \mu(x) + E(x, y) - 1) \leq \mu(y)$$

erfüllt, heißt extensional.

Definition 8.4.3 (Extensionale Hülle). *Es sei E eine Gleichheitsrelation auf X . Die extensionale Hülle der Menge $M \subseteq X$ ist die Fuzzy-Menge*

$$\mu_M : X \rightarrow [0, 1], x \mapsto \sup\{E(x, y) \mid y \in M\}.$$

Bei der Definition einer Funktion mit Hilfe von Gleichheitsrelationen wird zunächst für die Eingabe- und Ausgabemenge die Ähnlichkeit der Werte über eine geeignete Gleichheitsrelation definiert. Man kann dazu direkt die Gleichheitsrelationen angeben oder über Fuzzy-Partitionen der Ein- und Ausgabemengen geeignete Gleichheitsrelationen berechnen, wobei die Fuzzy-Mengen der Partition dann die lokale Ähnlichkeit beschreiben [25]. Als Ergebnis erhält man Gleichheitsrelationen E_1, \dots, E_n auf X_1, \dots, X_n und F auf Y .

Im zweiten Schritt werden typische Ein- und Ausgabetupel ausgewählt, die den gewünschten Funktionsverlauf in etwa beschreiben. Hier kann man linguistische Regeln

$$R_r: \text{ if } x_1 \text{ is approximately } x_1^{(r)} \text{ and } \dots \text{ and } x_n \text{ is approximately } x_n^{(r)} \\ \text{then } y \text{ is approximately } y^{(r)}$$

dazu verwenden, eine partielle Abbildung φ_0 durch die Zuordnungen $(x_1^{(r)}, \dots, x_n^{(r)}) \mapsto y^{(r)}$ zu bestimmen.

Im dritten Schritt wird dann die partielle Funktion φ_0 zu der Gesamtfunktion φ erweitert, wobei ähnliche Eingabewerte auf ähnliche Ausgabewerte abgebildet werden. Formal wird dazu aus der gegebenen partiellen Übertragungsfunktion und den Gleichheitsrelationen die extensionale

Hülle des Graphen ermittelt. Dafür wird eine Gleichheitsrelation auf dem Produktraum $X_1 \times \dots \times X_n \times Y$ benötigt. Hierfür wird meist

$$\begin{aligned} E((x_1, \dots, x_n, y), (x'_1, \dots, x'_n, y')) \\ = \min\{E_1(x_1, x'_1), \dots, E_n(x_n, x'_n), F(y, y')\} \end{aligned}$$

verwendet. Diese Gleichheitsrelation setzt die Unabhängigkeit der Gleichheitsrelationen auf den Mengen X_1, \dots, X_n bzw. Y voraus. So darf z.B. die Ununterscheidbarkeit des Tupels (x_1, x_2, \dots, y) vom Tupel (x'_1, x_2, \dots, y) nicht von der Wahl der Werte x_2, \dots, x_n, y abhängen. Danach wird die extensionale Hülle von φ_0

$$\begin{aligned} \mu_{\varphi_0}(x'_1, \dots, x'_n, y') \\ = \max_{r \in \{1, \dots, k\}} \left\{ \min\{E_1(x_1^{(r)}, x'_1), \dots, E_n(x_n^{(r)}, x'_n), F(y^{(r)}, y')\} \right\} \end{aligned}$$

bestimmt. μ_{φ_0} ist die kleinste Fuzzy-Menge, die den Graphen von φ_0 enthält und die Gleichheitsrelation respektiert. Offenbar gilt $\mu_{\varphi_0} \leq \mu_\varphi$, wobei μ_φ die extensionale Hülle des Graphen von φ ist. Da φ nicht bekannt ist, wird die Funktion aus μ_{φ_0} bestimmt.

Für ein Tupel (x_1, \dots, x_n) induziert μ_{φ_0} die Fuzzy-Menge

$$\begin{aligned} \mu_{\varphi_0}(x_1, \dots, x_n) : Y &\rightarrow [0, 1], \\ y &\mapsto \mu_{\varphi_0}(x_1, \dots, x_n, y), \end{aligned}$$

die der Menge $\{y \in Y \mid (x_1, \dots, x_n, y) \in G_{\varphi_0}\}$ unter Berücksichtigung der Gleichheitsrelation E entspricht. Die Ausgabe entspricht der eines Mamdani-Reglers vor der Defuzzifikation.

Abbildung 8.25 veranschaulicht die Vorgehensweise für ein Fuzzy-Regelsystem mit drei linguistischen Regeln. Die partielle Funktion und die aus den Fuzzy-Mengen abgeleitete Gleichheitsrelation führt zu den drei Pyramiden. Für einen vorliegenden Eingabewert wird die grau gezeichnete Fuzzy-Menge als Ausgabe bestimmt.

Die Funktionsdefinition nach Mamdani kann also als Interpolation in Gleichheitsumgebungen aufgefaßt werden.

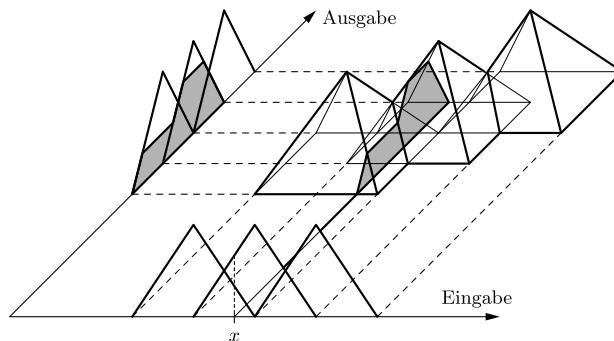


Abbildung 8.25: Ein Fuzzy-Regelsystem mit drei Regeln. Jede Pyramide wird durch eine Regel spezifiziert. Der Eingabewert x führt zu der grau gezeichneten unscharfen Ausgabe.

8.4.5 Fuzzy-Regelung und Relationalgleichungen

Ein anderes Modell zur Fuzzy-Regelung basiert auf der Verwendung von Relationalgleichungen [1, 27]. Auch bei diesem Ansatz werden linguistische Regeln der Art

if x_1 is A_1 and ... and x_n is A_n then y is B

verwendet. Der Unterschied zu der Vorgehensweise nach Mamdani bzw. über Gleichheitsrelationen ist, dass die Fuzzy-Mengen hier im Sinne der Possibilitätstheorie interpretiert werden.

Bei der *Possibilitätstheorie* [13] geht es darum, ein Maß zu definieren, dass die *Möglichkeit* angibt, dass ein Objekt oder ein Wert x_0 sich in einer Menge $A \subseteq X$ befindet. Possibilitätsverteilungen können als Pendant zur Wahrscheinlichkeitstheorie aufgefasst werden. Zur Beschreibung eines Possibilitätsmaßes $\Pi : 2^X \rightarrow [0, 1]$ werden folgende Axiome verwendet:

$$\begin{aligned}\Pi(\emptyset) &= 0, \\ \Pi(A) &\leq \Pi(B), \text{ falls } A \subseteq B \text{ und} \\ \Pi(A \cup B) &= \max\{\Pi(A), \Pi(B)\}, \text{ für alle } A, B \subset X.\end{aligned}$$

$\Pi(A) = 1$ bedeutet, dass es uneingeschränkt möglich ist, dass x_0 Element von A ist. Bei $\Pi(A) = 0$ ist es unmöglich, dass x_0 Element von A ist.

Eine einfache Möglichkeit, Possibilitätsmaße zu definieren, besteht in der Verwendung von Fuzzy-Mengen. Ist μ eine Fuzzy-Menge, so ist die Mengenfunktion

$$\Pi_\mu : 2^X \rightarrow [0, 1], A \mapsto \sup\{\mu(x) \mid x \in A\},$$

ein Possibilitätsmaß, das von der Fuzzy-Menge μ induziert wird. In diesem speziellen Fall ist das Possibilitätsmaß durch Angabe der Möglichkeitsgrade der einelementigen Mengen festgelegt. Es gilt $\Pi(\{x\}) = \mu(x)$, die Interpretation des Possibilitätsgrades und des Zugehörigkeitsgrades zur Fuzzy-Menge ist jedoch unterschiedlich.

Eine Regel R_r : „if x is μ_r then y is ν_r “ gibt an, dass der Fuzzy-Menge μ_r die Ausgabe-Fuzzy-Menge ν_r zugeordnet wird. Es gilt $\nu_r = \mu_r \circ \varrho$, wobei ϱ eine Fuzzy-Relation ist. Eine *Fuzzy-Relation* ist eine Fuzzy-Menge von $X \times Y$. Die Komposition einer Fuzzy-Menge μ mit einer Fuzzy-Relation ϱ ist definiert durch

$$\mu \circ \varrho : Y \rightarrow [0, 1], y \mapsto \sup_{x \in X} \{\min\{\mu(x), \varrho(x, y)\}\}.$$

Das Schließen bei einem Fuzzy-Regelsystem basierend auf Relationalgleichungen wird anhand des folgenden Beispiels erläutert.

Beispiel 8.4.1. Sei $X = \{k, m, g\}$ die Menge der Autoklassen Kleinwagen (k), Mittelklassewagen (m) und Autos der gehobenen Klasse (g) und $Y = \{140, 160, 180, 200, 220\}$ die Menge der möglichen Höchstgeschwindigkeiten (in km/h). Die Fuzzy-Relation ϱ (siehe Tabelle 8.2) über der Menge $X \times Y$ gibt für jedes Paar $(x, y) \in X \times Y$ an, inwieweit es für möglich gehalten wird, dass die Höchstgeschwindigkeit eines Wagens der Klasse x gerade y beträgt.

Tabelle 8.2: Die Fuzzy-Relation ϱ .

ϱ	140	160	180	200	220
k	1.0	0.5	0.1	0.0	0.0
m	0.0	0.5	1.0	0.5	0.0
g	0.0	0.0	0.4	0.8	1.0

Um etwas über die Höchstgeschwindigkeit von Fahrzeugen eines Anbieters zu erfahren, für den die Fuzzy-Menge μ mit $\mu(k) = 0.0$, $\mu(m) = 0.6$, $\mu(g) = 1.0$ die möglichen Fahrzeugklassen repräsentiert, ist $\mu \circ \varrho$ zu berechnen. Die Berechnung erfolgt analog zur Matrixmultiplikation, vgl. Tabelle 8.3. Die Fuzzy-Menge μ wird als Zeilenvektor links unten und die Fuzzy-Relation ϱ in Matrixform oben in das Schema eingetragen. Die Fuzzy-Menge ν ergibt sich dann als Zeilenvektor unten rechts. Zur Berechnung des Wertes $\nu(140)$ wird der Zeilenvektor, der die Fuzzy-Menge μ repräsentiert, transponiert und das Minimum mit dem ersten Spaltenvektor komponentenweise gebildet. Das Maximum der drei so erhaltenen Werte ist der Zugehörigkeitsgrad des Elementes 140 zu der Fuzzy-Menge ν . Die Zugehörigkeitsgrade der anderen Geschwindigkeiten erhält man entsprechend. Es ergibt sich $\nu(140) = 0.0$, $\nu(160) = 0.5$, $\nu(180) = 0.6$, $\nu(200) = 0.8$, $\nu(220) = 1.0$. \square

Tabelle 8.3: Schema zur Berechnung von $\nu = \mu \circ \varrho$.

			1.0	0.5	0.1	0.0	0.0
			0.0	0.5	1.0	0.5	0.0
			0.0	0.0	0.4	0.8	1.0
0.0	0.6	1.0					
			0.0	0.5	0.6	0.8	1.0

Bei der Fuzzy-Regelung mit Relationalgleichungen ist eine Fuzzy-Relation ϱ zu bestimmen, die die Gleichung $\nu_r = \mu_r \circ \varrho$ für alle k Regeln R_r erfüllt. Hierfür werden zuerst die Eingangswerte x_1, \dots, x_n zu einem Eingangsvektor $x = (x_1, \dots, x_n)$ zusammengefaßt. Der Wertebereich von x ist der Produktraum $X = X_1 \times \dots \times X_n$. Dies führt zu Regeln der Form

if x is A then y is B

und damit zu einer Menge von Relationalgleichungen $\nu_B = \mu_A \circ \varrho$. A und B sind dabei linguistische Terme, denen Fuzzy-Mengen μ_A von X und ν_B von Y entsprechen. μ_A ist definiert als

$$\mu_A : X \rightarrow [0, 1],$$

$$(x_1, \dots, x_n) \mapsto \min\{\mu_{A_1}(x_1), \dots, \mu_{A_n}(x_n)\}.$$

Danach wird für jede der k Relationalgleichungen die Gödel-Relation bestimmt. Die *Gödel-Relation* $\varrho_{\mu, \nu}^{\text{Gödel}}$ ist definiert durch

$$(x, y) \in \varrho_{\mu, \nu}^{\text{Gödel}} \iff (x \in \mu \rightarrow y \in \nu).$$

Dabei wird die Implikation \rightarrow im Sinne der *Gödel-Implikation* ausgewertet, d.h.

$$\varrho_{\mu, \nu}^{\text{Gödel}} = \begin{cases} \nu(x), & \text{falls } \nu(x) < \mu(x) \\ 1, & \text{sonst.} \end{cases}$$

Falls eine Lösung für die Relationalgleichung existiert, ist auch die Gödel-Relation eine Lösung.

Aus den Lösungen für einzelne Relationalgleichungen wird die Lösung für das System von Relationalgleichungen $v_r = \mu_r \circ \varrho, r = 1, \dots, k$, bestimmt. Wenn dieses System lösbar ist, ist $\varrho = \min_{r \in \{1, \dots, k\}} \{\varrho_{\mu_r, v_r}^{\text{Gödel}}\}$ eine Lösung. Diese Lösung ist gleichzeitig auch die größte Lösung.

Ein System von Relationalgleichungen $v_i = \varrho \circ \mu_i, i = 1, \dots, r$, kann also gelöst werden, indem zunächst für jede einzelne Relationalgleichung die größte Lösung in Form der Gödel-Relation bestimmt und anschließend aus dem Minimum der Lösungen der einzelnen Gleichungen eine Gesamtlösung für das System ermittelt wird.

Bei dem Relationalgleichungsansatz erhält man über die Beziehung $\Pi\{(x, y)\} := \varrho(x, y)$ eine Einschätzung, ob es möglich ist, dass dem Eingabetupel x der Ausgabewert y zugeordnet wird. Man erhält also *Soft Constraints* für die Funktionsdefinition.

Literaturverzeichnis

- [1] H. Bandemer und S. Gottwald. *Einführung in Fuzzy-Methoden (4. Auflage)*. Akademie Verlag, Berlin 1993
- [2] B. Bilgen. Application of Fuzzy Mathematical Programming Approach to the Production Allocation and Distribution Supply Chain Network Problem. *Expert Systems with Applications* 37(6):4488–4495. Elsevier, Amsterdam, Niederlande 2010
- [3] C. Borgelt, M. Steinbrecher und R. Kruse. *Graphical Models – Representations for Learning, Reasoning an Data Mining*, 2nd edition. J. Wiley & Sons, Chichester, Großbritannien 2009
- [4] B.D. Boulay, D. Hogg und L. Steels, eds. *Advances in Artificial Intelligence 2*. North Holland, Amsterdam, Niederlande 1987
- [5] B.G. Buchanan und E.H. Shortliffe. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Reading, MA, USA 1984
- [6] R. Carnap. *Introduction to Symbolic Logic and Its Applications*. Dover, New York, NY, USA 1958
- [7] E. Castillo, J.M. Gutierrez und A.S. Hadi. *Expert Systems and Probabilistic Network Models*. Springer, New York, NY, USA 1997
- [8] C.B. Chiou, C.H. Chiou, C.M. Chu und S.L. Lin. The Application of Fuzzy Control on Energy Saving for Multi-unit Room Air-conditioners. *Applied Thermal Engineering* 29(2–3):310–316. Elsevier, Amsterdam, Niederlande 2009
- [9] C.K. Chow und C.N. Liu. Approximating Discrete Probability Distributions with Dependence Trees. *IEEE Trans. on Information Theory* 14(3):462–467. IEEE Press, Piscataway, NJ, USA 1968
- [10] G.F. Cooper und E. Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning* 9:309–347. Kluwer, Dordrecht, Niederlande 1992
- [11] C.J. Date. *An Introduction to Database Systems, Vol. 1*. Addison Wesley, Reading, MA, USA 1986

- [12] A. Dawid. Conditional Independence in Statistical Theory. *SIAM Journal on Computing* 41:1–31. Society of Industrial and Applied Mathematics, Philadelphia, PA, USA 1979
- [13] D. Dubois und H. Prade. *Possibility Theory*. Plenum Press, New York, NY, USA 1988
- [14] D. Geiger, T.S. Verma und J. Pearl. Identifying Independence in Bayesian Networks. *Networks* 20:507–534. J. Wiley & Sons, Chichester, Großbritannien 1990
- [15] J.M. Hammersley und P.E. Clifford. *Markov Fields on Finite Graphs and Lattices*. Unveröffentlichtes Manuskript, 1971. Nach: [21]
- [16] R.V.L. Hartley. Transmission of Information. *The Bell Systems Technical Journal* 7:535–563. Bell Laboratories, USA 1928
- [17] D.E. Heckerman. Probabilistic Interpretations for MYCIN's Certainty Factors. In [24].
- [18] D. Heckerman, D. Geiger und D.M. Chickering. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning* 20:197–243. Kluwer, Dordrecht, Niederlande 1995
- [19] H. Hermes. *Einführung in die mathematische Logik*. Teubner, Stuttgart 1976
- [20] D. Hofbauer und R.-D. Kutsche. *Grundlagen des maschinellen Beweises*. Vieweg, Braunschweig/Wiesbaden 1989
- [21] V. Isham. An Introduction to Spatial Point Processes and Markov Random Fields. *Int. Statistical Review* 49:21–43. Int. Statistical Institute, Voorburg, Niederlande 1981
- [22] F.V. Jensen Junction Trees and Decomposable Hypergraphs. Research Report, Aalborg University, Dänemark 1988
- [23] F.V. Jensen. *An Introduction to Bayesian Networks*. UCL Press, London, Großbritannien 1996
- [24] L.N. Kanal und J.F. Lemmer. *Uncertainty in Artificial Intelligence*. North-Holland, Amsterdam, Niederlande 1986
- [25] F. Klawonn, J. Gebhardt und R. Kruse. Fuzzy control on the basis of equality relations with an example from idle speed control. *IEEE Transactions on Fuzzy Systems*, 3:336–350, 1995
- [26] E.-P. Klement, R. Mesiar und E. Pap. *Triangular Norms*. Kluwer, Dordrecht, Niederlande 2000
- [27] R. Kruse, J. Gebhardt und F. Klawonn. *Fuzzy Systeme (2. Auflage)*. Teubner, Stuttgart 1995
- [28] R. Kruse, C. Borgelt, F. Klawonn, C. Moewes, G. Ruß, und M. Steinbrecher. *Computational Intelligence: Eine methodische Einführung in Künstliche Neuronale Netze, Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze*. Vieweg+Teubner, Wiesbaden 1995
- [29] J.B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proc. American Mathematical Society* 7(1):48–50. American Mathematical Society, Providence, RI, USA 1956
- [30] S. Kullback und R.A. Leibler. On Information and Sufficiency. *Annals of Mathematical Statistics* 22:79–86. Institute of Mathematical Statistics, Hayward, CA, USA 1951
- [31] M.I. Jordan, ed. *Learning in Graphical Models*. MIT Press, Cambridge, MA, USA 1998
- [32] G.J. Klir und B. Yuan. *Fuzzy Sets and Fuzzy Logic*. Prentice Hall, Englewood Cliffs, NJ, USA 1995
- [33] S.L. Lauritzen und D.J. Spiegelhalter. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society, Series B*, 2(50):157–224. Blackwell, Oxford, Großbritannien 1988

- [34] S.L. Lauritzen, A.P. Dawid, B.N. Larsen und H.G. Leimer. Independence Properties of Directed Markov Fields. *Networks* 20:491–505. J. Wiley & Sons, Sussex, Großbritannien 1990
- [35] S.L. Lauritzen. *Graphical Models*. Oxford University Press, Oxford, Großbritannien 1996
- [36] D. Maier. *The Theory of Relational Databases*. Computer Science Press, Rockville, MD, USA 1983
- [37] E.H. Mamdami und S. Assilian. *An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller*. Int. J. of Man Machine Studies, 7:1–13. Academic Press, London, Großbritannien 1975
- [38] W.J. Van Melle. *System Aids in Constructing Consultation Programs*. UMI Research Press, Ann Arbor, MI, USA 1980
- [39] W.J. Van Melle, E.H. Shortliffe und B.G. Buchanan. EMYCIN: A Knowledge Engineer's Tool for Constructing Rule-Based Expert Systems. In [5].
- [40] R. von Mises. *Wahrscheinlichkeit, Statistik und Wahrheit*. Berlin 1928
- [41] E.I. Papageorgiou, A. Markinos und T. Gempts. Application of Fuzzy Cognitive Maps for Cotton Yield Management in Precision Farming. *Expert Systems with Applications* 36(10):12399–12413. Elsevier, Amsterdam, Niederlande 2009
- [42] J. Pearl und A. Paz. Graphoids: A Graph Based Logic for Reasoning about Relevance Relations. In: [4], 357–363
- [43] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, USA 1988 (2nd edition 1992)
- [44] R.C. Prim. Shortest Connection Networks and Some Generalizations. *The Bell System Technical Journal* 36:1389–1401. Bell Laboratories, USA 1957
- [45] H. Reichenbach. *Elements of Symbolic Logic*. Macmillan, New York, NY, USA 1947
- [46] W.C. Salmon. *Logic*. Prentice Hall, Englewood Cliffs, NJ, USA 1963
- [47] L.J. Savage. *The Foundations of Statistics*. J. Wiley & Sons, New York, N.Y., USA 1954. Reprinted by Dover Publications, New York, NY, USA 1972
- [48] U. Schöning. *Logik für Informatiker*. BI Wissenschaftsverlag, Mannheim 1989
- [49] R.D. Shachter, T.S. Levitt, L.N. Kanal und J.F. Lemmer, eds. *Uncertainty in Artificial Intelligence 4*. North Holland, Amsterdam, Niederlande 1990
- [50] C.E. Shannon. The Mathematical Theory of Communication. *The Bell Systems Technical Journal* 27:379–423. Bell Laboratories, USA 1948
- [51] E.H. Shortliffe und B.G. Buchanan. A Model for Inexact Reasoning in Medicine. *Mathematical Biosciences* 23:351–379, 1975
- [52] E.H. Shortliffe. *Computer Based Medical Consultations: MYCIN*. Elsevier, New York, NY, USA 1976
- [53] M. Studený. Conditional Independence Relations have no Finite Complete Characterization. *Trans. 11th Prague Conf. on Information Theory, Statistical Decision Functions, and Random Processes*, 377–396. Academia, Prague, Czechoslovakia 1992
- [54] J.D. Ullman. *Principles of Database and Knowledge-Base Systems, Vol. 1 & 2*. Computer Science Press, Rockville, MD, USA 1988
- [55] T.S. Verma und J. Pearl. Causal Networks: Semantics and Expressiveness. In: [49], 69–76
- [56] C.F. Weizsäcker. *Zeit und Wissen*. Hanser, München 1992
- [57] J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. J. Wiley & Sons, Chichester, Großbritannien 1990

- [58] L. Wittgenstein. *Philosophische Untersuchungen*. Suhrkamp, Frankfurt am Main 1984
(Erstveröffentlichung: Blackwell, Oxford, England 1953)
- [59] L.A. Zadeh. Fuzzy Sets. *Information and Control* 8:338–353, 1965. Academic Press, San Diego, CA, USA 1965
- [60] L.A. Zadeh. The Concept of a Linguistic Variable and Its Application to Approximate Reasoning. *Information Sciences* 9:43–80. 1975

9 Fallbasiertes Schließen

Michael M. Richter

9.1 Motivation und etwas Historie

Der Name **Fallbasiertes Schließen** (FBS, engl. **Case-Based Reasoning, CBR**) ist für sich genommen wenig informativ und bedarf einer Erklärung. Ein **Fall** ist eine irgendwie geartete Erfahrung beim Lösen eines Problems. Die Beschreibung einer Erfahrung ist oft erst einmal informal gegeben und sie betrifft gewöhnlich sowohl die Problemstellung wie die erarbeitete Lösung. Darüber hinaus werden aber oft auch Erklärungen, wichtige Teile des Lösungsweges und andere Informationen als nützlich erkannt und mitgeliefert. Die Nutzung einer solchen Erfahrung, oder eines Falles, wie wir jetzt sagen werden, besteht darin, die Lösung auf ein neues, aktuelles aber hinreichend ähnliches Problem anzuwenden.

Die nächste Erklärung muss sagen, was hier **Schließen** bedeutet. Schlussweisen zur Lösung von Problemen gibt es im täglichen Leben viele. In wissensbasierten Systemen reduzieren sie sich allerdings oft auf logikorientierte, deren Hauptcharakteristikum ihre logische Korrektheit ist. Das ist bei CBR jedoch anders. Grundsätzlich verwendet man im Leben nicht Schlussweisen im Sinne der formalen Logik. Die aktuelle Situation ist ja nicht genau dieselbe wie in der alten Erfahrung und deshalb kann man bei ihrer Verwendung auch daneben liegen. Die CBR Schlussweisen haben zum Ziel, sich diejenigen Erfahrungen auszusuchen, die für die Problemlösung am hilfreichsten sind.

Eine Menge solcher Erfahrungen ist eine Fallbasis. Sind die beiden Probleme (Erfahrung – aktuelles Problem) aus unterschiedlichen Domänen (z.B. Sonnensystem und Atommodell) spricht man von Analogie; fallbasiertes Vorgehen wird heute so verstanden, dass beide Probleme aus demselben Bereich stammen. Analogie und FBS haben aber viel Gemeinsames.

Diese Methodik, Probleme zu lösen, ist dem Menschen sehr vertraut, ihre Nutzbarmachung für die Unterstützung automatischen Problemlösens geht auf den Ansatz des Dynamischen Gedächtnisses [20, 21] zurück. Hier wird ein psychologisches Modell über menschliches Problemlösen und Lernen verbunden mit Vorstellungen, menschliche Intelligenz auf einem Rechner zu verwirklichen. Es werden dabei einige grundlegende Annahmen gemacht:

1. Erinnern und Anpassen (Adaptieren) sind zentrale mentale Prozesse beim Verstehen.
2. Indexierung ist für das Erinnern wichtig.
3. Verstehen führt zur Reorganisation des Gedächtnisses, weshalb dieses dynamisch ist.
4. Die Gedächtnisstrukturen für die Wissensverarbeitung sind dieselben wie die für die Wissensspeicherung

Parallel gibt es vielfältige Relationen von CBR zu anderen Entwicklungen. In Abbildung 9.1 sehen wir einen Überblick.

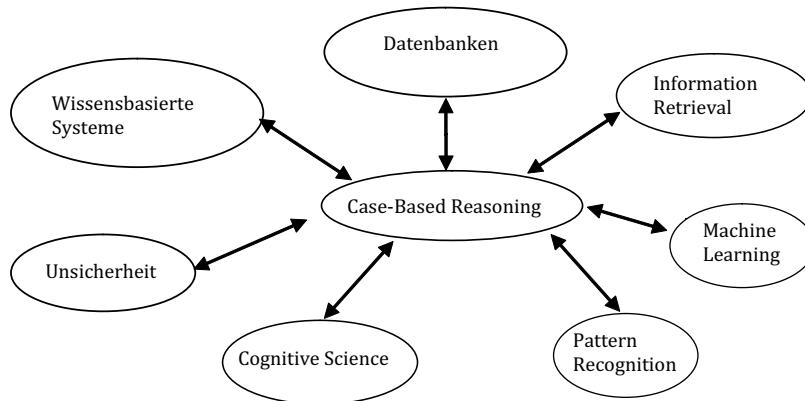


Abbildung 9.1: Diagramm 1.

Dabei unterscheiden wir drei Arten von Beziehungen:

1. Gebiete, die auf die Entwicklung von CBR einen Einfluss gehabt haben.
2. Gebiete, die teilweise in Konkurrenz zu CBR stehen.
3. Gebiete, die sich mit CBR gut ergänzen können.

Hier kann man natürlich keine genauen Abgrenzungen angeben. Zur ersten Kategorie zählen:

- Cognitive Science,
- Maschinelles Lernen,
- Unsicherheit, Fuzzy Systeme,
- Datenbanken.

In die zweite Kategorie fallen:

- Datenbanken,
- Information Retrieval,
- Pattern recognition

In der dritten Kategorie finden wir:

- Maschinelles Lernen,
- Pattern recognition,
- Wissensbasierte Systeme.

9.2 Einige Charakteristika von CBR

Es fällt zunächst auf, dass wir es nicht mit klassischen Berechnungen im Sinne der Rekursionstheorie zu tun haben. Anstelle von gleich-ungleich oder richtig-falsch haben wir es hier mit ungefähr gleich oder ziemlich richtig zu tun. Da erhebt sich sofort die Frage, ob wir es hier mit einer völligen Beliebigkeit beschäftigen. Anders ausgedrückt: Ist dies eine seriöse Wissenschaft?

Nun gibt es in der Mathematik wie in der Informatik viele Disziplinen, in denen nicht eine exakte sondern eine approximative Lösung im Mittelpunkt steht, denken wir nur an die Approximationstheorie, die Nutzentheorie oder die Optimierung. Der Nutzengesichtspunkt ist zentral für CBR, denn wir verwenden Erfahrungen, weil wir denken, dass sie von Nutzen sind. Der Verzicht auf Exaktheit spiegelt sich nun darin wider, dass nicht mehr der Wahrheitsbegriff sondern der Nützlichkeitsbegriff im Vordergrund steht: Lösungen sind nicht mehr einfach wahr, sondern mehr oder weniger nützlich und werden mehr oder weniger präferiert.

Es handelt sich dann nicht mehr um eine prinzipiell logische sondern um eine Optimierungsfrage, die im multikriteriellen Falle nach einem Paretooptimum sucht (also eines, bei dem die Verbesserung eines Kriteriums notwendigerweise zur Verschlechterung mindestens eines anderen führt). Während der Wahrheitsbegriff universell ist, wird der Nutzen individuell festgelegt; der Wahrheitsbegriff stellt aber einen extremen Fall des Nutzens dar, wenn man eben nur an der Wahrheit interessiert ist. Das „mehr oder weniger“ in den Mittelpunkt zu stellen ist allerdings eine allgemeine Tendenz und das prominenteste Beispiel hierfür ist der (sehr erfolgreiche) Fuzzy-Ansatz; in der Tat bestehen viele Relationen zwischen Fuzzy-Ansatz und FBS, vgl. [6]. Der Verzicht auf Exaktheit bereitet aber nicht nur (von einem formalen Standpunkt) Schwierigkeiten, sondern er eröffnet auch bei interaktivem Vorgehen neue Perspektiven. Aber diese Vorgehensweise bedarf einer genauen Arbeitsteilung zwischen Mensch und Maschine.

Die Verwendung einer so erhaltenen Lösung kann einmal so geschehen, dass sie unverändert übernommen, und zum andern so, dass sie geeignet adaptiert wird.

9.3 Grundbegriffe und ein einfaches Modell

9.3.1 Fälle als Erfahrungen

Als nächstes führen wir die wichtigsten elementaren Begriffe ein und kommentieren sie kurz. Ausgangspunkt sind zwei Mengen P (Problemmenge) und L (Lösungsmenge).

Definition 9.3.1.

1. *Ein Fall ist ein geordnetes Paar $F = (P, L)$ mit $P \in P$ und $L \in L$.*
2. *Eine Anfrage ist ein Element $P \in P$. Durch Verwendung eines speziellen Symbols „?“ lässt sie sich auch als Fall $(P,?)$ darstellen.*

Notiert man sich einen Fall F in der Form $P \rightarrow L$, dann sieht er wie eine Regel aus. Das ist auch ganz richtig, nur handelt es sich um eine Regel ohne Variable und deshalb ohne Instanzen. Als Regel könnte man den Fall daher stets nur auf ein und dasselbe Problem P anwenden.

Die Struktur von L kann recht beliebig sein, sie beschreibt die damalige Erfahrung.

Definition 9.3.2. Eine Fallbasis ist ein geordnetes Paar (FB, S) , wobei FB eine Menge von Fällen ist und S eine Struktur über FB ist. Diese Struktur organisiert die Fälle, ganz wesentlich um ihre Wiederverwendung zu erleichtern.

Es gibt mehrere Gründe, die Definition eines Falles zu erweitern:

1. Hinzunahme eines Eintrags K , genannt Kommentar, also $F = (P, L, K)$, wobei ein Kommentar inhaltlich eine Art von Erklärung sein soll, die z.B. Lösungswege und Kontexte beschreiben oder Güteinformationen beinhalten kann.
2. Hinzunahme eines Eintrags E genannt Effekt, also $F = (P, L, E)$. Ein Effekt beschreibt was tatsächlich bei der Anwendung der Erfahrung passiert ist. Seine Bedeutung ist interessant für Lösungen, die Handlungsanweisungen sind, deren Resultate bei der Handlungsausführung nicht sicher vorhersehbar sind.

In jeder konkreten Situation werden Problem- und Lösungsteil in gewissen Datenstrukturen beschrieben sein. Dabei sind oft beide Teile Instanzen einer allgemeinen Beschreibung, in denen gewisse Variable mit Werten belegt sind.

Beispiele:

1. Diagnostik (inklusive Therapie): Der Problemteil ist durch Werte für messbare Größen definiert, etwa Fieber = 39, Blutdruck = 180 etc., während der Lösungsteil Einträge der Art Krankheit = K , Bettruhe = 3 Tage, Medikament = Antibiotikum A etc. enthält.
2. Architektur: Ein Haus kann durch z.B. durch Werte für Zimmerzahl, Anzahl der Stockwerke, Art des Daches, Heizungsart beschrieben werden. Einige dieser Werte werden vom Kunden vorgegeben, die anderen müssen gefunden werden.

Dies führt auf folgende Unterscheidung:

- Eine Klasse von Problemen ist vom Regeltyp, wenn es eine Partition der Variablen in zwei Mengen Problemvariable und Lösungsvariable gibt, die entsprechend im Problem- und im Lösungsteil auftreten.
- Eine Klasse von Problemen ist vom Constrainttyp, wenn es keine solche Separierung der Variablen gibt.

Falls man nun einen Fall, der einem mit einer gewissen Ähnlichkeit vorgelegt, dann fragt man sich sofort, ob man dessen Lösung akzeptiert. Das kann man von dem Grad der Ähnlichkeit abhängig machen. Nun ist Akzeptanz keine graduelle sondern eine ja-nein Entscheidung. Wenn hinreichend sicher für eine Annahme ist, sagt man ja und wenn für eine große Ablehnung stimmt wird nein gesagt. In Abbildung 9.2 sehen wir das mit Hilfe von Konstanten α und β so:

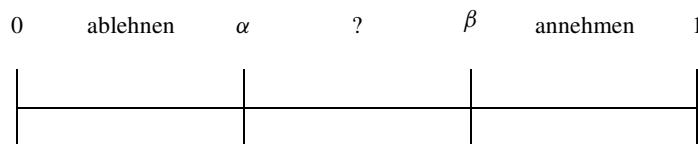


Abbildung 9.2: Unsicherheit.

In der Terminologie der fuzzy Mengen handelt es sich um eine „defuzzification“. Wie diese im Einzelnen zu geschehen hat, hängt von der Situation ab.

Wenn nun eine Fallbasis vorliegt, wie verwenden wir die Erfahrungen zur Lösung neuer Probleme? Wir suchen uns aus den Erfahrungen diejenige aus, die uns am besten erscheint. Dies geschieht mit Hilfe eines geeigneten Ähnlichkeitsbegriffes, der weiter unten detailliert behandelt wird. Der Fall, dessen Problemteil zu unserem Problem am ähnlichsten ist (der sog. nächste Nachbar) wird selektiert. Da dessen Lösung aber evtl. immer noch nicht richtig passt, wird diese dann noch angepasst (adaptiert). Die CBR Prozedur ist in Diagramm 9.3 dargestellt.

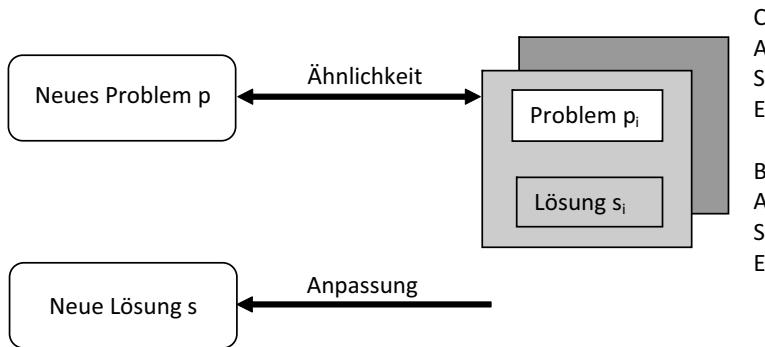


Abbildung 9.3: CBR Prozedur.

9.3.2 CBR Methodologie

CBR lässt sich von anderen Ansätzen die Erfahrungen verwenden durch eine klare Methodologie abgrenzen. Diese beruht auf zwei Pfeilern:

1. Ein Prozessmodell
2. Ein Wissensmodell

Beide Grundpfeiler werden zuerst skizziert und weiter unten genauer beschrieben.

9.3.3 Das Prozessmodell

Das grundlegende Prozessmodell wird durch ein Phasenmodell, den sog. CBR-Cycle [1] beschrieben. Das Modell enthält die wesentlichen für CBR relevanten Schritte, ist aber in vieler Hinsicht erweitert worden.

Zunächst sollen die einzelnen Schritte im *Prozessmodell* beschrieben werden.

Problemformulierung

Der erste Schritt ist die Problemformulierung. Die Formulierung eines Problems kann auf verschiedene Weisen erfolgen. Man hat sie jedoch so zu machen, dass ein vorgelegtes CBR System sie auch akzeptiert. Das ist z.B. problematisch zu verarbeiten, wenn das Problem in natürlicher Sprache oder in der Form von Bildern oder gemessenen technischen vorgelegt wird. Es bedarf

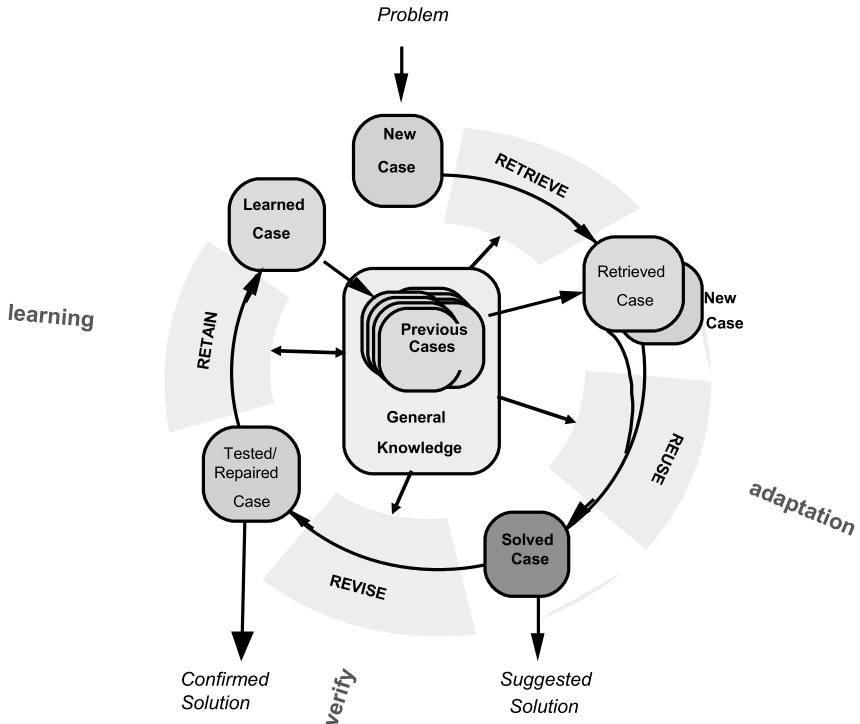


Abbildung 9.4: Das Prozessmodell.

dann oft recht aufwendiger Methoden, diese Darstellungen in solche zu transformieren, die eine Maschine direkt verarbeiten kann. Eine weitere häufig auftretende Schwierigkeit ist, dass der Benutzer sein Problem zuerst einmal gar nicht richtig formulieren kann. Zu diesem Zwecke kann man eine Unterhaltung mit dem Anwender einschalten, z.B. ein Verkaufsgespräch mit dem Endergebnis dass Problemwunsch dann klar ist. Hierzu sind Techniken unter dem Stichwort „Conversational CBR“ entwickelt worden.

Retrievalphase

Nach Präsentation eines neuen Problems wird dann ein geeigneter Fall selektiert. Dies ist schwieriger als bei einem Datenbankabruf.

Die Strukturen der Fallbasis haben im Wesentlichen eine Indexierung der Fälle für ein effizientes Retrieval zum Ziel. Die einfachste Struktur ist die Sequenz und der zugehörige sequentielle Retrievalalgorithmus ist offensichtlich und einfach zu implementieren. Seine Komplexität ist $O(n)$, was bei einer großen Fallbasis schon problematisch ist. Weiter ist der Retrievalaufwand unabhängig von der Anfrage.

Die Wiederverwendungsphase

Der durch Retrieval erhaltene Fall wird dann wiederverwendet. Wenn der Fall ein Paar (Problem, Lösung) ist, nimmt man das zweite Element, andernfalls die Lösung selbst. Damit ist aber noch nicht garantiert, dass sie auch zufriedenstellend ist. Aus diesem Grunde erfolgt eine An-

passung, also eine Lösungsadaption. Diese Adaption geschieht mittels geeigneter Regeln. Wir beschreiben das näher in Abschnitt über die Wissenscontainer.

Revision

In der Revisionsphase wird die Lösung auf Brauchbarkeit oder Korrektheit getestet und gegebenenfalls revidiert, wobei eine teilweise Überlappung mit der Reusephase bestehen kann. Das Testen geschieht nicht in einem Modell sondern in der Realität, allenfalls in einer Simulation.

Die eventuell erhaltene neue Lösung wird dann in der Retainphase wieder in das System eingebracht. In der einfachsten Form geschieht das durch Speicherung eines Falles. Die neue Erfahrung kann aber auch einem Lernprozess übergeben werden, wodurch auch das Ähnlichkeitsmaß oder die Adoptionsregeln verbessert werden können. Der CBR-Cycle stellt nur die Basisaktivitäten dar, es fehlt z.B. das Vergessen von unnötig gewordenen Fällen und überhaupt jede Art von Wartung.

9.3.4 Die Wissenscontainer und ihre Diskussion

Wissenscontainer sind strukturelle Elemente, die sich von traditionellen Modulen in Programmen wesentlich unterscheiden. Ein Modul löst eine bestimmte Teilaufgabe und damit strukturieren Moduln den Lösungsansatz. Wissenscontainer lösen für sich gar keine Aufgabe, auch an sehr einfachen Aufgaben sind stets mehrere Container beteiligt. Ein solcher Container enthält vielmehr Wissen, das für viele Aufgaben relevant ist. Wissenscontainer in regelbasierten Systemen sind z.B. Fakten und Regeln. In fallbasierten Systemen lassen sich vier Container identifizieren:

1. Die Repräsentationssprache und das Vokabular
2. Das Ähnlichkeitsmaß
3. Die Fallbasis
4. Die Lösungstransformation

Im Prinzip kann jeder Container (fast) alles Wissen beinhalten. Wenn man im Vokabular ein Attribut *Lösung* hat, so ist damit schon alles getan. Ist jeder denkbare Fall in der Fallbasis, so enthält diese alles nötige Wissen. Die Lösungstransformation könnte ein allgemeiner Problemlöser sein, ohne sich um die Fälle zu kümmern, und das Vokabular könnte ein auswertbares Attribut „Lösung“ enthalten. Schließlich kann auch das Ähnlichkeitsmaß im Wesentlichen über die Lösung Bescheid wissen. Hat man z.B. eine Klassifikationsaufgabe mit n Klassen und aus jeder Klasse ein Objekt in der Fallbasis, dann würde das Maß „ $\text{sim}(a, b) = 1$ “ genau dann, wenn a und b aus derselben Klasse sind“ sofort die Klassifikationsaufgabe lösen. Gewöhnlich ist das Wissen aber über alle Container verteilt, wobei die Verteilung jedoch nicht zeitlich invariant sein muss. Weil das Wissen für den CBR Prozess gebraucht wird, sind die Container eng mit dem Prozessmodell verknüpft. Zunächst werden die einzelnen Container näher diskutiert.

9.4 Eine Erweiterung

Die bisherige Vorgehensweise kann genauso anwenden, wenn man z.B. ein Auto kaufen will, aber keine Erfahrungen vorliegen. Man beschreibt einfach sein Wunschauto, und der Verkäufer vergleicht dies mit den Beschreibungen der vorhandenen Autos und sucht das Beste aus. Dies nutzen wir für eine generelle Erweiterung aus:

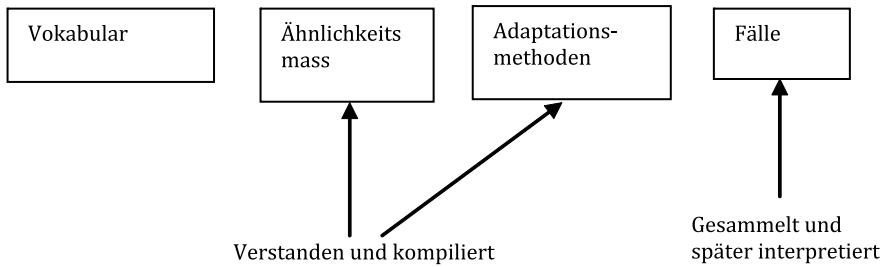


Abbildung 9.5: Wissenscontainer

- Wir ersetzen die Fallbasis durch eine Menge von Lösungen. Diese nennen wir Produktbasis.
- Wir vergleichen jetzt Probleme (Fragen) mit Lösungen (Produkte) wieder mit Hilfe eines Ähnlichkeitsmaßes.
- Anschließend wird dann das selektierte Produkt evtl. noch adaptiert.

Das eröffnet grundsätzlich viel erweiterte Anwendungsmöglichkeiten. In Abbildung 9.6 stellt sich das so da:

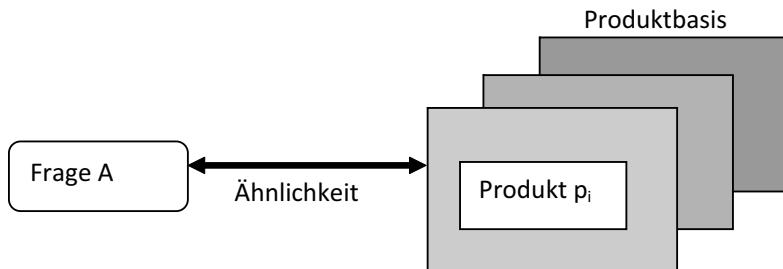


Abbildung 9.6: Erweiterte Sicht.

Einige Beispiele sind in 9.1 dargestellt: Weiter unten werden wir die Einzeltechniken für den Fall der Erfahrungen und die erweiterte Sicht erörtern und werden feststellen, dass sie sich nicht wesentlich unterscheiden. Ein wesentliches Problem ist aber, dass der Begriff der Ähnlichkeit bei der Erweiterung oft schwerer zu fassen ist.

Tabelle 9.1: Allgemeine Fragen-Lösungen.

Fragen	Lösungen
Gewünschte Produkte	Vorhandene Produkte
Symptome	Therapie
Bilder	Bedeutung
Texte	Interpretation
Benötigtes Wissen	Dokumente
Funktionalitäten	Maschinen
Benötigte Mitarbeiter	Mitarbeiter, Partner
Fragen	Antworten

9.5 Repräsentationssprachen

Die Formulierung von Problemen und Lösungen haben in einer Repräsentationssprache zu erfolgen. Für das Problem ist das eine gravierende Einschränkung, denn es muss vom CBR System verstanden werden.

Die **Sprache zur Repräsentation** von Fällen ist weitgehend beliebig, ihre Wahl ist von pragmatischen Gesichtspunkten diktiert. Grundsätzlich ist es wichtig, zwischen syntaktischen und semantischen Gesichtspunkten zu unterscheiden. Syntaktisch sind wenige Unterschiede zu allgemeinen, wissensbasierten Systemen festzustellen, deshalb werden wir nur einige Möglichkeiten kurz benennen.

9.5.1 Attribut-Wert Darstellungen

Attribut-Wert-Darstellungen sind am häufigsten zu finden. Hier werden Fälle (sowohl Problem- wie Lösungsseite) durch Attribut-Wert-Paare dargestellt.

Die Menge der Attribute ist entweder für alle Fälle fest vorgegeben oder sie kann von Fall zu Fall variieren. Letzteres ist in der Diagnostik häufig zu finden, weil meist nur ein kleiner Teil der Beobachtungen interessant ist.

Jedem Attribut ist A ein Wertebereich (Typ) $T(A)$ zugeordnet. Neben den Standardtypen spielen benutzerdefinierte Typen eine wichtige Rolle, also gewisse Unterbereiche, Aufzähltypen oder gar Texte. Bei fest vorgegebener Attributmenge A_1, \dots, A_n mit Wertebereichen T_1, \dots, T_n ist ein Fall ein n -stelliger Vektor:

$$F = (a_1, \dots, a_n) \in T_1 \times \dots \times T_n.$$

Man kann zwischen zwei Arten von Attributen unterscheiden:

1. Primäre Attribute: Diese sind von außen vorgegeben.
2. Virtuelle Attribute: Diese sind in Termini von primären Attributen definiert.

In konkreten, z.B. diagnostischen Aufgabenstellungen sind viele Attribute unnötig und es ergibt sich die Aufgabe, die relevanten primären Attribute zu selektieren. Die Aufgabe der virtuellen Attribute ist es, die für die Problematik wichtigen Zusammenhänge zu repräsentieren.

So können z.B. primäre Attribute Körpergewicht und Körpergröße eines Menschen sein. Für die gesundheitliche Beurteilung der Person ist jedoch der sog. Body-Maß-Index BMI¹ von Bedeutung. Virtuelle Attribute wirken sich auch auf die Ähnlichkeitsberechnung aus. In unserem Beispiel sind zwei Personen in gesundheitlicher Hinsicht ähnlich, wenn ihre BMIs nur wenig differieren, ein Sachverhalt, der nur in Termini der einzelnen Attribute Größe und Gewicht allein schwer auszudrücken ist. Die Suche nach geeigneten virtuellen Attributen spielt eine wichtige Rolle beim Aufbau eines fallbasierten Systems.

Die Vorteile von Attribut-Wert-Darstellungen sind generell Einfachheit und Verständlichkeit sowie Effizienz der Ähnlichkeitsbestimmung und des Retrieval. Auch Unvollständigkeit und Informationsvervollständigung lassen sich adäquat modellieren.

¹ BMI = Größe/(Gewicht)².

Zu den Nachteilen gehören vor allem Mängel an Möglichkeiten zur Repräsentation struktureller und relationaler Information. Geeignet ist die Darstellung vor allem für analytische Aufgaben, insbesondere für Klassifikation bei großen Fallbasen mit wenigen Merkmalen; weniger geeignet ist sie für synthetische Aufgaben wie Planung.

Eine Schwierigkeit ist noch die Repräsentation von unbekannten Attributwerten. Sie kann durch spezielle Typvariable oder ein generelles Symbol „unbekannt“ geschehen.

Dabei ist es ein Unterschied, ob ein Attributwert in einer Anfrage oder im Problemteil eines Falles fehlt. Geht man davon aus, dass die Lösung des Falles nicht durch einen Ratevorgang zustande kam und der Fall zudem korrekt protokolliert wurde, dann kann man sagen, dass der fehlende Wert auch unnötig war. Kommt dieser Wert dann in einer Anfrage vor, bezeichnet man ihn als redundant. Fehlt ein Wert aber in einer Anfrage und kommt in einem sonst vergleichbaren Fall vor, dann bezeichnet man ihn als fehlend. Fehlende Werte kann man auf verschiedene Weisen behandeln, wenn sie nicht erhoben werden können. So kann man etwa einen Defaultwert einsetzen oder denjenigen Wert, der die größte Wahrscheinlichkeit unter den Bedingungen der bereits bekannten Werte hat.

9.5.2 Weitere Darstellungen

Hierzu gehören praktisch alle weiteren Möglichkeiten der Wissensrepräsentation wie Prädikatenlogik, Frames, Graphen etc. Hier ist aber zu beachten, dass nicht nur der Wunsch nach Ausdrucksstärke Pate stehen darf, sondern es müssen alle im CBR-Cycle anfallenden Aktivitäten berücksichtigt werden. Dazu gehören insbesondere die Ähnlichkeitsberechnungen und das Retrieval, die in dieser Form in anderen wissensbasierten Systemen nicht auftreten. Die meisten Darstellungen erlauben eine generelle **Konstruktiorannahme**, die als das lokal-global Prinzip formuliert wird. Es besagt:

Jedes Objekt oder Konzept wird (global) beschrieben mittels eines Konstruktionsoperators aus lokalen Elementen:

$$A = C(A_i | i \in I).$$

Dabei ist I eine Indexmenge und die A_i sind die kleinsten lokalen Elemente, genannt Atome.

Man redet von attributbasierten Darstellungen, wenn die Atome Attribute sind. Alle Daten sind dann Werte von Attributen A_1, \dots, A_n mit nicht notwendig festem n , und es gibt einen Konstruktur C , so dass sich jedes Objekt O darstellen lässt als $C(A_1, \dots, A_n)$.

Es gibt jedoch Darstellungen, die nicht auf Attributen basieren wie Bilder, Texte oder gesprochene Sprache.

Die Konstruktoren sind aus der objektorientierten Programmierung bekannt. Man kann hier direkt die objektorientierten Beschreibungen aus einer Sprache wie Java übernehmen. Wir illustrieren dies am Beispiel von Anfragen und Produkten, deren Ähnlichkeit wir bestimmen wollen. Es sollen aber nicht nur komplexe Objekte verglichen werden, sondern wir haben die zusätzliche Schwierigkeit, dass ein und dasselbe Objekt in der Anfrage und der Produktbeschreibung ganz unterschiedlich dargestellt werden können. Das liegt daran, dass z.B. ein Kunde sich in

der Anfrage für die Funktionalität eines Objektes interessiert, während der Verkäufer nur die Produktbeschreibung anbieten. Dies haben wir in Abbildung 9.7 illustriert. Die allgemeine Objektbeschreibung hat dann zwei Teilobjekte. Wir modellieren damit dann auch eine Beziehung zwischen den Teilobjekten, hier die Relation „ein Objekt realisiert eine Funktionalität“.

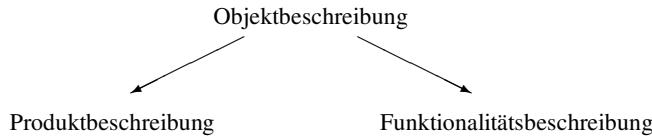


Abbildung 9.7: Zwei Sichten

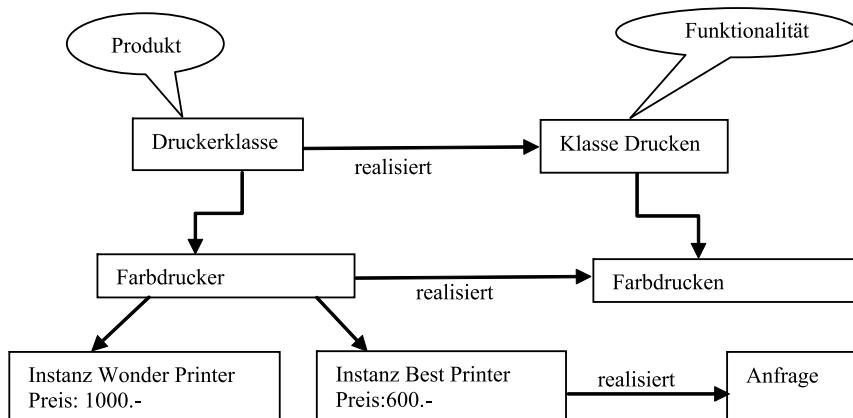


Abbildung 9.8: Realisierung.

Ein Beispiel: In einer Verkaufssituation wird oft die Funktionalität als Problemwunsch artikuliert. Der Verkäufer benutzt zur Darstellung der Ähnlichkeitsbeziehung dann die Beziehung „Produkt realisiert Funktionalität“. Da zwischen den beiden Beschreibungen oft eine erhebliche Lücke besteht, ist diese Ähnlichkeitsbeschreibung dann schwer zu erstellen.

Häufig sind Darstellungen mit annotierten Graphen, sowohl für die Frage wie auch das Problem. Abbildung 9.9 zeigt das Beispiel eines Problems, in dem nach kürzesten Verbindungen gefragt wird. Die Graphendarstellung ist sehr intuitiv.

9.6 Ähnlichkeiten

9.6.1 Generelles

Ähnlichkeiten sind ein zentrales Element approximativen Schließens. Für vieles Grundlegende vgl. [17].

Die Intention ist die Gleichheit zu verallgemeinern. Dies kann auf mannigfache Weisen geschehen. Die Haupteinflussfaktoren waren die Fuzzy Gleichheit und die Metriken der Mathematik.

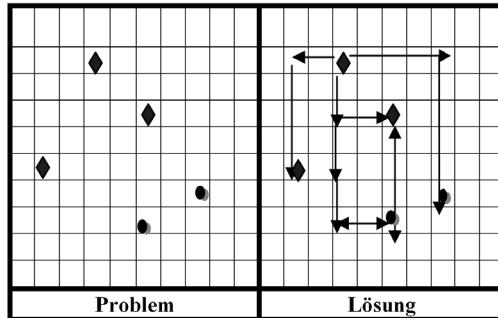


Abbildung 9.9: Graphendarstellung für Problem und Lösung.

Man kann mit einem gewissen Recht sagen, dass es seit dem Turmbau zu Babylon kaum einen Begriff gibt, der so vielen Wandlungen unterzogen ist, wie der der Ähnlichkeit. Nichtsdestoweniger genügt der Begriff gewissen Regelungen und Standards. Auf diese soll hier zunächst eingegangen werden.

Ähnlichkeiten sind für zwei Objekte aus einer Menge U von Objekten erklärt. Man unterscheidet *relationale* und *funktionale* Beschreibungen von Ähnlichkeit. Die simpelste relationale Form der Beschreibung benutzt ein zweistelliges Prädikat

$$\text{SIM}(x, y) \text{ für } x, y, \in U.$$

(SIM kommt vom Englischen similarity). Dies trägt jedoch dem approximativen Charakter des Ähnlichkeitsbegriffes keine Rechnung; ein solches Prädikat hat allenfalls nachträglich, etwa im Zusammenhang mit einer Schwellwertbetrachtung, Sinn. Ein besserer Ansatz benutzt eine vierstellige Basisrelation

$$R(x, y, u, v)$$

mit der Intention „ x und y sind mindestens so ähnlich wie u und v “. Die konkrete Interpretation von R erfolgt dann im Rahmen der intendierten Anwendung. Auf dieser Basis kann eine für den fallbasierten Ansatz zentrale Definition gegeben werden.

Definition 9.6.1.

$$NN(x, z) \Leftrightarrow (\forall y) R(x, z, x, y)$$

gelesen als: z ist ein nächster Nachbar von x

Dabei kann x natürlich auch mehrere nächste Nachbarn haben; ist dann eine Auswahl erwünscht, muss sie entweder willkürlich oder nach zusätzlichen Prinzipien erfolgen. Eine Erweiterung besteht darin, mehrere Objekte als etwa k nächste Nachbarn zu betrachten.

Ist x als Referenzobjekt gegeben, so werden durch die Relation $R(x, z, x, y)$ alle anderen Objekte partiell nach ihrer Ähnlichkeit zu x geordnet.

Eine Realisierung der Relation R kann durch reell wertige Funktionen (Maße) geschehen, wobei zwei mathematisch äquivalente Möglichkeiten bestehen, Ähnlichkeiten und Distanzen, wobei die letzteren Ausdrucksweisen in der Mathematik geläufiger sind. In einem prinzipiellen Sinne

sind beide gleichwertig, obwohl sie sich in praktischen Anwendungen in etwas verschiedene Richtungen entwickelt haben.

Quantitative Beschreibungen der Ähnlichkeit erfolgen durch Maße.

Definition 9.6.1. Ähnlichkeitsmaße sim und Distanzmaße d sind beides reell wertige Abbildungen

$$U \times U \rightarrow \mathbb{R}.$$

Normalerweise beschränkt man den Wertebereich von Ähnlichkeitsmaßen auf das reelle Intervall $[0, 1]$.

Tabelle 9.2: Mögliche Axiome.

	Fuzzy Gleichheit	Metrik
Reflexivität	$E(x, x) = 1$	$d(x, x) = 0$
Symmetrie	$E(x, y) = E(y, x)$	$d(x, y) = d(y, x)$
Transitivität	$E(x, y) \leq \sup\{z \min(E(x, z), E(z, y))\}$?
Dreiecksungleichung	?	$d(x, y) \leq d(x, z) + d(z, y)$
Substitutionsaxiom	$(s(a) = t(a) \wedge a = b) \Rightarrow (s(a) = t(b))$?

Hierzu gibt es zwei Vorgänger, Fuzzy-Gleichheiten und numerische Metriken, von denen die Maße gewisse natürliche Eigenschaften erben können. Einige wichtige sind in Tabelle 9.2 angegeben. Dabei bedeutet „?“, dass wir kein adäquates Analogon haben. Diese Axiome können jedoch in Frage gestellt werden, weil weitere Anwendungen den Anwendungsbereich über die naive Ähnlichkeit hinaus bedeutend erweitern. Ähnlichkeitsmaße sind also nichts anderes als Fuzzy Mengen auf Paaren. Im Gegensatz zur relationalen Darstellung erlaubt die funktionale Darstellung eine quantitative Sicht. Ein quantitatives Element ist in 9.10 illustriert.

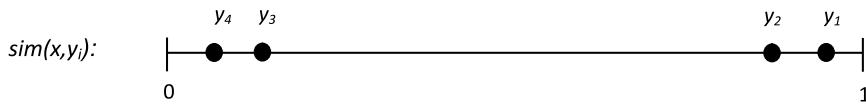


Abbildung 9.10: Ähnlichkeitsgrade.

Um den allgemeineren Fall mit zu erfassen, wird das Maß dadurch erweitert, dass man Objekte von zwei Mengen U und V in Beziehung setzt:

$$sim : U \times V \rightarrow [0, 1].$$

Der Wertebereich der Maße:

Er soll stets ein (beschränktes oder unbeschränktes) reelles Intervall I sein. Der einfachste und häufigste Fall ist $I = [0, 1]$ der zwei besondere Charakteristika hat:

1. Es gibt maximale und minimale Werte für Ähnlichkeiten und Abstände;
2. Es gibt keine negativen Werte.

Die Interpretation minimaler Abstände und maximaler Ähnlichkeiten ist klar, sie kennzeichnen Fälle, die für die Anfrage optimal sind. Die Interpretation maximaler Abstände und minimaler Ähnlichkeiten ist weniger klar. Es soll offensichtlich heißen, dass sie für die Selektion eines nächsten Nachbarn nicht in Frage kommen. Das lässt sich aber zweifach interpretieren. Es kann einmal heißen, dass Fälle mit maximalem Abstand zur Anfrage endgültig von der Betrachtung ausgeschlossen werden. Es kann aber in einem Diagnoseprozess, in dem sukzessive Werte erhoben werden, auch neutral gesehen werden, was sich durch weitere erhobene Werte ändern kann (vornehmlich bei gewichteten Hammingmaßen). Die Möglichkeit negativer Werte ist von besonderer Problematik. Einerseits wäre sie bei der Akkumulation verschiedener Ähnlichkeitsmaße manchmal durchaus wünschenswert, denn man könnte explizit ausdrücken, dass durch eine neue Beobachtung die Ähnlichkeit sinkt. Andererseits könnten bei Abständen negative Werte auftreten, die unerwünscht wären, weil dann nicht mehr jedes Objekt sein eigener nächster Nachbar wäre (setzt man $d(x, x) = 0$ voraus).

Die durch Maße und Abstände induzierten Relationen sind beschrieben durch:

Definition 9.6.2.

- (i) $R_{sim}(x, y, u, v) \Leftrightarrow sim(x, y) \geq sim(u, v)$
- (ii) $R_d(x, y, u, v) \Leftrightarrow d(x, y) \leq d(u, v)$
- (iii) *sim und d heißen relational kompatibel falls $R_{sim}(x, y, u, v) \Leftrightarrow R_d(x, y, u, v)$ für alle x, y, u, v gilt.*

Der letzte Begriff bezieht sich aber nur auf die Ordnungsstruktur und nicht auf die kardinalen Werte der Maße und insbesondere sagt er nichts darüber aus, wie *ähnlich oder unähnlich* zwei Objekte sind. Hat man hierüber eine Vorstellung kann man sich das oben erwähnte Prädikat SIM dadurch definieren, dass die Ähnlichkeit einen vorgegebenen Schwellwert überschreitet. Um Ähnlichkeitsmaße sim und Distanzen d quantitativ zu bewerten benötigt man Umrechnungsfunktionen. Hierzu nimmt man bijektive, ordnungsinvertierende Abbildungen:

$$\begin{aligned} f : [0, 1] &\rightarrow [0, 1] \text{ mit} \\ f(0) &= 1 \\ f(d(x, y)) &= sim(x, y) \end{aligned}$$

Unter diesen Bedingungen sind f und d kompatibel. Beispiele:

1. $f(x) = 1 - \frac{x}{max}$ wenn ein maximaler Abstand max existiert.
2. $f(x) = 1 - \frac{x}{1+x}$

9.6.2 Semantik der Ähnlichkeitsmaße

Wenn man sich mit dem Wissen in Ähnlichkeitsmaßen beschäftigt, möchte man gerne wissen, was man durch ihre Angabe mitgeteilt bekommt. Dazu betrachten wir als Beispiel folgende Fragen:

- F₁ Welche Information erhalte ich, wenn mir eine Person sagt „sim(a, b) = 0.83“ ?
 F₂ Was besagt mir „sim(a, b) > sim(b, c)“ ?
 F₃ Warum interessiere ich mich eigentlich für den nächsten Nachbarn mehr als für den dritt-nächsten?

Die Probleme liegen in zweierlei Aspekten begründet. Der eine ist, dass es keine klare Definition wie bei physikalischen Begriffen gibt. Der andere damit verbundene ist, dass ganz viele Leute sich darunter etwas Verschiedenes vorstellen. Mit anderen Worten, es fehlt uns eine Semantik. Wir gehen nach dem Vorbild der Prädikatenlogik vor und verwenden dazu Modelle. Diese Modelle kommen hier aus der Nutzentheorie.

Die einfachsten Modelle sind von der Form (U, \leq) wobei \leq transitive Halbordnung über einer Menge U ist. Solche Halbordnungen werden Präferenzrelationen genannt. Für $x \leq y$ sagt man, das man y vor x präferiert.

Diese relationale Fassung des Nutzens wird durch Nutzenfunktionen quantitativ erweitert. Eine Nutzenfunktion ist einfach eine reell wertige Funktion auf U . Sie induziert eine Präferenzrelation durch

$$x \leq y \Leftrightarrow f(x) \leq f(y).$$

Dies wird nun noch dafür verallgemeinert, dass das tatsächliche Resultat $\text{eff}(L)$ (der Effekt) einer Lösung L nur mit einer gewissen Wahrscheinlichkeit eintritt. Dann ist der erwartete Nutzen $u(L)$ definiert als

$$u(L) = \text{Prob}(\text{eff}(L)) \bullet u(\text{eff}(L)).$$

Die Ähnlichkeiten sollen nun den erwarteten Nutzen ausdrücken.

Eine Grundanforderung an das Ähnlichkeitsmaß ist aus der Sicht des Nutzens für ein vorgelegtes Problem P :

Wenn (P_1, L_1) und (P_2, L_2) zwei Fälle sind für die $\text{sim}(P, P_2) \leq \text{sim}(P, P_1)$ gilt, dann besteht keine Präferenz von L_2 über L_1 als Lösung für P .

Insbesondere wird nichts den Lösungen der nächsten Nachbarn vorgezogen. In diesem Sinn versetzt ein derartiges Maß das System dann in die Lage, die beste in der Fallbasis vorhandene Lösung zu finden und die Suche nach den nächsten Nachbarn ist ein Approximationsprozess.

Dies sind aber nach wie vor relationale Aspekte der Ähnlichkeit. Die Kardinalität betrifft Fragen der Form: Welche Information beinhalten Aussagen der Art „sim(x, y) = 0.85“ oder „sim(x, y) – sim(x, z) = 0.4“ ? Hierzu wollen wir uns auf Nutzenfunktionen zurückziehen, deren kardinale Informationen bereits etabliert sind.

Um Ähnlichkeiten und Nutzenfunktionen formal vergleichen zu können, definiert man für ein aktuelles Problem Q und Fälle $(P, L) \in FB$:

$$f_{\text{sim}, Q}(L) := \text{sim}(P, Q).$$

Diese Funktion hat dieselben Argumente und Werte wie die Nutzenfunktion $u_Q(L)$, die den Nutzen einer Lösung L für das Problem Q beschreibt. In einer idealen Situation würden diese

beiden Funktionen übereinstimmen. Das ist jedoch schon allein deswegen eine weitgehende Forderung, weil die konkrete Nutzenfunktion meist nicht genau bekannt ist.

Als Spezifikation für Ähnlichkeitsmaße nehmen wir nun Nutzenfunktionen u . Das führt zu drei-erlei Korrektheitsbegriffen:

Ein Maß sim ist bezüglich u

1. *Nächster Nachbar korrekt*, wenn für jedes Problem der nächste Nachbar die Lösung mit dem größten Nutzen liefert.
2. *Relational korrekt*, wenn die Präferenzrelation von $u_{\text{sim},q}(\cdot)$ gleich der Präferenzrelation von $u_q(\cdot)$ ist.
3. *Total korrekt*, wenn $u_{\text{sim},q}(a) = u_q(a)$ für alle q und a .

Falls zusätzlich die Retrievalmethode in dem Sinne korrekt ist, dass sie die Fälle in der Reihenfolge ihrer Ähnlichkeit selektiert, erhalten wir jetzt auch eine formale Korrektheitsdefinition des CBR Schließens: Die Spezifikation ist die Nützlichkeit, und ihre Erfüllung wird durch die Korrektheitsdefinitionen geliefert. Dies ist analog zu den entsprechenden Begriffen bei den Programmiersprachen.

Eine Abschwächung der Korrektheit fordert für Ähnlichkeit und Nutzen das gleiche Monotonieverhalten, das durch die nachfolgende Definition beschrieben wird.

Zwei Funktionen $f, g : X \rightarrow \mathbb{R}$ sind genau dann *ähnlich geordnet*, wenn

$$D(f, g, x, y) := (f(x) - f(y)) \times (g(x) - g(y)) \geq 0 \text{ für alle } x, y \in X$$

Gleiches Monotonieverhalten zweier Funktionen bedeutet, dass sie ähnlich geordnet sind. Liegt diese Eigenschaft vor, rechtfertigt sie auch das Prinzip des „nächsten Nachbarn“.

Damit werden jetzt auch die oben aufgeworfenen Fragen beantwortet:

F_1 $\text{sim}(a, b) = 0.85$ sagt, dass der Nutzen von b für ein Problem a 0.85 in Termini von Nutzeneinheiten ist.

F_2 $\text{sim}(a, b) > \text{sim}(a, c)$ sagt, dass der Nutzen von b größer ist als der von c für das Problem a .

F_3 Der nächste Nachbar ist der nützlichste.

Hierdurch wird auch erklärt, dass die Ähnlichkeit oft subjektiv ist, denn dies ist auch für den Nutzen der Fall. Der letztere hängt erstens vom Anfragenden und dazu noch von dessen spezieller Situation ab. Betrachten wir als Beispiel:

1. Ich bevorzuge Auto₁ vor Auto₂, weil es schneller fährt.
2. Ich bevorzuge Auto₂ vor Auto₁, weil es weniger Benzin verbraucht.

Dies zeigt, dass sehr verschiedene Präferenzrelationen existieren und entsprechend viele Ähnlichkeitsmaße. Um diese systematisch zu strukturieren, gehen wir genauso wie bei den Repräsentationen vor.

Um nun das Wissen in den Maßen näher zu betrachten, wenden wir dieselbe Strukturierung wie bei den Objektorientierungen an. Vgl. hierzu [18].

9.6.3 Das lokal-global Prinzip für Ähnlichkeitsmaße

Lokale Maße

Es werden die Maße also aus einzelnen lokalen Maßen sim_i mit Hilfe einer Konstruktorfunktion C zu einem globalen Maß sim zusammengefügt:

$$\text{sim} = C(\text{sim}_i | i \in I).$$

Als klassische Beispiele dienen uns die gewichteten Hammingmaße:

$$\sum (g_i \times \text{sim}_i(a_i, b_i) | 1 \leq i \leq n).$$

Die einzelnen sim_i beinhalten lokale Informationen über die Wertebereiche der sim_i . Dies sind oft die Wertebereiche von Attributen.

Hier betrachtet man verschiedene Probleme in derselben Domäne. Die Attribute und die lokalen Maße reflektieren (oder sollten es) die Eigenschaften der Domäne. Große Ähnlichkeit von Werten bedeutet dann intuitiv, dass ihr Unterschied von geringer Bedeutung ist. Das heißt aber keineswegs, dass das Maß uniform auf dem Wertebereich sein soll (also im reell wertigen Fall der euklidische Abstand), nur sollte es durch die Domäne motiviert sein. Hier besteht ein Zusammenhang zum qualitativen Schließen: Auch kleine Differenzen spielen eine große Rolle, wenn sie auf verschiedenen Seiten eines Landmarks liegen (wie kleine positive und negative Werte bei der Wassertemperatur). Es sei aber bemerkt, dass die Auswahl der Landmarks etwas von der Art der Probleme beeinflusst werden kann.

Globale Maße

Während also die Werte der lokalen Maße im Wesentlichen vom Problem unabhängig sein sollten, ist dies bei den Gewichten oder den Gewichtsfunktionen, die zum globalen Maß führen oft nicht der Fall. Die Gewichte reflektieren die Wichtigkeit des Attributes für das aktuelle Problem, was von Problem zu Problem verschieden sein kann. So hängt die Bedeutung einer Beobachtung in der Diagnostik oft von der speziellen Situation ab; im elektronischen Verkauf etwa von Reisebüros ist die Relevanz eines Attributs wie „Strandnähe“ sehr vom individuellen Kunden abhängig. In unserem Diagnosebeispiel wären die Gewichte auch sicher anders gewählt worden, wenn das Problem gewesen wäre, die Autos nach dem Verkaufspreis zu klassifizieren.

Damit haben wir eine Aufteilung: Die lokalen Aspekte sind primär durch die Domäne bestimmt, die globalen eher durch die individuelle Nutzenfunktion. Dieser Unterschied wird unwichtig, wenn alle Kunden die gleiche Nutzenfunktion haben.

Bei den gewichteten Hammingmaßen sollen die Gewichte die Relevanz der einzelnen Argumente (z.B. der Attribute) wiederspiegeln. Jedoch: Wie bestimmt man sie?

Zur Motivation de Vorgehensweise denken wir uns, dass jedes Attribut durch eine Person vertreten ist. Jede Person richtet sich bei ihrer Entscheidung danach, was ihr Attribut empfiehlt. Stellen wir uns nun vor, dass Willi sehr oft mit seinem Vorschlag richtig liegt. Würden wir dann wirklich alle Voten gleich behandeln? Wohl kaum, denn Willi bekäme einen größeren Einfluss. Das resultiert im Falle von Klassifikationsaufgaben in Folgendem.

Definition 9.6.3. Wenn das Attribut den Wert a hat und C die Klasse des betrachteten Objektes O ist, dann setzen wir:

- (i) $\text{PredictivePower}(A, a, O) = \text{Prob}(C | \text{value}(A) = a)$
- (ii) $\text{ExpPredPower}(A, O) = \text{Exp}(\text{PredictivePower}(A, a, O) \times \text{Prob}(a) | a \in \text{dom}(A))$
- (iii) $\text{PredPow}(A) = \text{Exp}(\text{PredPower}(A | O) \times \text{Prob}(O)).$

Dabei ist Exp der Erwartungswert.

Den Wert des Gewichtes $w(A)$ für das Attribut A setzen wir dann als

$$w(A) = \text{PredPow}(A).$$

Dies lässt sich in vielfältiger Weise verallgemeinern. Vgl. hierzu [15].

9.6.4 Spezielle Ähnlichkeitsmaße

Konkrete Maße sind auf Repräsentationen definiert. Es gibt zwei Urväter von vielen Maßen:

- Das Hammingmaß
- Das euklidische Maß.

Das Hammingmaß zählt für Attributvektoren einfach die Anzahl der Übereinstimmungen. Beide Maße sind rein syntaktisch und schnell zu berechnen, haben aber mit den Inhalten nichts zu tun.

Ein erster Ansatz für das Einbringen von Hintergrundwissen ist die Wichtigkeit der Attribute zu erfassen die mit einem Gewichtsvektor $g = (g_1, \dots, g_n), 0 \leq g_i \leq 1$ notiert werden. Meist wird noch die Gewichtssumme auf 1 normiert.

Weiteres Wissen besteht in der Einführung von lokalen Maßen für die einzelnen Attribute. Hier werden beliebige Wertebereiche der Attribute zugelassen. Auf jedem der Wertebereiche T_i sei nun ein Ähnlichkeitsmaß sim_i vorgegeben; diese Maße heißen lokale Maße.

Wir nehmen nun einen Maßvektor $\text{sim} = (\text{sim}_1, \dots, \text{sim}_n)$.

Das verallgemeinerte gewichtete Hammingmaß ist dann:

Definition 9.6.4. Das globale Hammingmaß mit dem Maßvektor sim und einem Gewichtsvektor g ist erklärt durch

$$H_{g, \text{sim}}((a_1, \dots, a_n), (b_1, \dots, b_n)) = \sum (g_i \times \text{sim}_i(a_i, b_i) | 1 \leq i \leq n).$$

Die lokalen Ähnlichkeiten stehen zwischen den Elementen der einzelnen Attribute.

Trotz ihrer Popularität sind gewichteten Hammingmaßen Grenzen gesetzt. Dazu betrachten wir das sog. XOR-Problem. Gegenstand sind zweistellige boolesche Attribute und eine Klasseneinteilung in die Klassen $K_1 = \{(0, 1), (1, 0)\}$ und $K_2 = \{(0, 0), (1, 1)\}$ liegt vor. Weiter enthalten die Fallbasis bereits drei der vier booleschen Vektoren. Man macht sich nun sofort klar, dass es kein gewichtetes Hammingmaß mit Gewichten $0 < g_1, g_2 < 1$ gibt, das nach dem Prinzip des nächsten Nachbarn auch den vierten Vektor korrekt klassifiziert. Der tiefere Grund hierfür ist ein geometrischer (jedes solch Maß definiert eine Schar paralleler Grade), der auch bei dem XOR-Problem im Zusammenhang mit Schwellwertneuronen auftritt.

Neben diesen bescheidenen Maßen gibt es sehr viel komplexere.

Die Ähnlichkeitsmaße lassen sich nun in Kategorien einteilen. Da unterscheiden wir folgende Typen:

1. *Abzählende Ähnlichkeiten*: Man zählt Vorkommen von Objekten (evtl. mit Gewichten) in der Darstellung ab, wie beim Hammingmaß.
2. *Metrische Ähnlichkeiten*: Dies sind Variationen des euklidischen Abstandes.

Als Beispiele haben wir gewichtete Abstände

$$d(x, y) = \sqrt{\sum_{i=1}^n \omega_i \cdot (x_i - y_i)^2}$$

sowie das Minkowski-Maß

$$d_{\text{mink}}(x, y) = \sqrt[p]{\sum_{i=1}^n |x_i - y_i|^p}$$

3. *Transformationsähnlichkeiten*: Hier addiert man die Kosten der Schritte die man braucht um ein Objekt in das andere zu überführen, z.B. die Anzahl der Druckfehler.

Die hauptsächlichen Transformationen sind bekanntlich: einfügen, weglassen und modifizieren.

4. Strukturorientierte Maße: Sie richten sich nach der Struktur der zu vergleichenden Objekte, z.B. in einer Objektorientierung. Wenn etwa Taxonomien involviert sind, kann man die Länge der Pfade von einem zum anderen Objekt berücksichtigen.

Die betrachteten Strukturen sind hier symbolischer Art und die Argumente sind keine Zahlen sondern Symbole. Die einfachste Art, Ähnlichkeiten zwischen Symbolen zu beschreiben, benutzt Tabellen der folgenden Form:

Tabelle 9.3: Ähnlichkeiten zwischen Symbolen

$[x, y]$	v_1	v_2	\dots	v_k
v_1	$s[1, 1]$	$s[1, 2]$	\dots	$s[1, k]$
v_2	$s[2, 1]$	$s[2, 2]$	\dots	$s[2, k]$
\dots	\dots	\dots	\dots	\dots
v_k	$s[k, 1]$	$s[k, 2]$	\dots	$s[k, k]$

Man kann sich unter Umständen viele Einträge sparen, wenn Beziehungen zwischen den Symbolen vorhanden sind.

Symbolische und numerische Ähnlichkeiten kann man auch kombinieren. Ein Beispiel

$$\text{HEOM}(x, y) = \sqrt{\sum_{i=1}^n d_i(x_i, y_i)^2} \quad d_i = \begin{cases} H(x_i, y_i) \\ \text{dist}(x_i, y_i) \end{cases}$$

Dabei ist H ein symbolisches Maß und dist ein numerischer (z.B. euklidischer) Abstand, je nachdem was die Argumente sind.

5. *Informationsorientierte Maße*: Sie vergleichen die Informationen, die in den Objekten vorhanden sind. Das betrifft z.B. Texte und Bilder.

Als Beispiel erwähnen wir das Lin-Maß:

$$\text{sim}_L(o_1, o_2) = 2IC(p_m(o_1, o_2))/(IC(o_1) + IC(o_2)).$$

Dabei ist $p_m(o_1, o_2) := \min(p(o)|o \in P(o_1 o_2)$ und $P(o_1 o_2)$ ist in einer Hierarchie die Menge der gemeinsamen Vorgänger der Argumente. IC bezeichnet den Informationsinhalt (information content).

6. *Relevanz orientierte Maße*: Sie gewichten die verschiedenen Relevanzaspekte, die in den Objekten vorhanden sind, und vergleichen sie. Das passiert in einer Dokumentensuche. Diese Methoden findet man auch oft im Information Retrieval.

7. *Dynamische Maße*: Sie betrachten dynamische Prozesse und deren Eigenschaften.

Hier spielt die Statistik eine wesentliche Rolle. Ein bekanntes Maß ist DTW (dynamic time warping), das zwei Zeitreihen vergleicht.

Die Variablen i und j stehen für natürliche Zahlen, die Zeitpunkte bezeichnen und

Die Funktion $d(i, j)$ ist der Abstand zwischen dem i -ten Wert der ersten Zeitreihe und dem j -ten Wert der zweiten Zeitreihe. Wenn wir nun einen Pfad haben der durch die Knoten $(i_1, j_1), \dots, (i_k, j_k), \dots$ geht, dann summieren wir diese auf:

$$D_{\text{Pfad}} := \sum_k d(i_k, j_k).$$

Ein optimaler Pfad definiert nun den Abstand zwischen zwei Zeitreihen. Der Startwert ist $DTW(0, 0) = 0$ und das Ende ist bei $i - 1, j - 1$.

Hier ist $DTW(i, j)$ die minimale akkumulierte Entfernung, wenn man die ersten i Werte der ersten Reihe auf die ersten j Werte der zweiten Reihe abbildet. Abbildung 9.11 zeigt die möglichen Schritte.

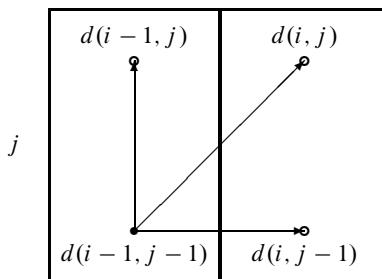


Abbildung 9.11: DTW Schritte.

Zwischen diesen Kategorien gibt es keinen klaren Abgrenzungen. Das Ganze gibt aber einen ersten Eindruck von der Vielfalt der möglichen Maße.

Auch wenn in der Anfrage oder Lösung Texte, Bilder oder vergleichbares verwendet wurden, ist deren Bedeutung häufig nicht klar. Von daher besteht das Bedürfnis, solche Repräsentationen in gleichwertige Attributdarstellungen umzuwandeln.

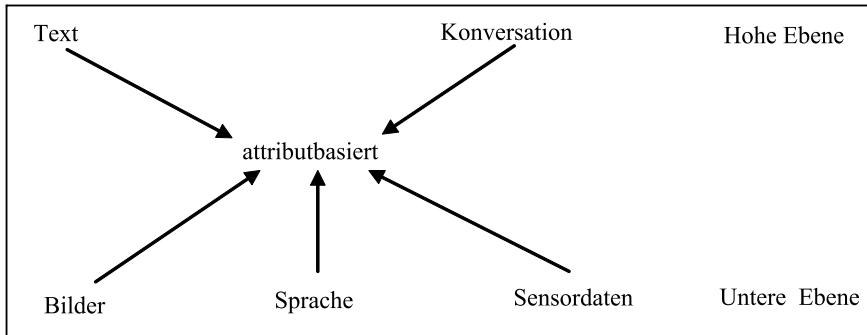


Abbildung 9.12: Transformationen.

Diese Transformationen sind oft nicht einfach.

9.7 Spezielle Retrievalfragen

In Datenbanken werden zum Retrieval gewöhnlich Baumstrukturen verwendet. Deren Grundidee, beim Abstieg auf einem Zweig alle anderen Zweige ausgeschaltet zu haben, ist hier nicht direkt anwendbar. Das liegt daran, dass gar kein bestimmtes Objekt gesucht wird sondern nur eines, das zum vorgelegten Objekt möglichst ähnlich ist. Trotzdem lässt sich die Idee der Baumstruktur verwenden; das am meisten verwendete Beispiel sind die kd-Bäume (kd: k-dimensional), deren Idee ist, die Fälle in möglichst homogener Weise zu repräsentieren ([7,24]). Gegeben seien k geordnete Wertebereiche T_1, \dots, T_k der Attribute A_1, \dots, A_k , eine Fallbasis $FB \subseteq T_1, \dots, T_n$ sowie ein Parameter b die Bucketgröße. Hier wird auch der Unterschied zwischen CBR und Datenbanken besonders deutlich.

Definition 9.7.1. Ein **kd-Baum** $\text{Tree}(FB)$ für die Fallbasis FB ist ein rekursiv definierter binärer Baum:

- Ist $|FB| \leq b$, dann ist $\text{Tree}(FB)$ ein Blattknoten (Bucket genannt), der mit FB markiert ist;
- ist $|FB| > b$, dann ist $\text{Tree}(FB)$ ein Baum, dessen Wurzel mit einem Attribut A_i und einem Wert $v_i \in T_i$ markiert ist und
- die zwei kd-Bäume $\text{Tree}(FB_{\leq})$ und $\text{Tree}(FB_{>})$ als Nachfolger besitzt, wobei $FB_{\leq} := \{x \in FB | x \leq v_i\}$ und $FB_{>} := \{x \in FB | x > v_i\}$

Ein kd-Baum partitioniert eine Fallbasis fortlaufend, die Buckets an den Blattknoten werden nicht weiter partitioniert. Bei der Erstellung eines kd-Baumes hat man an jedem Knoten zwei Freiheitsgrade, die Wahl des Attributs und die Wahl des partitionierenden Wertes. Eine zweckmäßige Wahl wird gewöhnlich aufgrund von statistischem Wissen (über effizientes Suchen)

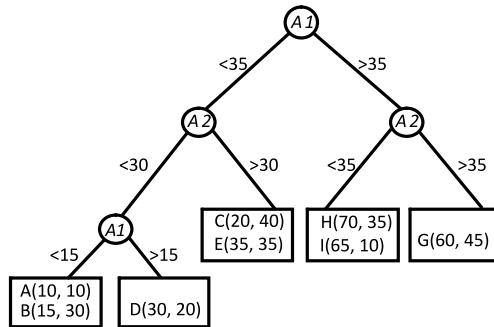
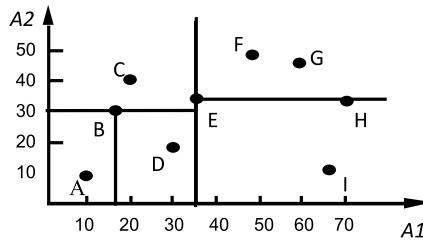


Abbildung 9.13: kd-Baum

getroffen, das aus der Fallbasis gewonnen wird. Für reell wertige T_i nimmt man oft dasjenige mit dem größten Interquartilabstand (wo also der Abstand der Punkte die das Ende des ersten und den Beginn des letzten Viertels markieren am größten ist). Bei der Wertauswahl kann man z.B. den Median nehmen.

Wir betrachten ein Beispiel für einen kd-Baum mit einer Fallbasis $FB = \{A, B, C, D, E, F, G, H, I\}$, wobei zwei reell wertige Attribute A_1 und A_2 angenommen werden; die Bucketgröße ist 2. Das erlaubt sowohl eine geometrische wie auch eine baumorientierte (d.h. eine graphische Baumdarstellung) Sicht. Die geometrische Sicht – siehe Abbildung 9.13 – kennt erst einmal nur Rechtecke.

Durch die Ähnlichkeit (oder die Abstände) kommen nun aber Hyperkugeln ins Spiel, mit den Beziehungen zwischen Hyperkugeln und Hyperquadern muss sich das Retrieval auseinandersetzen. Dazu werden die Fälle mit ihrer Ähnlichkeit in einem *Record* notiert und in einem Array zusammengefasst, Q bezeichnet die Anfrage und $Q[A_i]$ den Wert des Attributes A_i in der Anfrage:

```

SimCase = RECORD
  case: Fall;
  similarity: [0..1]
END;

SimCaseQueue = ARRAY[1..m] OF SimCase;

```

weiter verwenden wir als Variable scq : $SimCaseQueue$ (* die m ähnlichsten Fälle *).

Retrieval-Algorithmus in kd-Bäumen:

```

PROCEDURE Retrieve(K: kd-baum)
  IF K ist Blattknoten
  THEN
    FOR jeden Fall F von K DO
      IF sim(Q,F) > scq[m].similarity
      THEN füge F in scq ein
    ELSE (* innerer Knoten *)
      Sei  $A_i$  das Attribut und  $v_i$  der Wert mit dem K markiert ist
      IF  $Q[A_i] \leq v_i$ 
      THEN
        Retrieve( $K_{\leq}$ )
        IF BOB-Test ist erfüllt THEN Retrieve( $K_{>}$ )
      ELSE
        Retrieve( $K_{>}$ )
        IF BOB-Test ist erfüllt THEN Retrieve( $K_{\leq}$ )
      IF BWB-Test ist erfüllt
      THEN terminiere Retrieval mit scq
    ELSE RETURN
  
```

Hier müssen noch zwei Tests erklärt werden, die Kugeln mit Quadern vergleichen.

Der BOB-Test (*Ball-Overlaps-Bounds*) stellt fest, ob es ähnlichere Fälle als den bisher gefundenen m-ähnlichsten im angrenzenden Teilbaum gibt:

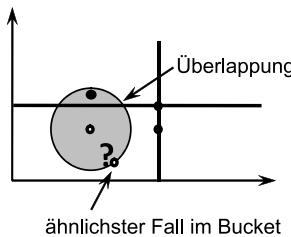


Abbildung 9.14: BOB-Test.

In diesem Test wird die Kugel um den Anfragefall durch ein Quader ersetzt. Die Kugel um den Anfragefall ist zunächst durch den aktuell m-nächsten Nachbarn definiert.

Der BWB-Test (*Ball-Within-Bounds*) stellt auf dieselbe Weise fest, ob die entsprechende Kugel ganz in einem Quader liegt.

Diese Tests können durch ein Preprocessing verbessert werden und führen dann zu den dynamischen Bounds-Tests. In Abhängigkeit von der aktuellen Fallbasis werden die durch den kd-Baum gegebenen Quadern verkleinert. Diese Quadern sind durch die in ihnen enthaltenen Fälle mit extremen Koordinatenausprägungen definiert. Dadurch werden Überlappungen mit anderen Quadern, in denen aber keine Fälle liegen, weitgehend vermieden.

Das Retrieval mit einem kd-Baum garantiert das Finden der m-nächsten Nachbarn, wenn das verwendete Ähnlichkeitsmaß eine Verträglichkeit mit der Ordnung auf den Wertemengen der Attribute aufweist:

$$\text{Wenn } x_i <_i x'_i <_i x''_i \text{ dann} \\ \text{sim}((x_1, \dots, x_n), (x_1, \dots, x''_i, \dots, x_n)) \geq \text{sim}((x_1, \dots, x_n), (x_1, \dots, x'_i, \dots, x_n))$$

Die nächste Methode der **Case-Retrieval-Netze** benutzt eine Variation der bisherigen Darstellungsmethoden wobei der Bedeutung von Relevanzen Rechnung getragen wird. In diesen Netzen treten an den Knoten sowohl Fälle wie auch Teile der Problembeschreibungen auf. Den Ausgangspunkt bilden Informationseinheiten (IE's) die als atomare Wissenseinheiten wie Attribut-Wert-Paare gedacht werden. Ein Fall besteht aus einer (mit einem Konstruktor CON strukturell aufgebauten) Menge von Informationseinheiten.

Definition 9.7.2. Ein **Retrieval Netz** (Basic Case Retrieval Net, BCRN) ist ein 5-Tupel $N = (E, C, \sigma, \varrho, \Pi)$ mit:

- (i) E ist eine endliche Menge von Knoten mit Informationseinheiten.
- (ii) C ist eine endliche Menge von Fallknoten.
- (iii) σ ist ein Ähnlichkeitsmaß: $\sigma : E \times E \rightarrow \mathbb{R}$; $\sigma(e, e')$ beschreibt die lokale Ähnlichkeit zwischen zwei IE's e und e' .
- (iv) ϱ ist eine Relevanzfunktion genannte Funktion $\varrho : E \times C \rightarrow \mathbb{R}$, $\varrho(e, c)$ beschreibt die Relevanz (das Gewicht) der IE e für den Fall c
- (v) Π ist eine Menge von Propagierungsfunktionen $\pi_n : \mathbb{R}^E \rightarrow \mathbb{R}$ für jeden Knoten $n \in UC$.

Zwischen den Informationseinheiten bestehen also Ähnlichkeiten und Informationseinheiten besitzen eine Relevanz für einen Fall. Knoten können nun aktiviert werden, wobei die initiale Aktivierung von gewissen Informationseinheiten durch die Anfrage geschieht. Diese Aktivierung wird durch das Netz bis zu den Fallknoten propagiert und der Fallknoten mit der höchsten Aktivierung geht dann als Sieger hervor, was der Auswahl als nächster Nachbar zur Anfrage entspricht.

Definition 9.7.3.

- (i) Eine Aktivierung eines BCRN ist eine Funktion $a : E \cup C \rightarrow \mathbb{R}$.
- (ii) Die Aktivierung zum Zeitpunkt t ist eine Funktion $\alpha_t : E \cup C \rightarrow \mathbb{R}$ die folgendermaßen bestimmt ist: durch
 - a) $\alpha_0(e) = 1$ falls e eine IE ist die in der Anfrage vorkommt und 0 sonst;
 - b) Für eine IE $e : \alpha_{t+1}(e) = \pi_e(\sigma(e', e)\alpha_t(e') | e' \in E)$
 - c) Für einen Fallknoten: $c : \alpha_{t+1}(c) = \pi_c(\varrho(e', c)\alpha_t(e') | e' \in E)$

Dabei werden zuerst die Ähnlichkeiten und dann die Relevanzen propagiert. Die Vorteile dieses Ansatzes sind effizientes Retrieval, Abhängigkeit des Aufwandes von der Anzahl der aktivierten IEs der Anfrage und die Möglichkeit des inkrementellen Erweiterns beim Auftreten neuer Fälle. Für viele Anwendungen ist auch von Vorteil, dass die Relevanzen zur Anfrage neu definiert werden können. Nachteile sind erhöhte Kosten zum Aufbau des Netzes; bei numerischen Attributen müssen ggf. neue Anfrageknoten erzeugt werden. Außerdem müssen bei hohem Vernetzungsgrad (viele IEs sind zueinander ähnlich) viele Propagierungen durchgeführt werden. Dies ist jedoch nur ein Basisansatz, der vielfach verbessert wurde, vgl. [11].

9.8 Fallbasisprobleme

In der Fallbasis steht das meiste Lösungswissen. Es hat den Vorteil, dass man die Fälle einfach in die Fallbasis eintragen kann, ohne sie im Geringsten zu verstehen. Das passiert erst, wenn man sie zur Laufzeit selektiert hat.

Es gibt zwei widersprechende Anforderungen an eine Fallbasis. Sie sollte einerseits möglichst groß sein, um die Anwendbarkeit zu erweitern, andererseits aber auch wieder möglichst klein, um die Suche nach dem nächsten Nachbarn effizient zu gestalten. Möglichkeiten zur Verkleinerung der Fallbasis sind durch die IBL-Algorithmen gegeben [2]; ein typischer Vertreter ist der IBL2-Algorithmus. Gegeben sei eine Trainingsmenge M von klassifizierten Objekten sowie ein Abstandsmaß d .

IBL2 Algorithmus:

```

1) FB := Ø;
2) FOR x ∈ M DO
   FOR y ∈ FB DO
     d(y) := d(x, y)
     Wähle  $y_{\max}$  so, dass  $d(y_{\max}) \leq d(y)$  für alle  $y \in FB$ 
     IF Klasse(x) = Klasse( $y_{\max}$ ) THEN "Klassifikation korrekt"
     ELSE "Klassifikation inkorrekt"
   FB := FB ∪ {x}.

```

Hier werden also nur bisher falsch klassifizierte Objekte in die Fallbasis aufgenommen. Problematisch ist, dass der Algorithmus von der Reihenfolge abhängig ist, in der die Trainingsmenge durchlaufen wird. Auch nach Erstellung einer Fallbasis muss diese gepflegt werden. Der Grund dafür ist, dass sich normalerweise der Kontext der Anwendung teils stetig, teils abrupt ändert.

Das gezielte Verkleinern der Fallbasis auf der Grundlage eines expliziten Kompetenzmodells findet man in [Smyth, Keane 95].

Um der Rolle des Wissens gerecht zu werden, ist es zweckmäßig, die Art der Probleme und Lösungen einer etwas genaueren Analyse zu unterziehen und zwar sowohl was die Semantik wie auch was die Pragmatik angeht. Die Aufgabenstellung erfolgt durch eine Institution genannt Kunde. Durch den Kunden wird insbesondere der Nutzen der möglichen Lösungen festgelegt. Mathematisch ist der Nutzen durch eine reell wertige Funktion u auf den Lösungen definiert, d.h. es ist $u(L) \in \mathbb{R}$; genau gesagt sollte eigentlich auch das Problem genannt werden, d.h. wir betrachten den Nutzen $u_P(L)$ für P . Es ergibt sich daraus die Anforderung, dass das fallbasierende System das zur Optimierung der Nutzenfunktion benötigte Wissen besitzen sollte. Um dies genauer beschreiben zu können ist der Begriff des Wissenscontainers zweckmäßig.

9.9 Adoptionsfragen

Anstatt viele Fälle zu speichern, kann es zweckmäßig sein, mit weniger Fällen auszukommen und die fehlenden durch eine Adaption der Lösung zu kompensieren. Hier ist vor allem ein Kompromiss zwischen dem Ziel, die Fallbasis klein zu halten und aufwendige und sehr wissensintensive Lösungstransformationen zu finden. Die Darstellung der Transformation erfolgt meist in

Form von Regeln. Diese bieten sich vor allem dann an, wenn bestimmte Änderungen von Parametern der Problemstellung funktional in Abhängigkeiten von Parametern der Lösung resultieren. Wenn z.B. im Problemteil a und im Lösungsteil c auftritt und die Beziehung $a : b = c : d$ besteht, so kann man dies in einer Regel der Form $C \wedge \text{analog}(a : b, c : d) \wedge P(b) \rightarrow L(d)$, wobei $L(c)$ die durch den Fall gelieferte Lösung für $P(a)$ war und C ein eventuelles Constraint ist. Solche **Adaptionsregeln** enthalten allgemein **Adaptionsoperatoren**, die die Art der Lösungstransformation genauer beschreiben. Die wichtigsten Arten sind:

1. Substitutionsadaptation: Ändern, Einfügen, Löschen von Lösungsbestandteilen oder Parametern;
2. Strukturelle Adaptation: Reorganisation (Umstrukturierung) der Lösung, Einfügen und Löschen von Objekten.

Die Adaptionsoperatoren beschreiben die Änderungsaktionen, weshalb eine Lösungstransformation gewöhnlich durch Anwendung einer Folge von Operatoren entsteht.

Dies führt zu dem Thema Adaptionsprozesse. Solche Prozesse sind sehr schwer zu konfigurieren. Dass liegt vor allem daran, dass einzelne Schritte den Nutzen der Lösung nicht unbedingt verbessern müssen, was aus dem Bergsteigermodell ja wohlbekannt ist.

9.10 Ein paar typische Anwendungen

9.10.1 Aufwands-Prognose

Wir möchten den Aufwand für Softwareprojekte vorhersagen und haben dazu eine Fallbasis angelegt. Den Aufwand messen wir in Mitarbeiterstunden.

Ein Fall ist hier von der Form (Projektbeschreibung, Aufwand). Das ganze steht und fällt mit einer adäquaten Projektbeschreibung. Diese sollte alle relevanten Einflussfaktoren in der Form von Attributen mit ihren Wertebereichen enthalten.

Das kann z.B. so geschehen:

- Sprachen = {C++, Java, VB, Java Script, VB Script, SQL, Php, Perl, Asp, Html, XML, andere}
- Datenbanksysteme = {Oracle, Access, SQLServer, MySQL, andere}
- Methodologie = {OO, SA, SD, RAD, JAD, MVC, andere}
- Komplexität von internen Berechnungen = {sehr niedrig, niedrig, mittel, hoch, sehr hoch}
- Anzahl der Monate mit Erfahrungen mit den Tools = natürliche Zahl
- Anzahl der anzusprechenden Datenfiles = natürliche Zahl
- Anzahl der Inputeinträge = natürliche Zahl
- Anzahl der Outputeinträge = natürliche Zahl
- Teamgröße = natürliche Zahl
- Funktionalität = Vektor der prozentualen Anteile von (1-Interner Prozess, 2-Daten Einträge/Modifikation/ Entfernen, 3-Output, 4-Datenanfrage an Datenbanken / 5-Drucken, 6-Report.

Wenn man diese Attribute noch gewichtet, kann man mittels gewichteter linearer Maße vernünftige Ergebnisse erzielen. Grundsätzlich handelt es sich hier um eine Vorhersage, für die

man klassische Methoden hat. In sehr komplexen Situationen wie der vorliegenden ist jedoch der CBR-Ansatz sinnvoll. Vgl. hierzu [13].

Ein weiteres Beispiel, diesmal zur Vorhersage von Fehlern in Softwareprojekten wird in [15] gezeigt.

9.10.2 E-commerce

Als Illustration zum e-commerce betrachten wir eine elektronische Firma die Häuser verkauft. Dazu werden den Kunden Häuser angeboten, die mit Attributen beschrieben werden.

Dazu muss der Kunde alles Wichtige wissen. Zuerst müssen wir den Kundentyp festlegen, z.B. Berufstätiges Paar.

Bei den Attributen ist zu sagen, dass man sie nicht unbedingt für jedes Haus einzeln aufzählen muss, wenn sie für viele Häuser gleichzeitig, etwa für ganze Straßenzüge, zutreffen. Dann werden sie einmal beschrieben und als Attributwert erscheint dann eine Referenz. Auch müssen nicht alle Details aufgelistet werden, wenn sie die Kaufentscheidung nicht groß beeinflussen. Als Attribute kann man die folgenden wählen:

Tabelle 9.4: *Hausverkauf*

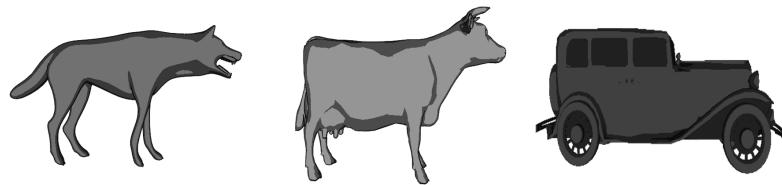
Attribute	Werte	Wertebereiche
Haustyp	Symbol	Duplex, Hochhaus, Einfamilienhaus, Zweifamilienhaus, Landhaus
Soziale Umgebung	Symbole	Referenz
Preis	Zahl	Reelle Zahl
Zimmer	Zimmerzahl	Natürliche Zahl, 1–8
Badezimmerzahl	Kleine Zahl	{0, 1, 2}
Details	Hausdetails	Referenz
Zusätze	Details	Referenz
Alter	Anzahl der Jahre	0–30
Gegend	Ortsteile	{Aufzählung}
Bezahlung	Art	{Hypothek, Barzahlung}
Einzug	Wann?	Datum
Hausgröße	Quadratmeter	Natürliche Zahl

Diese Attribute müssen nun gewichtet werden. Macht das die Firma, so geschieht das uniform für einen ganzen Kundentyp. Der Kunde trägt als Wunsch die Attributwerte ein und der nächste Nachbar wird offeriert.

9.10.3 Skizzen von Bildern

Wir wollen Bilder aufgrund von Skizzen, sog. Skeletons, erkennen.

Für die Darstellung können jetzt keine Attribute verwendet werden, sondern Bilder und Skizzen, Als Beispiel betrachten wir Bilder von Tieren und Autos:



Dies ist die „Produktbasis“, d.h. Die Lösungsmenge. Eine Frage ist nun eine Skizze. Dazu drei Beispiele:

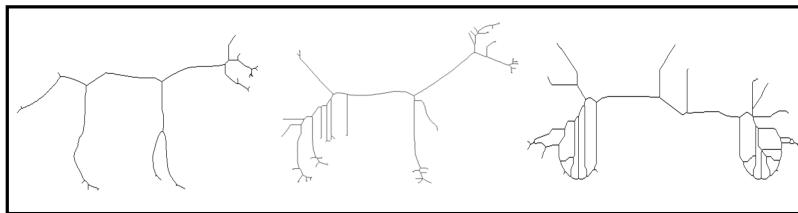


Abbildung 9.15: Bilder und Skeletons.

Die Antwort der Frage ist dann ein Objekt der Lösungsmenge. Dieser Art Probleme können sehr komplex werden, wo sich dann die Anwendung von CBR empfiehlt.

9.11 Methodologie zum Aufbau eines CBR-Systems und Integrationsfragen

9.11.1 Generelles

Ein wichtiger Aspekt für die Erstellung von CBR-Systemen ist, dass sie auch in einem weniger ausgereiften Zustand bereits hilfreich sein können. Das liegt wesentlich an dem Approximationscharakter ihrer Lösungen: Sie liefern dann eben weniger gute oder in weniger Situationen gute Lösungen, versagen aber nicht völlig.

Zum Aufbau eines Systems hat man eigentlich nur die Wissenscontainer zu füllen.

Der einfachste Container in dieser Hinsicht ist die Fallbasis. In vielen Situationen liegen die Fälle bereits vor, wenn auch nicht immer in der benötigten Syntax. Oft muss man noch zusätzlich Fälle erzeugen, aber das geschieht meist zu einem späteren Zeitpunkt.

Das Vokabular der Repräsentationssprache wird zu Anfang überwiegend aus vorhandenen Attributen aufgebaut sein. Das Vokabular kann sehr reichhaltig und komplex sein und hierarchisch aus einfachen Elementen aufgebaut sein. Dann muss nachgeprüft werden welche virtuellen Attribute man braucht.

Das Ähnlichkeitsmaß wird zu Anfang eher grobe Einteilungen machen. Seine Wahl richtet sich einmal nach der vorliegenden Repräsentationssprache und zum anderen nach den allgemeinen Zielsetzungen. Hier ist ein wichtiger Aspekt der Anforderungsanalyse, das herauszufinden, was

für den Kunden wichtig ist. Dies resultiert wesentlich im Aufstellen der einzelnen Präferenzrelationen des Kunden sowie einer Abschätzung ihrer Wichtigkeit. Das Ziel der Einführung neuer virtueller Attribute ist es, diese so zu wählen, dass die zugehörigen lokalen Ähnlichkeitsmaße ähnlich geordnet zu den Nutzenfunktionen der einzelnen Präferenzrelationen sind.

Das Adoptionswissen wird zu Anfang eher klein sein. Eine Ausnahme bilden elektronische Kataloge, wo die Adaptionen in der Regel mitgeliefert werden.

Weitere Verbesserungen erreicht man durch Verschieben des Wissens zwischen den Containern. Das kann u.a. mit folgenden Methoden erfolgen:

1. Verkleinern der Fallbasis z. B. nach dem Vorbild des IBL2-Algorithmus sowie evtl. auch gezielter Konstruktion von Fällen dienen, hinzugenommen werden;
2. Verbesserung des Maßes (im Wesentlichen der Gewichte) durch Experimente und Einsatz von Lernverfahren vgl. [25]
3. Aufbau des Adoptionswissens durch Wissensakquisition und Einsatz von Lernverfahren

Betrachtet man Systeme die dauernd und für längere Zeit im Einsatz sind, dann ist dieser Prozess nie abgeschlossen und findet in Form von Wartung statt. Die Entwicklung eines CBR-Systems ist wesentlich eine Softwareengineeringaufgabe und sollte sich der in diesem Gebiet erprobten Methoden bedienen. Für CBR-Systeme ergeben sich hier eigene Charakteristika. Für ein breites Spektrum von CBR-Anwendungen wurde hierfür eine Methodologie entwickelt, die in [5] beschrieben ist.

9.11.2 Integration in übergeordnete Problemlöser

Das fallbasierte Schließen hat sich zu einer Technologie entwickelt, die auf bestimmte und hier umschriebene Probleme zugeschnitten ist. Wie bei jeder Technologie treten diese Probleme nur selten in der reinen Form auf, dass gerade sie ausschließlich zu verwenden wäre. Die Integration mit anderen Techniken kann auf verschiedenen Ebenen erfolgen:

1. Toolbox-Ebene: Die einzelnen Teile sind über ein gemeinsames Benutzerinterface zugänglich.
2. Kooperative Ebene: Die Teile tauschen Resultate in einer gemeinsamen Repräsentationssprache aus.
3. Werkbank-Ebene: Einzelne Module arbeiten zusammen.
4. Nahtlose Integration: Die Techniken arbeiten auf einer Plattform für den Benutzer unsichtbar zusammen.

Jede dieser Ebenen hat eigene Rechtfertigungen, die nahtlose Integration erfordert aber das tiefste Verständnis der Einzeltechniken.

Abschlussbemerkung :

Vieles in diesem Artikel beruht auf einem Lehrbuch: Case-Based Reasoning, a Textbook by Rosina Weber und Michael Richter, das im Springer-Verlag erscheint.

Literaturverzeichnis

- [1] A. Aamodt, E. Plaza: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications* 7(1) (1994), S.39–59.
- [2] D. W. Aha, D. Kibler, M. K. Albert: Instance-based Learning Algorithms. *Machine Learning* 6 (1991), S. 37–66.
- [3] R. Bergmann, W. Wilke: Towards a new formal model of transformational adaptation in case-based reasoning. In H. Prade (Hrsg.) 13th European Conference on Artificial Intelligence (ECAI'98), John Wiley & Sons, 53–57.
- [4] R. Bergmann, S. Stahl: Similarity measures for object-oriented case representations. In B. Smyth & P. Cunningham (Hrsg.) *Advances in Case-Based Reasoning (EWCBR'98)*, Springer, 25–36.
- [5] R. Bergmann, S. Breen, M. Göker, M. Manago, S. Wess: Developing Industrial Case-Based Reasoning Application – The INRECA- Methodology. *Springer Lecture Notes in AI* 1612, 1999.
- [6] H.-D. Burkhard, M. M. Richter: On the Notion of Similarity in Case-Based Reasoning and Fuzzy Theory. Erscheint in: „Soft Computing in Case Based Reasoning“, ed. P. Sankar.
- [7] J. H. Friedman, J. L. Bentley, R. A. Finkel: An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Software* 3 (1977); S. 209–226.
- [8] K. Hanney, M. T. Keane: The Adaptation Knowledge Bottleneck: How to Ease it by Learning from Cases. In: *Case-Based Reasoning Research and Development* (ed. D. B. Leake, E. Plaza), *Springer Lecture Notes in AI* 1266, S. 359–370, 1997.
- [9] J. Kolodner: *Case-Based Reasoning*. Morgan Kaufman Publ. 1993.
- [10] J. Kolodner: Making the Implicit Explicit: Clarifying the Principles of Case-Based Reasoning. In: *Case-Based Reasoning – Experiences, Lessons and Future Directions*, (ed. D. B. Leake), AAAI Press/The MIT Press (1996), S. 349–370.
- [11] M. Lenz: Case Retrieval Nets as a Model for Building Flexible Information Systems. Dissertation Humboldt Universität Berlin 1999.
- [12] M. Lenz, B. Bartsch-Spörl, H.-D. Burkhard, S. Wess (eds.): *Case-Based Reasoning Technology*. *Springer Lecture Notes in AI* 1400, 1998.
- [13] Li, J., Ruhe, G., Al-Emran, A., Richter, M.M.: A Flexible Method for Software Effort Estimation by Analogy. *Journal of Empirical Software Engineering EMSE* 12, no.1, <http://dx.doi.org/10.1007/s10664-006-7552-4>. 2006.
- [14] H. Muñoz-Avila, F. Weberskirch: Planung for Manufacturing Workpieces by Storing, Indexing and Replayng Planning Decisions. *AIPS 96* (ed. Brtian Drabble), S. 150–157
- [15] Elham Paikari, Michael M.Richter, Guenther Ruhe: Defect Prediction, using Case-Based Reasoning: An Attribute Weighting Technique Based upon Sensitivity Analysis in Neural Networks. *International Journal of Software Engineering and Knowledge Engineering*, 2012.
- [16] M. M. Richter: On the notion of similarity in case-based reasoning. *Mathematical and Statistical Methods in Artificial Intelligence* (ed. G. della Riccia et al.), Springer Verlag 1995, S. 171–184.
- [17] Michael M. Richter: Similarity. In: *Case-Based Reasoning for Signals and Imaging*, ed. Petra Perner, Springer Verlag 2007, pp.25–90.
- [18] Michael M. Richter: Foundations of Similarity and Utility. *Proc. Flairs 07*, AAAI Press

- [19] Richter, M. M., Weber, R.: Case-Based Reasoning, a Textbook. To appear in Springer Verlag.
- [20] C. K. Riesbeck, R. C. Schank: Inside Case-Based Reasoning. Lawrence Erlbaum Associates 1989.
- [21] R. C. Schank: Dynamic Memory: A theory of Learning in Computers and People. Cambridge University Press 1982.
- [22] S. M. Sohan, Michael M. Richter and Frank Maurer, http://ase.cpsc.ucalgary.ca/ase/uploads/Publications/XP_2010_Sohan.pdf, Proc. of 11th International Conference on Agile Processes and eXtreme Programming (XP 2010), Trondheim, Norway, 2010.
- [23] B. Smyth, M. R. Keane: Remembering to Forget, Proc. IJCAI '95.
- [24] S. Wess: Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik. Dissertation Kaiserslautern 1995, Reihe DISKI 126, infix Verlag.
- [25] D. Wettschereck und D. W. Aha: Weighting Features, Proc. ICCBR '95, Springer Lecture Notes in AI

10 Planen

Alexandra Kirsch, Josef Schneeberger

Intelligentes Verhalten wird oft damit assoziiert, systematisch auf ein nachvollziehbares Ziel hin zu agieren. Aus diesem Grund werden seit langem Verfahren erforscht, die aus einer Aufgabenbeschreibung eine geeignete Folge von Aktionen – einen Plan – ermitteln. Es wurden zwei komplementäre Ansätze zur Erzeugung von Aktionsfolgen entwickelt: *Handlungsplanung* und *situierter Aktivität*. Beide Ansätze haben spezifische Stärken und Schwächen. Handlungsplanung – kurz als Planen oder Planung bezeichnet – ist angemessen, wenn Aktionen in einer sinnvollen Reihenfolge angewendet werden müssen, um ein dezidiertes Ziel zu erreichen, oder auch, wenn die einzelnen Aktionen stark voneinander abhängen. Die Verwendung situierte Aktivitäten ist sinnvoll, wenn die beste Aktion in der aktuellen Situation leicht berechnet werden kann, d.h. wenn keine Vorausschau auf mögliche Komplikationen von Handlungen notwendig ist.

Wenn es z.B. das Ziel ist, an einem Kongress in Berlin teilzunehmen, dann empfiehlt es sich vorher die Reise zu planen. Teilziele bei der Reise zur Konferenz sind etwa: Erwerb einer Bahnfahrkarte mit Sitzplatzreservierung, Hotelreservierung, Fahrten zum Bahnhof und zum Hotel, Registrierung bei der Konferenz etc. Um alle diese Ziele zu erreichen, muss eine komplexe Folge von Aktionen in der richtigen Reihenfolge ausgeführt werden. Schon ein kleiner Fehler bei der Planung kann zu unangenehmen Folgen führen: Eine falsche Einschätzung der Fahrzeit zum Bahnhof kann zum Verpassen des Zuges führen.

Wenn andererseits das Ziel lautet, von einem überfüllten Bahnsteig aus den Zug zu besteigen, dann ist eine ausgefeilte Planung von Aktionen nicht so wichtig. Zunächst reicht es aus, anderen Personen oder Gepäckstücken auszuweichen und evtl. den Weg für andere Fahrgäste freizumachen. Dann muss rechtzeitig der Weg in den Zug gefunden werden, damit dieser nicht abfährt, bevor man eingestiegen ist. Zu langes Planen kann verhindern, dass die richtige Gelegenheit zum Einsteigen genutzt wird.

In diesem Kapitel¹ konzentrieren wir uns auf die Handlungsplanung – auch als „klassisches Planen“ bezeichnet – und vernachlässigen die situierter Aktivität. Es soll aber noch betont werden, dass Agenten für ein sinnvolles Vorgehen fast immer beide Arten der Planung benötigen: Handlungsplanung um mittel- und langfristige Ziele zu erreichen, und situierter Aktivität, um auf aktuelle und unvorhersehbare Einflüsse zu reagieren. In Abschnitt 10.1 klären wir zunächst was wir unter Aktionen und Plänen verstehen und diskutieren die Aufgabenstellung des Planens. Im den folgenden Abschnitten beschreiben wir Planen als Suche im Raum der Zustände (Abschnitt 10.2) und im Raum der Pläne (Abschnitt 10.3). Graphbasierte Verfahren werden in

¹ Ausgangspunkt für diesen Beitrag war der Abschnitt „Planen“ aus früheren Auflagen dieses Buches mit den weiteren Autoren S. Biundo und J. Hertzberg. Teilweise wurden Absätze aus diesem Artikel wörtlich übernommen. Inhaltlich stützt er sich auch auf die beiden Überblicksartikel [47, 48] im AI Magazine.

Abschnitt 10.4 vorgestellt. Das Kapitel endet mit einem Ausblick auf Erweiterungen der vorgestellten Verfahren (Abschnitt 10.5) und Hinweisen zu weiterführender Literatur und anderen Informationsquellen (Abschnitt 10.6).

10.1 Repräsentation von Planungsproblemen

Die klassische Sicht auf das Planen ohne Berücksichtigung von Zeit wurde geprägt durch den Situationskalkül von McCarthy und Hayes [28]. Grundelemente des Situationskalküls sind *Situationen*² und *Aktionen*. Eine Situation ist der Schnapschuss des interessierenden Ausschnitts der Welt zu einem Zeitpunkt, beschrieben durch eine Menge logischer Formeln, die in dieser Situation gelten. Eine Aktion ist die formale Beschreibung einer Handlung in der Welt und sie überführt eine gegebene Situation in eine andere Situation, die Nachfolgesituation. Der Situationskalkül ist eine formale Vorschrift, wie man aus einer gegebenen Situation S und einer Aktion, die in S ausgeführt wird, die Nachfolgesituation S' errechnen kann.

Die Verwendung von Situation, Aktion, Nachfolgesituation und so fort ist die Art, wie beim klassischen Planen Zeit betrachtet wird. Dabei geht es um das Vorher und Nachher von unterschiedlichen, untereinander widersprüchlichen Situationen beziehungsweise um das Vorher und Nachher von Aktionen. Beim zeitlichen Schließen im allgemeinen sind es nicht nur Aktionen, die Veränderungen bewirken, sondern beliebige Ereignisse.

Bei dieser Art der Behandlung von Zeit treten einige grundsätzliche Schwierigkeiten auf, die jedes Planungssystem effizient lösen oder durch entsprechende Maßnahmen umgehen muss, wenn die Veränderung von Fakten in der Zeit betrachtet wird.³

- Das Qualifikationsproblem (*engl. qualification problem*) ist das Problem, korrekte Vorhersagen über die Fakten nach Ausführung einer Menge von Aktionen zu machen, *ohne alle (vorhandene) Information über die Situation vor ihrer Ausführung berücksichtigen zu müssen*.
- Beim Vorhersageproblem (*engl. prediction problem*) geht es um die Schwierigkeit vorherzusagen, welche Fakten nach Ausführung einer Menge von Aktionen *wahr sein werden*.
- Das Persistenzproblem (*engl. persistence problem*) ist das Problem, vorherzusagen, welche Fakten durch Ausführung einer Menge von Aktionen *unverändert bleiben werden*.
- Das Frame-Problem (*engl. frame problem*) ist das Problem, vorherzusagen, welche Fakten bei Ausführung *einer einzigen Aktion* unverändert bleiben werden.
- Das Verzweigungsproblem (*engl. ramification problem*) ist das Problem, vorherzusagen, *welche weiteren Fakten wahr werden*, wenn durch Ausführung einer Aktion in einer Situation ein Faktum wahr wird.

Ein Planer, der effizient laufen soll, muss diese Probleme alle entweder effizient lösen, oder er muss sie umgehen. Alle diese Probleme zusammen sind im allgemeinen unentscheidbar. Wer also halbwegs effiziente Planer bauen will, sollte seine Ansprüche an eine allgemeine Lösung der beschriebenen Probleme innerhalb eines solchen Planers zurücknehmen. In der Regel wird man mit Verfahren arbeiten, welche die beschriebenen Ableitungen vorwärts und rückwärts in

² Situationen werden auch oft als *Zustände* bezeichnet. Wir verwenden beide Begriffe synonym.

³ Das Frame-Problem und das Qualifikationsproblem werden auch im Kapitel 6 diskutiert.

der Zeit nur inkorrekt machen, oder man wird den Bereich so einschränken, dass korrekte und vollständige Verfahren noch effizient sein können. Die Verwendung inkorrekt und unvollständiger Verfahren ist nicht so bedenklich wie sie klingen mag, weil Planung sich in der Regel sowieso unter Unsicherheit vollzieht, so dass es nicht immer Sinn macht, auf korrekte und vollständige Verarbeitung von Information zu beharren, die von sich aus unsicher ist.

Durch die Existenz der genannten Probleme wird praktisch die Korrektheit und Vollständigkeit eingeschränkt, mit der Planer in ihrem Anwendungsbereich arbeiten können. Darüber hinaus ist jedoch noch eine grundsätzliche Einschränkung in der Ausdrucksfähigkeit des Vokabulars zu beachten, mit dem die Effekte von Aktionen beschrieben werden.

Versteht man Planen als Arbeiten im Situationskalkül, und nimmt man weiterhin an, dass Situationen als Mengen von logischen Formeln dargestellt sind, dann liegt es nahe, Aktionen als Einheiten zu verstehen, die Situationen in Situationen dadurch abbilden, dass sie aus der Vor-Situation Formeln herausnehmen und neue hinzutun, um ihre Nach-Situation zu erzeugen. Das ist genau das Vorgehen von STRIPS [13], das im folgenden Abschnitt 10.1.1 genauer beschrieben wird.

10.1.1 Mengenbasiertes Planen: STRIPS

Wir beginnen diesen Überblick über Planungsmethoden und -systeme mit dem Pioniersystem STRIPS⁴ [12]. Viele Detailentscheidungen der Autoren von STRIPS sind auch heute noch für das Planen maßgebend. Zum Beispiel gehen die beiden oben genannten Annahmen (Weltbeschreibung bzw. Operatordefinition als Mengen) auf STRIPS zurück. Die Idee, die Welt durch eine Menge von Prädikaten zu beschreiben, hat einige wichtige Vorteile: Zum einen ist das simulierte Ausführen einer Aktion durch simple Mengenoperationen möglich; zum anderen können wir die auftretenden Prädikate als logische „Theorie“ (über die Welt) auffassen, indem wir sie als konjunktiv verknüpft interpretieren.

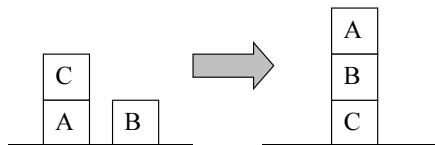


Abbildung 10.1: Start- und Zielsituation eines einfachen Beispiels aus der Klötzenwelt.

Betrachten wir ein einfaches, jedoch klassisches Spielproblem aus der Klötzenwelt (*engl. blocks world*), das aus einem Tisch und drei Klötzen A, B und C besteht, graphisch dargestellt auf der linken Seite von Abb. 10.1. Diese Situation kann mit folgenden Literalen repräsentiert werden:

auf(A,Tisch) auf(C,A) auf(B,Tisch) frei(B) frei(C)

STRIPS erfordert, dass die Startsituation vollständig beschrieben ist. Das bedeutet, dass alle Literale, die nicht explizit aufgeführt wurden, als falsch angenommen werden (diese Annahme

⁴ STRIPS steht für „Stanford Research Institute Problem Solver“, ein früher Planer, der zur Steuerung des Roboters „Shakey“ implementiert wurde.

heißt Closed World Assumption (CWA); siehe Kap. 6). Neben den angegebenen Literalen gilt also z.B. auch $\neg \text{auf}(A,C)$ in dieser Situation.

STRIPS macht auch Einschränkungen bei der Formulierung von Zielen. Es sind nur positive Literale erlaubt, und nur solche Ziele können formuliert werden, die durch den Plan herbeigeführt werden sollen. Das schließt beispielsweise alle Ausdrücke aus, die das Verhalten eines Agenten über einen Zeitraum hinweg beschreiben. Wenn die Aufgabe lautet nach Berlin zu reisen und dabei die Fahrkarte nicht zu verlieren, so lässt sich in der STRIPS Repräsentation nur die Zielposition Berlin beschreiben, nicht aber das Aufbewahren der Fahrkarte. Entsprechend wird die Zielsituation auf der rechten Seite der Abb. 10.1 in unserem Beispiel durch die Konjunktion der Literalen $\text{auf}(B,C)$ und $\text{auf}(A,B)$ beschrieben.

Um das Planungsbeispiel zu vervollständigen, benötigen wir noch eine Beschreibung der Operatoren (bezeichnet mit Λ), die vom Agenten ausgeführt werden können. Wie oben erwähnt geschieht dies durch die Spezifikation einer Vor- und einer Nachbedingung. Die Vorbedingung besteht – wie das Ziel – aus einer Konjunktion von positiven Literalen, während die Nachbedingung auch negative Literalen enthalten kann (ebenfalls konjunktiv verknüpft). Der Operator „bewege Klötzchen x von Klötzchen y auf den Tisch“ $\text{bewege}(x,y,\text{Tisch})$ wird also spezifiziert durch:

Vorbedingung: $\text{auf}(x,y) \wedge \text{frei}(x)$

Nachbedingung: $\text{auf}(x,\text{Tisch}) \wedge \text{auf}(x,y) \wedge \text{frei}(y)$

Im Unterschied zur Startsituation und zum Ziel enthalten solche Operatoren Variablen, die durch die Bezeichnungen der Klötzchen ersetzt werden müssen, bevor die entsprechende Aktion in den Plan aufgenommen werden kann. Wir erhalten also die Aktion **bewege-C-von-A-auf-den-Tisch** mit Vor- und Nachbedingung:

Vorbedingung: $\text{auf}(C,A) \wedge \text{frei}(C)$

Nachbedingung: $\text{auf}(C,\text{Tisch}) \wedge \text{auf}(C,A) \wedge \text{frei}(A)$

Aktionen können ausgeführt werden, wenn die Vorbedingungen erfüllt sind, in unserem Fall wenn das Klötzchen C auf A steht und sich auf C kein weiterer Gegenstand befindet. Beim Ausführen einer Aktion ändert sich die Situation wie folgt: alle positiven Literalen der Nachbedingung werden zur Situationsbeschreibung hinzugefügt, während alle negativen Literalen daraus gelöscht werden. Führen wir beispielsweise die Aktion **bewege-C-von-A-auf-den-Tisch** in der oben beschriebenen Startsituation aus, dann erhalten wir eine Situation mit den folgenden Literalen:

$\text{auf}(A,\text{Tisch}) \text{ auf}(B,\text{Tisch}) \text{ auf}(C,\text{Tisch}) \text{ frei}(A) \text{ frei}(B) \text{ frei}(C)$

Ein Planer, der die Operatorbeschreibungen Λ sowie die Start- und Zielbeschreibung als Eingabe erhält, sollte eine Folge von Aktionen zurückliefern, die, nach Ausführung in der Startsituation, das Ziel herbeiführt. Wenn wir die oben spezifizierte Start- und Zielsituation nehmen, den **bewege-C-von-A-auf-den-Tisch** Operator und zwei weitere sehr ähnliche Operatoren, dann würden wir den trivialen Plan

bewege-C-von-A-auf-den-Tisch
 bewege-B-vom-Tisch-auf-C
 bewege-A-vom-Tisch-auf-B

10.1.2 Die Planungsaufgabe

Bevor wir uns konkreten Verfahren zur Lösung von Planungsaufgaben zuwenden, wollen wir zunächst eine Planungsaufgabe⁵ bzw. Planungsproblem formal beschreiben [16]. Eine *Planungsaufgabe* bzw. ein *Planungsproblem* ist ein Tripel (Σ, S_0, Z) mit folgenden Eigenschaften:

- S_0 ist die Startsituation,
- Z ist die Menge der Zielsituationen, die durch eine Menge von Atomen spezifiziert sind,
- $\Sigma = \langle \mathcal{K}, \mathcal{A}, \gamma \rangle$ ist ein Zustandsübergangssystem, das Situationen in ihre Nachfolgesituationen abbildet.
- \mathcal{K} ist die Menge aller Grundatome, die Objekte der Planungsaufgabe bezeichnen.
- \mathcal{A} ist die Menge aller Aktionen, d.h. Grundinstanzen der Operatoren Λ , und
- γ ist die Zustandübergangsfunktion, sie bildet Situationen auf Situationen ab.

Das Zustandsübergangssystem Σ hat beim klassischen Planen eine Reihe von Einschränkungen, die eine Planungsaufgabe einfacher machen. Die Welt ist für den Planer vollständig wahrnehmbar, d.h. es gibt keine Eigenschaften einer Situation, die der Planer nicht sehen und damit nicht im Plan berücksichtigen kann. Die Planungsdomäne ist endlich, d.h. es gibt nur endliche viele Zustände bzw. Aktionen. Situationen sind statisch und Aktionen deterministisch. Dies bedeutet, dass alle Änderungen in einer Situation ausschließlich durch die bekannten Aktionen des Planers verursacht werden können und diese haben genau die Wirkung, die in der Operatorbeschreibung angegeben ist. Als Ziele von Aktionen können nur solche formuliert werden, die durch den Planer aktiv herbeigeführt werden können. Das Vermeiden bestimmter Situationen oder die Berücksichtigung von Zwischenzielen ist nicht möglich. Pläne sind sequenziell, d.h. eine total geordnete Liste von Aktionen und Zeit wird in den Plänen nur implizit betrachtet, d.h. die zeitliche Dauer von Ereignissen und Aktionen wird ignoriert. Schließlich wird das Planen und die Plan-Ausführung getrennt voneinander betrachtet. Der Planer kennt die tatsächlichen Effekte der Ausführung des einmal geplanten Plans nicht.

10.1.3 Propositionale Repräsentationen

Im vorangegangenen STRIPS Beispiel wurden die Operatoren mit Variablen, die Start- und Zielbeschreibung aber mit Konstanten formuliert. Andere Planungsaufgaben erfordern auch andere Formulierungen von Start und Ziel sowie den Operatoren. Ganz allgemein können hier alle Varianten von logischen Formeln verwendet werden, jedoch hat diese Wahl weitreichende Folgen in Bezug auf die Komplexität der erforderlichen Planungsverfahren.

⁵ Auf Englisch wird eine Aufgabe als *problem* bezeichnet, deshalb hat sich auch der Begriff Planungsproblem eingebürgert und wir verwenden die Begriffe Aufgabe und Problem synonym.

Im einfachsten Fall besteht die Beschreibung eines Planungsproblems aus Propositionen. Die Startsituation aus dem vorangegangenen Abschnitt wird dann wie folgt formuliert:

auf-A-Tisch \wedge auf-C-A \wedge auf-B-Tisch \wedge frei-B \wedge frei-C

Das Ziel ist die Formel auf-B-C \wedge auf-A-B und der Operator bewege-C-von-A-auf-den-Tisch hat die Vor- und Nachbedingung:

Vorbedingung: auf-C-A \wedge frei-C

Nachbedingung: auf-C-Tisch \wedge auf-C-A \wedge frei-A

Die Verwendung propositionaler Beschreibungen von Planungsproblemen vereinfacht die erforderlichen Algorithmen, da keine Bindung der Variablen (Unifikation, s. Kapitel 5) erforderlich ist. Dadurch kann allerdings die Anzahl der benötigten Variablen sehr stark ansteigen. Bekanntlich kann jede prädikatenlogische Repräsentation in eine propositionale umgewandelt werden, indem alle vorhanden Konstanten für alle Variablen in Prädikaten eingesetzt werden. Das Anwachsen der Anzahl der Formeln kann dadurch eingeschränkt werden, dass Konstanten wie Variablen typisiert werden, und nur noch die Konstanten eines Typs für die Variablen des selben Typs verwendet werden.

10.2 Planen als Suche im Zustandsraum

Unabhängig davon, ob eine prädikatenlogische oder präpositionale Repräsentation gewählt wird, beschreiben diese Prädikate den Zustand der Welt. Die Aktionen einer Planungsaufgabe überführen einen Zustand in einen anderen und das Ziel eines Planers ist es, eine Aktionsfolge zu finden, die den Start- in den Zielzustand überführt. Wenn wir annehmen, dass die ausgeführten Aktionen deterministisch sind und wenn alle Eigenschaften der Welt beobachtet werden können, dann kann die Zustandsfolge eindeutig bestimmt werden. Die Aufgabe eines Planers ist es dann, eine Folge von Zuständen zu finden, die vom Start- zum Zielzustand führen. Ein Planer sucht also eine Folge von Übergängen eines Weltzustandes in den nächsten, wobei der jeweils nächste Weltzustand durch die Anwendung einer Aktion bzw. ihrer Vor- und Nachbedingung erzeugt wird.

Vorwärts-Planung (Progression)

Die beschriebene Planungsaufgabe kann als ein Suchproblem (s. Kapitel 3) aufgefasst werden. Die klassische uninformierte Suche vorwärts oder rückwärts wird beim Planen auch Progression und Regression genannt. *Progression* angewendet auf unser Beispiel geht wie folgt vor (s. Algorithmus 10.1): Ausgehend von der Startsituation S_0 werden die Voraussetzungen aller Operatoren überprüft und dann eine geeignete Aktion, z.B. bewege-C-von-A-auf-den-Tisch ausgewählt. Natürlich kann es sehr viele anwendbare Operatoren geben, und nicht jede Wahl einer Aktion führt zum Erfolg. Mit Hilfe der ausgewählten Aktion berechnen wir nun die Nachfolgesituation S_1 . Anschließend wählen wir weitere Planungs-Operatoren aus, so lange, bis wir einen Plan für unser Problem gefunden haben.

Die Auswahl der nächsten Aktion ist in dem Grundalgorithmus eine „nichtdeterministische“ Auswahl, d.h. ein allwissendes System müsste eine Aktion wählen, die den weiteren Plan zum

```
progression(S, Z, A, Pfad)
1. Wenn S jedes Literal in Z erfüllt
2. Dann liefere Pfad
3. Andernfalls:
   a) Sei A eine Aktion aus A, deren Vorbedingung in S erfüllt ist.
   b) Sei S' das Ergebnis der Anwendung von A auf S.
   c) Wenn kein A gefunden wurde ...
   d) Dann Fehlschlag
   e) Andernfalls liefere progression(S', Z, A, concat(Pfad, A)).
```

Algorithmus 10.1: Ein progressiver Zustandsraum Planer.

Ziel unterstützt. Leider ist solch eine allwissende Komponente nicht verfügbar, sodass die Wahl der nächsten Aktion gerade das Hauptproblem für effiziente Suchalgorithmen ist. Wurde an einer Stelle eine „falsche“ Aktion gewählt und somit im weiteren Prozess kein gültiger Plan gefunden werden, dann liefert der Algorithmus einen Fehler, was dazu führt, dass der letzte Rekursionsschritt revidiert und an dieser Stelle eine andere Aktion gewählt wird (Backtracking). Eine blinde Vorwärts-Suche ist daher nicht besonders gut für Planungsaufgaben geeignet. Sie muss wahllos Operatoren anwenden und kann dabei auch in Zyklen laufen. Deshalb ist es von großer Bedeutung, dass die Suche durch Heuristiken gesteuert wird (s. Abschnitt 10.2.1).

Regression

Ein analoges Verfahren ist die *Regression* (s. Algorithmus 10.2) die von [45] eingeführt wurde. Der initiale Aufruf lautet

```
regression(S0, Z0, Λ, { }),
```

d.h. wir beginnen mit dem Ziel Z_0 und wählen eine Aktion aus \mathcal{A} mit solchen Merkmalen in der Nachbedingung, die im Zielzustand Z_0 erfüllt sein sollen. Nachdem eine geeignete Aktion ausgewählt wurde, müssen wir nun das Ziel Z_0 „über die Operatordefinition zurückrechnen“, so dass wir ein neues Ziel Z_1 erhalten. In diesem Fall müssen wir die allgemeinste bzw. „schwächste Vorbedingung“ der Aktion berechnen. Der Regressionsschritt wird rekursiv so lange wiederholt, bis wir einen Plan für unser Problem erhalten haben. Dieses Verfahren wird auch

```
regression(S, Z, A, Pfad)
1. Wenn S jedes Literal in Z erfüllt
2. Dann liefere Pfad
3. Andernfalls:
   a) Sei A eine Aktion aus A, deren Nachbedingungen mindestens in einem Literal mit Z übereinstimmen.
   b) Sei Z' das Ergebnis der Regression von Z über A.
   c) Wenn kein A gefunden wurde, Z' undefiniert ist, oder  $Z' \supset Z$ 
   d) Dann Fehlschlag
   e) Andernfalls liefere regression(S, Z', A, concat(A, Pfad)).
```

Algorithmus 10.2: Ein regressiver Zustandsraum Planer.

Rückwärtsplanen genannt. Regression im Rahmen von STRIPS kann durch einfache Mengen-subtraktion und Vereinigung erreicht werden.

Die drei Fälle des Abbruchkriteriums in Zeile 3(c) wollen wir noch etwas näher betrachten:

1. Wenn es keine Aktion gibt, der eine Nachbedingung enthält, die einen Beitrag zu Z leistet, dann wäre das Ergebnis der Regression Z' eine Obermenge von Z . Wenn Z' durch S erfüllt wäre, dann wäre es auch Z .
2. Wenn die Regression von Z über A undefiniert wäre, dann kann *regression* auch keinen korrekten Plan zurückliefern. Dieser Fall tritt ein, wenn eine der Nachbedingungen von A mit Z' im Widerspruch steht. Dadurch ist auch die schwächste Vorbedingung undefiniert.
3. Wenn $Z \subseteq Z'$ gilt, dann gibt es – wie schon für den 1. Fall – keinen vernünftigen Grund A zum Plan hinzuzufügen. Genau genommen subsumiert diese Bedingung den Fall 1., aber die Benutzung beider verhindert unnötige Regressionsschritte.

Regression und Progression haben zunächst einmal die selben Eigenschaften. Beide Verfahren erzeugen einen strikt linearen (total geordneten) Plan, sie sind vollständig und korrekt und sie haben die selbe theoretische Komplexität ($O(b^n)$, wobei n die Anzahl der Auswahlschritte und b die Anzahl der Wahlmöglichkeiten/Verzweigungen ist). Normalerweise weist die Regression einen geringeren Verzweigungsfaktor auf, da die Beschreibung einer Situation oft sehr viele Literale enthält, ein Ziel dagegen nur einige wenige. In diesem Fall konzentriert sich die Regression auf die Erzeugung dieser wenigen Schritte, während die Progression alle (Kombinationen von) Situationsliteralen ausprobieren würde. Manchmal ist der Anwendungsfall aber so geartet, dass es sehr viele Ziele, aber nur wenige Literale in den Situationsbeschreibungen gibt. In so einem Fall wird die Progression schneller das Ergebnis liefern.

10.2.1 Planungsheuristiken

Eines der Grundprobleme beim Lösen von Planungsaufgaben ist der große Suchraum, also die verschiedenen Möglichkeiten aus den gegebenen Aktionen ein Plan zu erzeugen, der schließlich die Startsituation in die Zielsituation überführt. Heuristiken sind allgemeine Strategien, die helfen solche Aktionen auszuwählen, die direkt zum Ziel führen, anstatt blind verschiedene Aktionen auszuprobieren. Eine ideale Heuristik würde immer die beste Aktion auswählen, sodass nie eine Entscheidung beim Planen revidiert werden müsste. Leider existieren normalerweise keine idealen Heuristiken, trotzdem können sie den Planungsaufwand erheblich reduzieren.

Eine sehr einfache Heuristik – die Mittel-Ziel-Analyse (*engl. Means-Ends Analysis*) – ist eine Heuristik um in einer bestimmten Situation den nächsten Operator auszuwählen. Dazu vergleicht der Planer den aktuellen Zustand mit dem Ziel und identifiziert die Unterschiede. Als nächster Operator wird dann derjenige ausgewählt, der die Unterschiede am stärksten reduziert. Bei Planungssystemen kann dafür eine Zuordnung zwischen Differenzklassen und Operatoren verwendet werden. Anhand der identifizierten Differenz zwischen aktuellem Zustand und Ziel kann dann einer oder mehrere Operatoren ausgewählt werden, und wenn einer dieser Operatoren in der aktuellen Situation anwendbar ist, dann kommt er als nächste Aktion in Frage. Damit kann die Anzahl der anwendbaren Operatoren in einer Situation deutlich reduziert werden.

Neben problemunabhängigen und einfachen Heuristiken wie Means-Ends Analysis, werden beim Planen vor Allem Heuristiken verwendet, die auf die spezifische Form der Operatorbeschreibungen Bezug nehmen.

```
progression(S, Z, A, Pfad, h)
1. Wenn  $S$  jedes Literal in  $Z$  erfüllt
2. Dann liefere  $Pfad$ 
3. Andernfalls:
   a) Wähle  $A$  eine Aktion aus  $\mathcal{A}$ :
      i.  $K$ : alle Aktionen, deren Voraussetzung in  $S$  erfüllt ist (Kandidaten)
      ii. für alle  $k \in K$ : berechne heuristische Funktion  $h(k)$ 
      iii. wähle  $A \leftarrow k_i$  mit minimalem heuristischen Wert  $h(k_i)$ 
   b) Sei  $S'$  das Ergebnis der Anwendung von  $A$  auf  $S$ .
   c) Wenn kein  $A$  gefunden wurde ...
   d) Dann Fehlschlag
   e) Andernfalls liefere progression(S', Z, A, concat(Pfad, A), h).
```

Algorithmus 10.3: Ein progressiver Zustandsraum Planer unter Anwendung von Heuristiken.

Algorithmus 10.3 ist eine Erweiterung des Progressionsplaners (Algorithmus 10.1). In der ersten Version des Progressionsplaners wurde die Aktionsauswahl in Schritt 3(a) nicht weiter spezifiziert, wir mussten davon ausgehen, dass die anwendbaren Aktionen blind in irgendeiner Reihenfolge angewendet werden. In Algorithmus 10.3 wurde dieser Schritt um eine heuristische Aktionsauswahl erweitert. Dabei werden zunächst Kandidaten-Aktionen K identifiziert, die anwendbar sind, d.h. deren Voraussetzungen im aktuellen Zustand S erfüllt sind. In Schritt 3(a)ii wird eine Heuristik (s. Abschnitt 3.3.1) verwendet, die den erfolgversprechendsten Kandidaten auswählt.

Dabei wird eine Heuristikfunktion $h(k)$ verwendet, die alle Kandidaten k aus K bewerten. Für den bevorzugten Kandidaten liefert $h(k)$ den kleinsten Wert. Eine Heuristikfunktion schätzt die Anzahl der nötigen Planschritte um eine bestimmte Eigenschaft des Ziels zu erreichen.

Bei der Wahl einer Heuristikfunktion haben sich solche bewährt, die die Planungsaufgabe vereinfachen, indem Details der Problembeschreibung weggelassen werden. Ein typischer Ansatz ist es, bei dieser Schätzung die negativen Effekte von Aktionen zu ignorieren. Außerdem nimmt man an, dass die einzelnen Prädikate des Ziels unabhängig voneinander erreicht werden können. Die Heuristikfunktion berechnet also die minimale Anzahl der Schritte, die aus einer Situation zur Erreichung einer Eigenschaft des Ziels unter Vernachlässigung der negativen Effekte von Aktionen erforderlich sind. Wenn das Ziel mehrere Eigenschaften gleichzeitig erfüllen muss, dann wird die Summe der Schätzungen für die einzelnen Eigenschaften des Ziels verwendet.

Als Variante dieser Heuristik kann man auch das Maximum der zu erwartenden Planschritte verwenden anstatt der Summe, insbesondere wenn die Ziele nicht unabhängig voneinander sind. Eine weitere Alternative besteht in der Betrachtung von Paaren oder Tripeln von Zielprädikaten des Ziels. Auch eine Zusammenfassung von Zuständen in einem vereinfachten Planungsproblem ist möglich (s.a. Abschnitt 10.2.2).

Der Fast-Forward Planer [24] ist ein Progressionsplaner, der eine Menge von unterschiedlichen Heuristiken sehr erfolgreich nutzt. Die Wahl der spezifischen Heuristiken, auch wenn sie an sich problemunabhängig sind, ist jedoch immer noch eine knifflige Aufgabe. Eine Heuristik kann keinen Erfolg garantieren. Deshalb benutzen erfolgreiche Planer immer eine Kombination unterschiedlicher Heuristiken.

10.2.2 Hierarchische Abstraktion

Um das Problem großer Suchräume zu lösen hat man auch die Möglichkeit die Beschreibung der Zustände zu abstrahieren, indem man Zustände zu einem umfassenderen Zustand zusammenfasst. Beispielsweise könnte man anstatt einzelner Plätze in einer Stadt nur die ganze Stadt als Ortsbeschreibung zulassen. Damit reduziert man die möglichen Zustände, macht aber die Repräsentation weniger ausdrucksstark, sodass die eigentliche Planungsaufgabe eventuell gar nicht mehr beschrieben werden kann (eine Fahrt vom Alexanderplatz zum Brandenburger Tor kann nicht mehr repräsentiert werden, wenn beide in dem Zustand „Berlin“ zusammengefasst sind). Dies kann wiederum durch zwei Maßnahmen behoben werden:

1. das abstrahierte Problem wird zunächst gelöst und aus der gefundenen Lösung wird eine heuristische Funktion abgeleitet, die dann für die Lösung des ursprünglichen Problems verwendet wird;
2. das ganze Planungsproblem wird auf mehreren Hierarchiestufen gleichzeitig ausgeführt, so dass die Operatoren des abstrakten Plans selbst wieder Planungsaufgaben darstellen. Wenn die Aufgabe beispielsweise ist nach Berlin zu fahren, wird man zuerst einen groben Plan machen, dass man beispielsweise zum Bahnhof fährt, mit dem Zug nach Berlin fährt und dort wiederum in die Stadt fährt. Die Fahrt zum Flughafen ist dann wiederum ein Planungsproblem, das man durch eine Autofahrt, das Nutzen öffentlicher Verkehrsmittel oder eines Taxis löst.

Die zweite Methode wird beim Planen mit *Hierarchical Task Networks (HTNs)* angewendet. Hier können Aktionen entweder Basis-Aktionen sein, die wie bei klassischen Planungsaufgaben durch Vor- und Nachbedingungen definiert sind. Zusätzlich gibt es komplexe Aktionen, die durch eine Menge von *Methoden* definiert sind. Methoden sind Unterpläne, die selbst wieder komplexe oder Basis-Aktionen enthalten können und für deren Ausführbarkeit Vorbedingungen spezifiziert sind.

Die Planungsaufgabe besteht dann darin, einen vollständigen Plan aus Basis-Aktionen zu finden, der das gegebene Ziel erreicht. Der Suchbaum für dieses Problem ist durch die hierarchische Struktur vorgegeben. Dadurch dass zu einer komplexen Aktion mehrere Methoden vorhanden sein können, die jeweils unterschiedliche Effekte erzeugen, ist auch hier ein Suchproblem zu lösen. Beispielsweise kann die Entscheidung mit öffentlichen Verkehrsmitteln zum Bahnhof zu fahren, bei der Rückfahrt ein Problem auslösen, wenn zu dieser Zeit keine öffentlichen Verkehrsmittel unterwegs sind. In diesem Fall wäre die Entscheidung mit dem Auto zu fahren besser, um die spätere Aktion der Rückfahrt zu ermöglichen. Trotzdem wird die Suche durch die hierarchische Definition der Aktionen erheblich erleichtert.

Ein weiterer Vorteil von hierarchischem Planen ist, dass auch Pläne erzeugt werden können, die nur aus komplexen Aktionen bestehen und die entsprechenden Methoden erst bei der Ausführung gewählt werden. Die Fahrmöglichkeiten zum Bahnhof werden beispielsweise von der aktuellen Stausituation oder Störungen im Betriebsablauf von öffentlichen Verkehrsmitteln beeinflusst. Je nachdem, welche Situation bei der Ausführung vorliegt, kann dann die entsprechende Methode gewählt werden. Somit ermöglicht hierarchisches Planen einen Übergang zwischen klassischem Planen und der eingangs erwähnten situierten Aktivität zur Planausführung.

Zu beachten ist jedoch, dass die Abstraktion der Aktionen mit einem Aufwand verbunden ist: die Methoden für jede komplexe Aktion müssen spezifiziert werden. Damit verlagert man

die Suchaufgabe zumindest teilweise wieder zurück zu den Entwicklern des Systems, was der Grundidee in der KI widerspricht, dass die Maschine eine Aufgabe möglichst eigenständig lösen soll. Deshalb versucht man zumindest teilweise komplexe Aktionen aus bereits generierten Plänen zu lernen, entweder indem man Pläne speichert und nach Bedarf als komplexe Aktionen wiederverwendet (s.a. Kapitel 9) oder indem man vorhandene Pläne als neue Operatoren abstrahiert (*explanation-based learning*).

10.3 Planen im Planraum

Bei den bisher vorgestellten Verfahren wurden die Zustände der Umgebung auch als Zustände für die Suchaufgabe verwendet. Ein Plan ist dann ein Pfad durch den Zustandsraum. Eine andere Möglichkeit ist, im Raum von möglichen Plänen zu suchen und durch schrittweise Verfeinerung einen Plan in einen anderen zu überführen.

Um eine Suche im Planraum zu ermöglichen, müssen wir die Definition von Plänen etwas breiter fassen. Während ein Plan bisher eine Abfolge von Aktionen war, deren Vorbedingungen jeweils nach Ausführung aller vorherigen Aktionen erfüllt sein müssen, wird ein Plan jetzt durch eine Menge von partiell geordneten Aktionen definiert. Dabei sind als Zustände bei der Suche auch Pläne möglich, die nicht ausführbar sind, weil Vorbedingungen einzelner Aktionen noch nicht erfüllt sind. Der endgültige Plan muss natürlich ausführbar sein und die Zielbedingungen herbeiführen.

Die erweiterte Definition von Plänen lässt nun auch als Ergebnis des Planungsprozesses partiell geordnete Pläne zu. Diese legen die Ausführungsreihenfolge nur soweit fest, dass das Ziel erreicht wird. Wenn man beispielsweise nach Berlin reisen möchte, ist es egal ob man zuerst die Abfahrtszeit des Zuges recherchiert oder zuerst seine Sachen packt. Je nach Situation können diese Aktionen in beliebiger Reihenfolge ausgeführt werden, das Ziel wird in jedem Fall erreicht. Diese Eigenschaft der Pläne bzw. Planer wird mit *least-commitment* („geringste Festlegung“) bezeichnet. Daher lassen partiell geordnete Pläne mehr Freiheit bei der Planausführung zu.

Ein weiterer Vorteil von partiellen Plänen ist die bessere Verständlichkeit für Menschen. Planungsaufgaben werden nicht immer vollautomatisch gelöst, allein schon weil es kaum möglich ist, jede notwendige Information entsprechend zu kodieren. Oft werden Planungsaufgaben interaktiv zwischen Mensch und Planungswerkzeug erledigt. Dabei kann der automatische Planer schnell Pläne generieren und mögliche Probleme aufzeigen, während der Mensch zusätzliches Wissen einfließen lassen kann. Für so eine Zusammenarbeit sind partielle Pläne sehr gut geeignet, da sie nur die notwendigen Abhängigkeiten zwischen Aktionen angeben, der Nutzer sieht sofort die Freiheiten bei der weiteren Erarbeitung des Plans. Hingegen ist bei einem vollständig geordneten Plan wie er bei der Suche im Zustandsraum generiert wird, nicht sofort ersichtlich, ob zwei Aktionen hintereinander ausgeführt werden, weil sie voneinander abhängen oder aus reinem Zufall.

Das Planen im Raum partieller Pläne wurde zuerst von Sacerdoti [36][37] mit dem System NOAH eingeführt. Zusätzlich zu der Nicht-Linearität enthält NOAH auch eine Verfeinerungsoperation für Operatoren, die es erlaubt eine Aktion in Teilaktionen zu zerlegen (s. hierarchisches Planen, Abschnitt 10.2.2).

Leider waren die Beschreibungen der frühen nichtlinearen Planer NOAH und NONLIN [42] im Detail ungenau und nicht verallgemeinerbar, so dass eine ganze Reihe von Formalisierungen [7, 22] und Nachimplementierungen (insbesondere SNLP [27]) notwendig waren, um zu einem besseren Verständnis des grundlegenden Prinzips zu gelangen. Nachfolgend stellen wir den Planer POP vor.

10.3.1 Partiell geordnete Pläne

Ein partiell geordneter Plan wird als Tripel $\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle$ dargestellt, wobei \mathcal{A} eine Menge von Aktionen ist, \mathcal{O} ist eine Menge von Ordnungsconstraints über \mathcal{A} und \mathcal{L} eine Menge von Abhängigkeiten. Ist beispielsweise $\mathcal{A} = \{A_1, A_2, A_3\}$ dann könnte \mathcal{O} die Menge $\{A_1 < A_3, A_2 < A_3\}$ sein. Damit wird ein Plan aus drei Aktionen beschrieben, wobei A_1 und A_2 vor A_3 liegen, aber zueinander ungeordnet sind. Diese Ordnungsconstraints sind *konsistent*, wenn sie sich zu einer totalen Ordnung vervollständigen lassen. Diese Konsistenz muss vom Planungsverfahren während der Planung immer wieder überprüft und eingehalten werden.

Eine Abhängigkeit⁶ $A_p \xrightarrow{Q} A_c$ hat ebenfalls drei Bestandteile:

1. eine (propositionale) Eigenschaft Q ,
2. eine Aktion A_p , die Q als Nachbedingung hat (d.h. der „Produzent“ von Q ist) und
3. eine Aktion A_c , die Q als Vorbedingung enthält (d.h. der „Konsument“ von Q ist).

Eine Abhängigkeit beschreibt im Plan den Zusammenhang, dass eine ganz bestimmte Aktion A_p das Literal Q erzeugt, welches wiederum von A_c verbraucht wird. Mit Hilfe von Abhängigkeiten kann erkannt werden, wenn eine neu eingefügte Aktion mit dem bereits bestehenden partiellen Plan in Konflikt (engl. *threat*) steht. Formal steht eine Abhängigkeit $A_p \xrightarrow{Q} A_c$ aus \mathcal{L} mit einer anderen Aktion A_t in Konflikt, wenn die folgenden Bedingungen erfüllt sind:

- $\mathcal{O} \cup \{A_p < A_t < A_c\}$ ist konsistent, und
- A_t hat $\neg Q$ als Nachbedingung.

Zur Illustration kehren wir noch einmal zu unserer Berlinreise zurück: Wenn unser Plan die Aktion „Fahrschein kaufen“ enthält, dann muss auch sichergestellt werden, dass die Fahrkarte beim „Betreten des Zuges“ tatsächlich verfügbar ist. Jede andere Aktion, die dazu führt, dass die Fahrkarte weggelegt wird, steht in Konflikt dazu und muss also verhindert werden (oder es muss dafür gesorgt werden, dass anschließend eine andere Aktion die Fahrkarte wieder aufnimmt.)

10.3.2 Planen mit partiell geordneten Plänen

Wir beschreiben nun den einfachen, regressiven Planer POP („partial order planner“), der den Raum partiell elaborierter Pläne durchsucht. POP startet mit dem leeren Plan und fügt neue Aktionen zum Plan hinzu, bis alle Vorbedingungen und Ziele durch Abhängigkeiten abgestützt sind und alle möglicherweise vorhandenen Konflikte eliminiert wurden. Ein leerer Plan wiederum besteht nur aus den beiden Pseudo-Aktionen A_0 und A_∞ , die die Start- und Zielsituation

⁶ In der englischsprachigen Literatur werden Abhängigkeiten als *causal links* bezeichnet; wir verwenden den von Hertzberg [22] eingeführten Begriff.

repräsentieren (d.h. A_0 ist eine leere Aktion ohne Vorbedingungen, deren Nachbedingungen der Startsituation entsprechen; A_∞ hat als Vorbedingungen alle Prädikate der Zielsituation und keine Nachbedingungen). Der Algorithmus 10.4 hat als Parameter drei Argumente: einen partiell elaborierten Plan, die Menge der offenen Ziele (Agenda) und die Beschreibung der möglichen Operatoren des Anwendungsbereichs. Dabei ist jeder Eintrag in der Agenda wiederum ein Paar der Form $\langle Q, A_i \rangle$ wobei Q ein Literal aus der Konjunktion der Vorbedingungen von A_i ist.

Wir illustrieren das POP-Verfahren an dem oben eingeführten Beispiel aus der Klötzenwelt (Abb. 10.1). Der initiale Aufruf von pop enthält den leeren Plan (siehe Abb. 10.2) und die Agenda mit den Literalen $\text{auf}(A, B)$ und $\text{auf}(B, C)$ als Vorbedingungen der finalen Aktion A_∞ :

$$pop(\{\{A_0, A_\infty\}, \{A_0 < A_\infty\}, \{\}, \{\langle \text{auf}(A, B), A_\infty \rangle, \langle \text{auf}(B, C), A_\infty \rangle\}, \Lambda)$$

Da die Agenda im ersten Schritt des Plans nicht leer ist, wird im zweiten Schritt ein Ziel aus der Agenda ausgewählt. Da alle Ziele von pop abgearbeitet werden müssen, hat diese

$pop(\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle, \text{Agenda}, \Lambda)$

1. **Terminierung:** Wenn die Agenda leer ist liefere $\langle \mathcal{A}, \mathcal{O}, \mathcal{L} \rangle$
2. **Wahl eines Ziels:** Wähle ein Paar $\langle Q, A_{need} \rangle$ aus der Agenda
3. **Wahl einer Aktion:** Wähle eine Aktion A_{add} aus \mathcal{A} oder (eine neue Instanz) aus Λ , die Q als Effekt hat.
wenn keine solche Aktion existiert
dann Fehlschlag
anderfalls $\mathcal{L}' = \mathcal{L} \cup A_{add} \xrightarrow{Q} A_{need}$
 $\mathcal{O}' = \mathcal{O} \cup \{A_{add} < A_{need}\}$
wenn A_{add} eine neu instanzierte Aktion
dann $\mathcal{A}' = \mathcal{A} \cup \{A_{add}\}$
 $\mathcal{O}' = \mathcal{O}' \cup \{A_0 < A_{add} < A_\infty\}$
andernfalls $\mathcal{A}' = \mathcal{A}$
4. **Anpassung der Ziele:**
 $\text{Agenda}' = \text{Agenda} \setminus \{\langle Q, A_{need} \rangle\}.$
wenn A_{add} eine neu instanzierte Aktion
dann Für alle Vorbedingungen Q_i füge $\langle Q_i, A_{add} \rangle$ zur Agenda hinzu.
5. **Schutz der Abhängigkeiten:** Für jede Aktion A_t , die in Konflikt zu einer Abhängigkeit $A_p \xrightarrow{R} A_c$ steht:
Demotion: $\mathcal{O}' = \mathcal{O}' \cup \{A_t < A_p\}$ oder
Promotion $\mathcal{O}' = \mathcal{O}' \cup \{A_c < A_t\}$
wenn \mathcal{O}' konsistent ist.
6. **Rekursion:**
liefere $pop(\langle \mathcal{A}', \mathcal{O}', \mathcal{L}' \rangle, \text{Agenda}', \Lambda)$

Algorithmus 10.4: Das POP Verfahren.



Abbildung 10.2: Initialisierung von *pop* mit dem leeren Plan.

Wahl eines Ziels nur Einfluss auf die Effizienz des Verfahrens, nicht aber auf seine Vollständigkeit. Nehmen wir an, *pop* wählt das Ziel $\text{auf}(B, C)$, d.h. A_{need} wird mit A_∞ gleichgesetzt. In Schritt 3. erfolgt eine (nichtdeterministische) Wahl einer Aktion A_{add} , die $\text{auf}(B, C)$ als Nachbedingung hat. Nehmen wir wieder an, dass *pop* diesmal die neu instanzierte Aktion $A_1 = \text{bewege-B-vom-Tisch-auf-C}$ wählt. Zunächst wird die neue Abhängigkeit

$\{A_{\text{add}} \xrightarrow{\text{auf}(B,C)} A_\infty\}$ bezüglich des Literals $\text{auf}(B, C)$ notiert und zu \mathcal{L}' hinzugefügt. Weiter wird hinzugefügt: ein neues Ordnungsconstraint $A_{\text{add}} < A_\infty$ zu \mathcal{O}' , die neue Aktion zu \mathcal{A}' und die Agenda wird um $\langle \text{auf}(B, C), A_\infty \rangle$ reduziert und um die Vorbedingungen von A_1 erweitert. Ein Schutz der Abhängigkeiten ist in diesem Schritt nicht notwendig. Das Ergebnis dieser Iteration von *pop* ist in Abb. 10.3 dargestellt und es folgt der rekursive Aufruf.

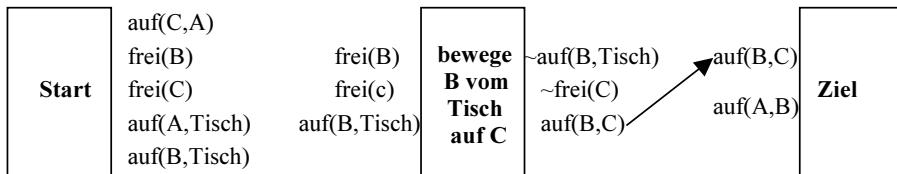


Abbildung 10.3: *pop* nach der Instanziierung von *bewege-B-vom-Tisch-auf-C*.

Nehmen wir an, dass nun im zweiten Schritt das neue Element $\text{frei}(B)$ der Agenda ausgewählt wird. In diesem Fall kann eine Abhängigkeit eingeführt werden, ohne dass eine neue Aktion in den Plan eingefügt werden muss: $\{A_0 \xrightarrow{\text{frei}(B)} A_1\}$. Das Ergebnis ist in Abb. 10.4 zu sehen.

Nehmen wir weiter an, dass beim dritten Aufruf das Originalziel $\langle \text{auf}(A, B), A_\infty \rangle$ gewählt wird und *pop* die neue Aktion $A_2 = \text{bewege-A-vom-Tisch-auf-B}$ in den Plan einfügt. Entsprechend wird eine neue Abhängigkeit etabliert und die Agenda revidiert. Bei diesem Durch-

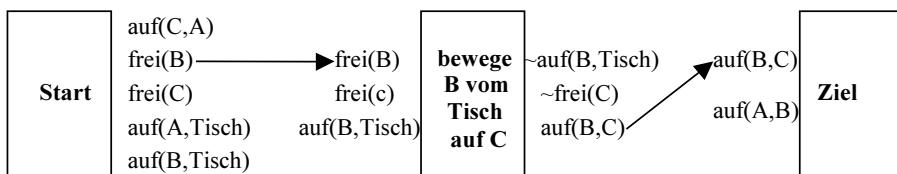


Abbildung 10.4: *pop* nach zwei Durchgängen.

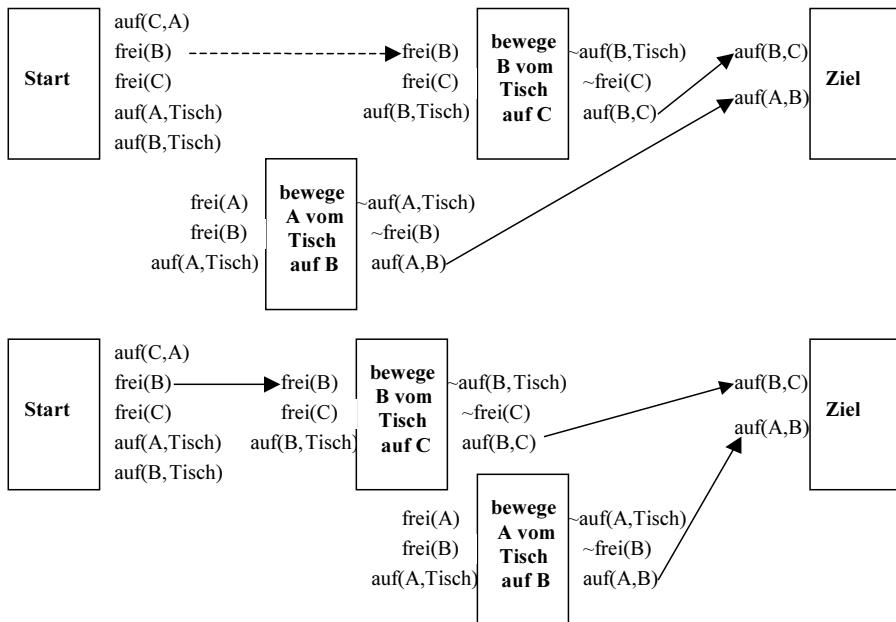


Abbildung 10.5: Entdeckung und Auflösung eines Konflikts.

lauf muss nun Schritt 5 (Schutz der Abhängigkeiten) berücksichtigt werden. A_1 und A_2 sind zueinander nicht geordnet, sie müssen nur vor der finalen und nach der initialen Aktion ausgeführt werden. Damit ist es also durchaus möglich, dass A_2 vor A_1 ausgeführt wird, wodurch ein Konflikt im Plan entstanden ist: Die neue Aktion `bewege-A-vom-Tisch-auf-B` hat als Nachbedingung $\neg\text{frei}(B)$ und es besteht die Abhängigkeit $\{A_0 \xrightarrow{\text{frei}(B)} A_1\}$!

In *pop* können Konflikte durch zwei alternative Techniken aufgelöst werden: *Promotion* und *Demotion*. Bei der Promotion wird die konfigurierende Aktion A_t nach der Abhängigkeit eingeordnet, bei der Demotion *davor*. Da in unserem Beispiel die problematische Aktion A_2 nicht vorgeordnet werden kann – A_0 ist die initiale Aktion –, wird per Promotion eine neues Ordnungsconstraint $A_1 < A_2$ zum Plan (genauer: \mathcal{O}') hinzugefügt. Abb. 10.5 stellt den Konflikt dar und zeigt seine Auflösung durch Anordnung der Operatoren (von links nach rechts).

In den weiteren Schritten müssen die restlichen Ziele der Agenda von *pop* abgearbeitet werden. Dazu muss noch eine neue Aktion in den Plan eingefügt werden, und eine Menge Abhängigkeiten sind zu etablieren. Der resultierende Plan ist in Abb. 10.6 dargestellt.

Was das Laufzeitverhalten betrifft, so zeigt sich, dass theoretisch die Komplexität von *pop* gleich der von *regression* ist. Allerdings ist der Verzweigungsfaktor durch die Wahl des Suchraumes – Suche im Raum der Pläne – normalerweise deutlich geringer ist, so dass *pop* tatsächlich schneller ist.

Das beschriebene Verfahren *pop* ist in der vorgestellten Form nur für Spielbeispiele geeignet. Insbesondere die Einschränkung auf aussagenlogische Literale macht die Formulierung von

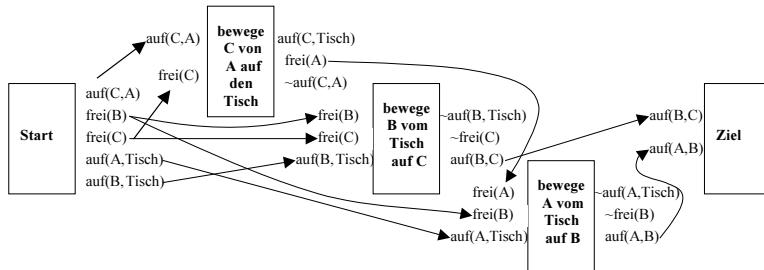


Abbildung 10.6: Finaler Plan des Beispiels aus der Klötzenwelt.

Planungsaufgaben umständlich. In der Klötzenwelt muss beispielsweise für jede Bewegung zweier Objekte relativ zueinander ein eigener Operator formuliert werden. Aus diesem Grund ist es naheliegend, Variablen bei der Beschreibung von Operatoren zuzulassen. Der Algorithmus UCPOP bietet genau diese Erweiterung. Dabei müssen zusätzlich Bindungen von Variablen beim Planen mitberücksichtigt werden. Für eine detaillierte Darstellung verweisen wir auf die Originalliteratur [33].

10.3.3 Transformationsplanen

Das Planen von partiell geordneten Plänen ist beendet sobald der Plan das Ziel erfüllt. Das ändern von bestehenden (partiell oder vollständig geordneten) Plänen kann jedoch auch nützlich sein um Pläne wiederzuverwenden.

Das Transformieren von bestehenden Plänen ist ein zentrales Element in dem HACKER System von Sussman [41]. Die Grundidee besteht darin, dass ein Agent selten eine völlig neue Aufgabe lösen muss, sondern dass ähnliche Aufgaben bereits gelöst wurden. Die Pläne, die dafür erzeugt wurden, können in einer Planbibliothek gespeichert werden. Wird der Agent mit einer neuen Aufgabe konfrontiert, sucht er aus seiner Planbibliothek Pläne, die ähnliche Aufgaben bereits erfolgreich erfüllt haben. Da die aktuelle Aufgabe jedoch von den schon erledigten abweichen kann, muss der Plan zunächst ausprobiert werden. Im Fall von HACKER wurde dies mit einem logischen Beweis erledigt, da als Umgebung die voll beobachtbare, deterministische Klötzenwelt verwendet wurde. Wenn der Beweis erfolgreich ist, kann der Plan für die aktuelle Aufgabe verwendet werden, ansonsten muss er gezielt verändert werden, ähnlich wie bei der Auflösung von Konflikten im POP Algorithmus. Dieses Vorgehen von HACKER wurde unter dem Begriff *Fallbasiertes Schließen* weiterentwickelt (s. Kapitel 9).

Planungssysteme gehen üblicherweise von einer deterministischen, statischen Welt aus, d.h. dass Aktionen genau die spezifizierte Wirkung haben werden und dass sich die Welt nur durch Aktionen des Agenten ändert. Beide Annahmen treffen nur in den seltensten Fällen zu. Sobald ein System mit einem Menschen interagiert (und dies ist in vielen Fällen der Existenzgrund für technische Systeme), kann auch der Nutzer Änderungen vornehmen oder wird anders reagieren als durch die Operatoren spezifiziert (beispielsweise halten sich Nutzer von Navigationsgeräten oft nicht an den Vorschlag des Systems). Transformationsplanen kann diese Probleme teilweise beheben, indem man Pläne zur Laufzeit oder kurz vor der Ausführung an die aktuelle Situation anpasst. Um herauszufinden, ob ein Plan in der aktuellen Situation angemessen ist, kann man sich moderner, auf physikalischen Modellen beruhender Simulationen bedienen [25]. Durch mehrmaliges Ausführen eines Plans in der Simulation erhält man eine Vorhersage über

die Wahrscheinlichkeit, dass der Plan das gewünschte Ziel erreichen wird. Sollte dies nicht der Fall sein, wird der Plan transformiert und wiederum getestet [30].

Zusätzlich zu der Vorgabe ein bestimmtes Ziel zu erreichen, kann Transformationsplanen auch effizientere Pläne erzeugen. Obwohl es Planungsansätze gibt, die den Faktor Zeit explizit berücksichtigen, so gehen diese davon aus, dass die benötigte Zeit für eine Aktion immer dieselbe ist. Je nach Situation kann die Ausführungsdauer variieren (beispielsweise hängt die Dauer einer Taxifahrt von der aktuellen Situation auf den Straßen ab). Durch das Ausprobieren des Plans mit einer Simulation, in der die aktuelle Situation nachgestellt wird, erlangt man direkt Aussagen über die erwartete Dauer der Ausführung. Durch Plantransformation kann dann nach einem effizienteren Plan gesucht werden [30].

10.4 Graphbasiertes Planen

Graphbasiertes Planen ist eine interessante Kombination aus der Suche im Zustandsraum und der Suche im Planraum. Durch eine spezielle Repräsentation der Planungsaufgabe und des Suchbaums (also der theoretisch möglichen Pläne) durch einen Graphen wird ein effizienter Algorithmus ermöglicht, der sowohl Elemente aus heuristischer Suche im Zustandsraum als auch aus dem Planen mit partiell geordneten Plänen beinhaltet.

Das Graphplan-Verfahren von Blum und Furst [4][5] hat zu wichtigen Fortschritten geführt. Dies hat vor allem zwei Gründe:

- Graphplan ist ein einfacher und eleganter Algorithmus, der es erlaubt schnelle Planer zu implementieren – manchmal um Größenordnungen schneller als die Planer der vorangegangenen Generation wie SNLP oder UCPOP.
- Die von Graphplan benutzte Repräsentation eines Planungsproblems, ist die beste Transformation des Planungsproblems in ein propositionales Constraint-Satisfaction-Problem (CSP oder SAT abgekürzt; siehe Kapitel 7).

Graphplan wechselt alternierend zwischen zwei Phasen: Der *Expansion des Planungsgraphen* und der *Extraktion der Lösung*. Die Expansionsphase erweitert den Planungsgraphen progressiv („vorwärts in der Zeit“) bis eine notwendige aber nicht hinreichende Bedingung für die Existenz eines Lösungs-Plans erfüllt ist. Anschließend sucht die Extraktionsphase den Planungsgraphen rückwärts nach einem Plan ab, der die Planungsaufgabe löst. Wenn keine Lösung gefunden wird, beginnt der Zyklus von vorne, indem der Planungsgraph weiter expandiert wird.

Expansion des Planungsgraphen

Der Planungsgraph besteht aus zwei Sorten von Knoten, Propositionsknoten und Aktionsknoten, die in Schichten angeordnet sind. Wenn die Schichten bei 0 beginnend nummeriert werden, dann enthalten die Schichten mit geraden Ordnungsnummern Propositionsknoten (z.B. mit Grundliteralen). Die Sicht 0 enthält genau die Literale, die durch die Startsituation des Planungsproblems spezifiziert wurden.

Die Knoten der ungeraden Sichten repräsentieren Aktionen. Es gibt einen Knoten für jede Aktion, deren Vorbedingungen in der vorhergehenden Schicht vorhanden sind. Propositionsknoten

und Aktionsknoten werden durch Kanten miteinander verbunden, wenn die entsprechenden Literale Vorbedingungen der Aktionen sind. Ebenso sind Aktionsknoten mit Propositionsknoten durch Kanten verbunden, wenn die zugeordneten Literale Nachbedingungen der Aktionen sind. Abb. 10.7 zeigt die schematische Darstellung eines Planungsgraphen, wobei Propositionsknoten durch Kreise und Aktionsknoten durch Rechtecke dargestellt wurden.

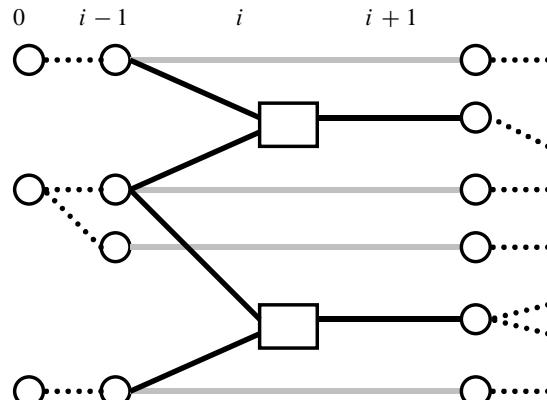


Abbildung 10.7: Repräsentation eines Planungsgraphen.

Zusätzlich zu den explizit repräsentierten Aktionen, enthält der Planungsgraph sog. Persistenzaktionen. Für jedes Literal, das in Schicht i existiert, gibt es auf der Schicht $i + 1$ eine Persistenzaktion: Jedes Literal ist auch weiterhin gültig, wenn weiter nichts damit geschieht.

Ein Planungsgraph beinhaltet mögliche („parallele“) Aktionen auf jeder Ebene. D.h., ein Planungsgraph mit k Schichten von Aktionsknoten kann einen Plan mit mehr als k Aktionen repräsentieren. Umgekehrt heißt das aber nicht, dass zwei Aktionen der selben Schicht gleichzeitig ausgeführt werden können. Knoten der selben Ebene können sich gegenseitig ausschließen. Wir definieren die „mutex“ (engl. *mutual exclusive*) Relation wie folgt (siehe Abb. 10.8):

Definition: mutex Relation

- Zwei Aktionen der Schicht i des Planungsgraphen sind mutex, wenn eine der folgenden Bedingungen gilt:
 - *Inkonsistente Nachbedingungen*: Die Nachbedingung einer Aktion ist die Negation der Nachbedingung einer anderen (Aktion).
 - *Interferenz*: Die Nachbedingung einer Aktion löscht (ist Negation von) die Vorbedingung einer anderen.
 - *Konkurrierende Vorbedingungen*: Zwei Aktionen besitzen Vorbedingungen, die in der Schicht $i - 1$ mutex sind.
- Zwei Literale einer Schicht i sind mutex,
 - wenn die Literale komplementär (ein Literal ist die Negation des anderen) sind, oder
 - alle Aktionen, die für die Erzeugung der Literale in Frage kommen, paarweise mutex sind (*Inkonsistente Vorbedingung*).

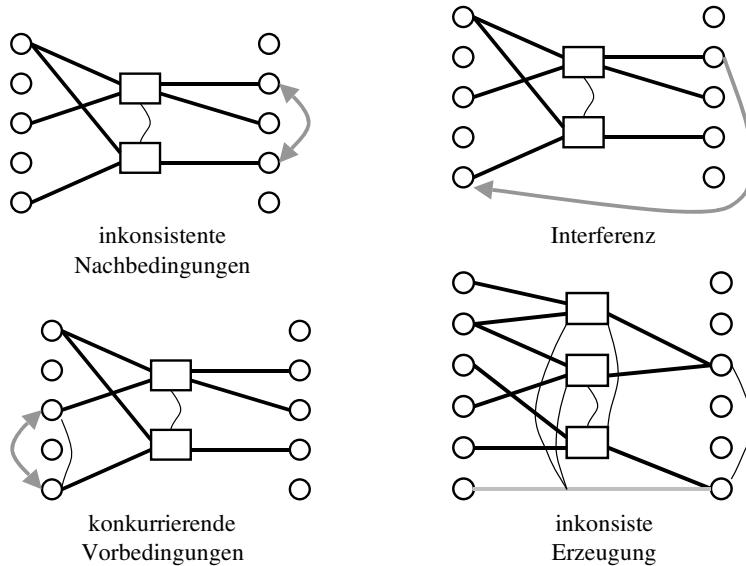


Abbildung 10.8: Graphische Darstellung der mutex Relation.

Betrachten wir als Beispiel eine romantische Aufgabenstellung: Ziel ist es, für den schlafenden Schatz ein Überraschungessen zu kochen. Vier Aktionen sind einzuplanen: wischen, (staub-)saugen, kochen und bluva (die gekauften Blumen in eine Vase stellen). Da es im Zimmer schmutzig ist, muss entweder gesaugt (Aktion saugen) oder mit dem feuchten Lappen gewischt werden (Aktion wischen). Beide Aktionen habe die gewünschte Wirkung, nämlich dass es anschließend sauber im Zimmer ist. Saugen macht allerdings Lärm (\neg Ruhe), so dass sie aufwachen könnte und die Überraschung misslingt. Wischen hat schmutzige Hände (\neg sHand) zur Folge, was wiederum das anschließende Kochen verbietet. Die Aktion des Blumen in die Vase Stellens hat als Vorbedingung Ruhe, da es ja eine Überraschung werden soll.

Wir fassen die Spezifikation dieser Domäne zusammen:

Startsituation: Schmutz \wedge sHand \wedge Ruhe

Dinner \wedge Blumen \wedge \neg Schmutz

kochen Vorbedingung: sHand

Nachbedingung: Dinner

bluva Vorbedingung: Ruhe

Nachbedingung: Blumen

wischen Vorbedingung: —

Nachbedingung: \neg Schmutz \wedge \neg sHand

saugen Vorbedingung: —

Nachbedingung: \neg Schmutz \wedge \neg Ruhe

Abbildung 10.9 zeigt den Planungsgraphen nach der ersten Expansion mit allen anwendbaren Aktionen ausgehend von der Startsituation. Dabei ist wischen beispielsweise mutex mit der

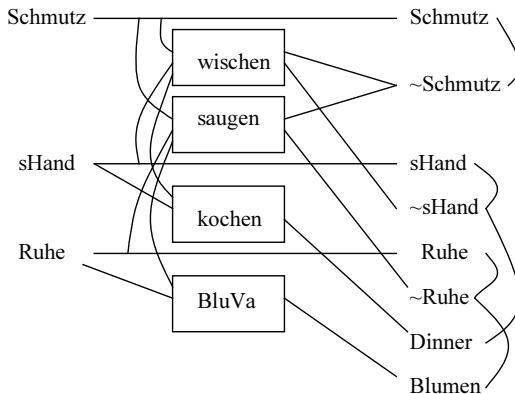


Abbildung 10.9: Der Planungsgraph nach der initialen Expansionsphase. Aktionen wurden als Kästchen dargestellt, die Zugehörigkeit zu einer Vor- oder Nachbedingung als gerade Linie, eine Persistenzaktion durch eine durchgehende waagrechte gerade Linie und mutex Relationen durch gekrümmte Linien.

Persistenz von Schmutz, aufgrund inkonsistenter Nachbedingungen. Saugen ist mutex mit bluva wegen Interferenz, denn saugen stört die Ruhe. Auf der zweiten propositionalen Ebene ist ist \neg Ruhe mutex mit Blumen aufgrund von inkonsistenten Vorbedingungen. Da (1) alle Literale der Zielsetzung ($\text{Dinner} \wedge \text{Blumen} \wedge \neg\text{Schmutz}$) auf der zweiten propositionalen Ebene im Planungsgraphen enthalten sind, und da (2) keines der Zielliterale mutex mit einem anderen ist, gibt es eine Chance, dass bereits ein Plan existiert. Dies ist eine notwendige aber nicht hinreichend Bedingung dafür, dass im Planungsgraphen eine Lösung für das Planungsproblem enthalten ist. Deshalb wechselt Graphplan in die Phase der Lösungsextraktion.

Lösungsextraktion

Die Lösungsextraktion untersucht jedes der Literale der Zielspezifikation. Wenn der Planungsgraph bis zur Schicht i expandiert wurde, dann selektiert die Lösungsextraktion für jedes Zielliteral eine Aktion a der Schicht $i - 1$. Diese Selektion muss evtl. zurückgenommen (Backtracking) werden: Wenn es mehr als eine Aktion gibt, die das Zielliteral erzeugen kann, müssen alle Aktionen a der Reihe nach berücksichtigt werden, um Vollständigkeit zu gewährleisten. Wenn a mit allen anderen aus der selben Schicht ausgewählten Aktionen konsistent ist (d.h. nicht mutex), dann wird das nächste Zielliteral untersucht. Wenn keine geeignete Wahl für a gefunden wird, muss Graphplan backtrackingen.

Nachdem Graphplan eine konsistente Menge von Aktionen aus Schicht $i - 1$ gefunden hat, wird rekursiv die nächste Schicht $i - 2$ untersucht: Für die Vereinigung aller Vorbedingungen der Aktionen aus Schicht $i - 1$ muss ein Plan gefunden werden. Die Rekursion terminiert bei Ebene 0; wenn dort alle Literale existieren, dann wurde eine Lösung gefunden. Wenn das Backtracking über alle möglichen Kombinationen von Aktionen fehlschlägt, dann muss der Planungsgraph weiter expandiert werden.

In unserem romantischen Beispiel werden aus der Schicht 2 drei Literale untersucht. $\neg\text{Schmutz}$ wird durch wischen und durch saugen erzeugt, Dinner durch Kochen und Blumen durch bluva. Folglich gibt es zwei Kandidaten, die als Pläne in Frage kommen: {wischen,kochen,bluva}

und $\{\text{saugen}, \text{kochen}, \text{bluva}\}$. Leider ist keiner dieser Kandidaten als Plan geeignet, da wischen mit kochen mutex ist und saugen mit bluva. Damit schlägt die Lösungsextraktion fehl und der Plangraph muss um zwei Schichten erweitert werden (siehe Abb. 10.10).

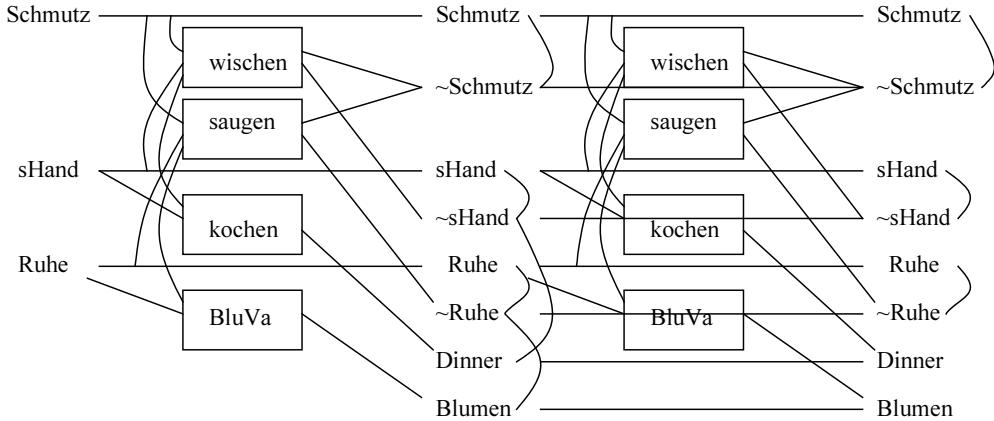


Abbildung 10.10: Der Planungsgraph nach der zweiten Expansionsphase.

Der Unterschied zwischen den Schichten 2 und 4 ist bemerkenswert! Obwohl es keine neuen Literale in Schicht 4 gibt, gibt es dort weniger mutex Relationen. Beispielsweise gibt es keine mutex Relation mehr zwischen Dinner und $\neg s\text{Hand}$. Der Grund dafür ist Schicht 3, die wesentlich mehr Aktionen – einschließlich aller Persistenzaktionen enthält. Jedes der Literale wird durch zusätzliche Aktionen erzeugt, die bei der Lösungsextraktion heran gezogen werden können:

- $\neg \text{Schmutz}$ wird durch wischen, saugen und durch eine Persistenzaktion erzeugt.
- Dinner durch kochen und eine Persistenzaktion.
- Blumen durch bluva und eine Persistenzaktion.

Von der Lösungsextraktion können nun mehrere Kombinationen von Aktionen gefunden werden, die einen konsistenten Plan ergeben: Wir wählen als Erzeuger für $\neg \text{Schmutz}$ wischen, für Dinner die Persistenzaktion und für Blumen bluwa. In Schicht 3 sind alle diese Aktionen konsistent (nicht mutex). Damit müssen die Literale Dinner (als Vorbedingung der Persistenzaktion) und Ruhe (Vorbedingung von bluwa) in Schicht 2 betrachtet werden. Die Lösungsextraktion wird rekursiv aufgerufen und wählt kochen als Erzeuger von Dinner und die Persistenzaktion zur Erzeugung von Ruhe aus. Beide Aktionen sind wiederum nicht mutex, so dass diese Auswahl konsistent ist. Außerdem sind die Vorbedingung beider Aktionen in Schicht 0. Die getroffene Auswahl ist also konsistent und ein Plan wurde gefunden. Abbildung 10.11 stellt diese Lösung dar.

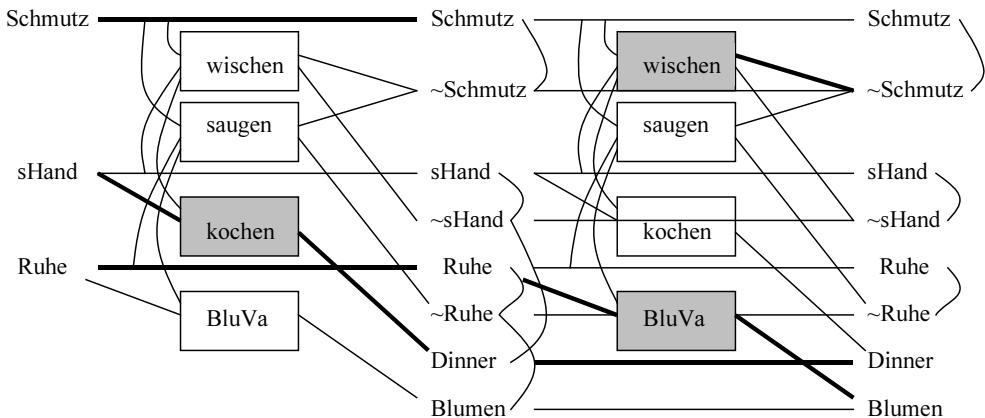


Abbildung 10.11: Eine von vier Lösungen, die durch die Lösungsextraktion gefunden wurde. Planschritte sind fett bzw. grau hervorgehoben.

10.5 Erweiterungen und Ausblick

Mit der Diskussion von STRIPS, POP und Graphplan wurde ein kurzer Überblick über die wichtigsten Verfahren der Generierung von Handlungsplänen gegeben. Dabei wurde unter anderem angenommen, dass alle Situationen vollständig bekannt sind, dass die Effekte von Operatoren immer entsprechend ihrer Definition eintreten, dass sich die Welt nur durch Aktionen des Plans ändert und dass weder der Ausführungszeitpunkt noch die Dauer von Aktionen eine Rolle spielen. In realistischen Anwendungen sind diese Annahmen normalerweise nicht erfüllt.

10.5.1 Erweiterungen des Planungsproblems

Wie bereits in den Anfängen der Behandlung des Themas KI-Planen (beispielsweise mit STRIPS, [13]) und in Planungsanwendungen auch immer wieder später (beispielsweise in SIPE [49, Kapitel 11]) klargeworden ist, treten gravierende Probleme bei der Verwendung erzeugter Pläne dadurch auf, dass die Welt sich bei der Ausführung eines Plans nicht so verhält, wie bei der Erzeugung des Plans vorausgesetzt war. Dass das so ist, überrascht nicht, denn derselbe Effekt tritt bei jeder Form „natürlicher“ Planung entsprechend auf; dass die meisten klassischen KI-Arbeiten zur Planung diese Tatsache ausgeblendet haben, liegt eher an Gründen der Forschungsmethode (Planen *an sich* ist schon kompliziert genug und eigene Arbeit wert) als daran, dass dieser Effekt nicht gesehen worden wäre.

Spätestens seit Mitte der 80er Jahre ist das Problem der Unsicherheit im Planen jedoch stärker ins Bewusstsein gerückt, und es wurden Ansätze entwickelt, dieses Problem anzupacken.

Zum Einen können Fakten oder Zusammenhänge der Startsituation und aller weiteren, daraus sich ergebenden Situationen unsicher sein in dem Sinne, dass ihr Wahrheitswert oder ihre exakte Ausprägung nicht bekannt ist oder Information darüber nur vorläufigen Charakter hat. Beispielsweise mag die Position oder der Zustand eines Objektes nur innerhalb gewisser Toleranzgrenzen gegeben sein.

Unsicherheit über die Operatoren im Allgemeinen bzw. über Aktionen im Plan heißt Unsicherheit über deren Wirkungen im Plan. Zu dieser Unsicherheit kann es kommen, weil Auswirkungen von Operatoren nicht oder nicht exakt bekannt sind. Das ist beispielsweise der Fall bei „pathologischen“ Operatoren wie dem Werfen eines Würfels, aber im Detail auch bei Fügeoperatoren wie dem ungeführten Aneinandersetzen von Objekten, wo Lagetoleranzen auftreten können. Im Detail gibt es drei Klassen dieser Art von Unsicherheit:

1. Für die *Nachbedingung* eines Operators kann dieselbe Form von Unsicherheit zutreffen wie für die Startsituation (z.B. ist die Nachbedingungen des Operators „würfeln“ offensichtlich disjunktiv).
2. *Ausführungsfehler* werden im Allgemeinen erst zur Ausführungszeit eines Plans relevant; andererseits sollten sie aber auch schon während der *Planung* berücksichtigt werden können, z.B. wenn man weiß, dass Aktionen normalerweise nur innerhalb gewisser Toleranzen korrekt ausgeführt werden können. Werden diese Toleranzen in die Nachbedingung des entsprechenden Operators integriert, dann wird diese zweite Form der Unsicherheit durch die erste subsumiert.
3. Unter der *Kontextabhängigkeit* von Operatoren verstehen wir seine Abhängigkeit von den bei seiner Ausführung gültigen Gegebenheiten. Eine weitere Form der Kontextabhängigkeit tritt bei der Verfeinerung solcher Operatoren auf. Zum Beispiel Die Konsequenz einer solchen Unsicherheit ist etwa ein Plan mit Aktionsalternativen oder – im ungünstigsten Fall – ein Ausführungsfehler.

Auch die *Vorbedingung* eines Operators mag nicht genau bekannt sein; die Auswirkung davon ist allerdings mit der Unsicherheit über die Nachbedingung insofern vergleichbar, als auch dann, wenn eine Aktion unvorhergesehenerweise nicht ausgeführt werden konnte, anschließend nicht alles das gilt, was geplant war.

Da wir, wie gesagt, nicht auf die *Gründe* eingehen wollen, warum ein Faktum in der Welt nicht so ist wie gedacht, subsumieren Unsicherheit in der Bereichs- und Operatorbeschreibung gemeinsam eine intuitiv wichtige Klasse von Unsicherheiten: das Auftreten „spontaner“ Ereignisse, wobei sich „spontan“ auf die Sicht des Planers bezieht, der aus seiner Information über den Bereich das Ereignis nicht vorhersehen konnte.

Gewöhnlich setzt man beim Planen voraus, dass bei der Ausführung des fertigen Plans alle Aktionen *unbedingt* ausgeführt werden. Demgegenüber enthalten *bedingte Pläne* (vgl. [19, 26, 34, 46]) boolesche Ausdrücke, die bei der Planausführung getestet werden, wobei entsprechend dem Testergebnis in Unterpläne verzweigt wird. Aufgrund der Überprüfungen während der Planausführung kann man für Situationen planen, in denen nicht die gesamte Information, die man zur erfolgreichen Planausführung (aber erst dann!) benötigt, zum Zeitpunkt der Plangenerierung verfügbar ist. Der Planer muss aber zumindest die Bedingungen angeben können, die zur Ausführungszeit abzuprüfen sind: er muss zur Planungszeit Wissen über mögliche oder zulässige Werte von Variablen besitzen, die in den Test einfließen. Weiter müssen die Planungsalternativen im Prinzip bekannt sein. Praktikabel scheinen die bedingten Pläne dann zu sein, wenn nur wenige Planungsalternativen berücksichtigt werden müssen.

Neben den bedingten Plänen wurden eine Reihe von Methoden entwickelt, die wir hier unter dem Begriff des reaktiven Planens zusammenfassen. Solche Ansätze sind beispielsweise Schoppers' [39][40] *universal plans*, das PRS [14, 15], Agres und Chapmans [1, 8] Systeme *Pengi* und

Blockhead, Nilssons [32] *action networks* oder Drummonds [11] *situated control rules*. Die Sprache CRAM-PL⁷ [3] ist eine Neuimplementierung der Reactive Plan Language (RPL) von McDermott [29].

Ein reaktiver Plan ist nicht *ein* Plan, den man zur Lösung *eines* Problems verfolgt, sondern er beschreibt Handlungswissen über den *gesamten* Anwendungsbereich, mit dem unerwartete Änderungen in der Welt flexibel und normalerweise ohne Neuplanung abgefangen werden können. Unter Umständen läuft die Steuerung der Handlungen eines Akteurs zur Lösung eines Problems darauf hinaus, dass man eben *nicht* plant, sondern dass der Akteur sich so verhält, wie es der vorgefertigte reaktive Plan vorgibt. Das ist in Reinform der Fall bei *universal plans*; hat sich vor der Ausführung einer Aktion die Welt in einer „eigentlich“ nicht vorhergesehenen oder vorhersehbaren Weise geändert, stört das die Handlungen des Akteurs überhaupt nicht, da die Auswahl der jeweils nächsten Aktion stets nur auf die aktuellen Gegebenheiten eingeht. Eine grundlegende Kritik an *universal plans* findet sich in [17].

Die Verwendung von reaktiven Plänen und klassischer Planung widersprechen sich nicht, sondern ergänzen und überlappen sich. Ein reaktiver Plan enthält gleichsam kompiliertes Handlungswissen über einen Anwendungsbereich. Dadurch kann vorhandene Unsicherheit im Bereich oder den Operatoren bei der Ausführung abgefangen werden, indem jeweils die Aktion als nächste ausgeführt wird, die in der aktuellen Situation unter den aktuellen Zielen angemessen ist. Bemerkt man bei der Ausführung oder in einer der Ausführung vorhergehenden Planungsphase, dass der vorhandene reaktive Plan nichts aussagt, fällt man auf klassische Planung zurück. Entsprechende Arbeiten sind beispielsweise [10, 44]; letztlich gehen alle Arbeiten in dieser Richtung zurück auf den Aufbau und die Ausführung der sogenannten Dreieckstafeln in STRIPS [12].

10.5.2 Was haben wir ausgelassen?

In diesem schlaglichtartigen Überblick können wir die meisten Bereiche des Gebiets Planen nur kurz streifen. Hier wollen wir zumindest darauf hinweisen, dass es noch eine Menge spannender Themen gibt, und im Einzelfall können wir einen Einstieg in die Literatur geben.

Historisches. Das Planen hat in der KI eine lange Geschichte, und für das Verständnis besonders des klassischen Planens ist es hilfreich, historische Arbeiten zu studieren. Einflussreich war neben den schon genannten Arbeiten von McCarthy [28] beispielsweise der General Problem Solver (GPS) von Newell und Simon [31].

„Anytime planning“. Klassische Planungsprogramme liefern einen komplett bis in minimale Details ausgearbeiteten Plan oder gar nichts. Das Ziel bei einem *anytime planner* ist, dass er in der Lage sein soll, nach einer begrenzten Rechenzeit bereits einen „vernünftigen“ Plan abzuliefern. Kommt in dem Plan quantitative Zeitinformation vor, soll der Plan darüber hinaus für die frühesten Termine am weitesten ausgearbeitet sein. Eine zentrale Arbeit zu diesem Thema ist [9]. [51] gibt einen allgemeinen Überblick über die Verwendung von Anytime Algorithmen für Intelligente Systeme.

Interaktives Planen. Die vorgestellten Planungsalgorithmen versuchen eine gegebene Aufgabe vollständig zu lösen. In der Praxis ist es jedoch fast unmöglich das gesamte Problem

⁷ http://ros.org/wiki/cram_language.

korrekt zu modellieren, Menschen haben meist eine Menge implizites Wissen, das ebenfalls berücksichtigt werden sollte. Interaktive Planungssysteme dienen als Werkzeug für Planungsaufgaben, indem sie zusammen mit einem Nutzer Aufgaben lösen. Ein klassisches Beispiel für ein interaktives Planungssystem ist SIPE (System for Interactive Planning and Execution) [50]. Meist werden hierfür partial-order planners eingesetzt, oft in Verbindung mit hierarchischer Abstraktion [38]. Interaktives Planen wird unter anderem bei der Planung von Weltraummissionen häufig eingesetzt [6].

Wahrscheinlichkeitsbasierte Modelle. Im nichtklassischen Planen geht man zunehmend davon aus, dass nur mit einer gewissen Wahrscheinlichkeit Fakten zutreffen oder Effekte einer Aktion eintreten. Entsprechend bietet es sich an, Verfahren aus anderen Gebieten ins Planen zu übertragen, wie klassische Entscheidungstheorie oder Kontrolltheorie. Eine wichtige Rolle spielen Markow-Prozesse, so wie sie z.B. in Kapitel 8 eingeführt werden.

Lernen. Es gibt eine Reihe von Arbeiten, Planen mit Lernen in dem Sinne zu verbinden, dass durch das Planen und die Planausführen im Anwendungsbereich das Wissen über den Bereich verfeinert wird, wodurch wiederum das Planen besser werden kann. Eine einflussreiche Arbeit hierzu ist [20]; einen Überblick bieten Gratch und DeJong [18]. Siehe dazu auch das Kapitel 12 über Maschinelles Lernen.

10.6 Literatur und Verweise

Eine umfassende Darstellung von Planungsmethoden bietet das Lehrbuch von [16]. Kürzere Übersichten gibt es in allgemeinen Lehrbüchern zur künstlichen Intelligenz, beispielsweise dem Buch von Russell und Norvig [35].

Die International Conference on AI Planning and Scheduling (ICAPS)⁸ findet jährlich statt und ist die einschlägige internationale Konferenz zum Thema Planen. Im deutschsprachigen Raum organisiert die Fachgruppe Planen und Konfigurieren (PUK)⁹ der Gesellschaft für Informatik jedes Jahr einen Workshop, der jeweils in Verbindung mit einer größeren Konferenz zum Thema Planen oder künstliche Intelligenz allgemein abgehalten wird. Als eines der Kerngebiete der KI findet man auf allen KI Konferenzen und in allen wichtigen Zeitschriften einschlägige Beiträge zum Thema Planen.

Literaturverzeichnis

- [1] Agre, P. und Chapman, D. (1987). Pengi: An implementation of a theory of activity. In *Proc. AAAI-87*, page 268.
- [2] Allen, J., Hendler, J., und Tate, A. (1990). *Readings in Planning*. Morgan Kaufmann, San Mateo, CA.
- [3] Beetz, M., Mösenlechner, L., und Tenorth, M. (2010). CRAM – A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1012–1017, Taipei, Taiwan.

⁸ <http://www.icaps-conference.org/>

⁹ <http://www.puk-workshop.de/fachgruppe.html>

- [4] Blum, A. und Furst, M. (1995). Fast planning through planning graph analysis. In *Proceedings of AAAI-95*, pages 1636–1642.
- [5] Blum, A. und Furst, M. (1997). Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300.
- [6] Bresina, J. L. und Morris, P. H. (2007). Mixed-initiative planning in space mission operations. *AI Magazine*, 28(2).
- [7] Chapman, D. (1987). Planning for conjunctive goals. *Artificial Intelligence*, 32(3):333–377. Nachgedruckt in: [2].
- [8] Chapman, D. (1989). Penguins can make cake. *AI Magazine*, 10(4):45.
- [9] Dean, T. L. und Boddy, M. (1988). An analysis of time dependent planning. In *Proc. AAAI-88*, pages 49–54.
- [10] Downs, J. und Reichgelt, H. (1991). Integrating classical and reactive planning within an architecture for autonomous agents. In [23], pages 13–26.
- [11] Drummond, M. (1989). Situated control rules. In *Proc. 1st Int. Conf. on Principles of Knowledge Representation and Reasoning.*, pages 103–113, San Mateo, CA. Morgan Kaufmann.
- [12] Fikes, R., Hart, P., und Nilsson, N. J. (1972). Learning and executing generalized robot plans. *Artificial Intelligence*, 3(4):251–288. Nachgedruckt in: [2].
- [13] Fikes, R. E. und Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 5(2):189–208. Nachgedruckt in: [2].
- [14] Georgeff, M. P. und Lansky, A. (1987). Reactive reasoning and planning. In *Proc. AAAI-87*, page 677.
- [15] Georgeff, M. P., Lansky, A. L., und Schoppers, M. J. (1987). Reasoning and planning in dynamic domains: An experiment with a mobile robot. Tech. Note 380, SRI.
- [16] Ghallab, M., Nau, D., und Traverso, P. (2004). *Automated planning: theory and practice*. Morgan Kaufmann Publishers, Elsevier.
- [17] Ginsberg, M. L. (1989). Universal planning: An (almost) universally bad idea. *AI Magazine*, 10(4):40.
- [18] Gratch, J. und DeJong, G. (1992). A framwork of simplifications in learning to plan. In [21], pages 78–87.
- [19] Green, C. (1969). Application of theorem proving to problem solving. In *Proc. IJCAI-69*, pages 219–239. Morgan Kaufmann Publishers. Nachgedruckt in: [2].
- [20] Hammond, K. (1989). *Case Based Planning. Viewing Planning as a Memory Task*. Perspectives in Artificial Intelligence Series. Academic Press, Boston.
- [21] Hendler, J., editor (1992). *Artificial Intelligence Planning Systems: Proc. of the First International Conference (AIPS92)*. Morgan Kaufmann.
- [22] Hertzberg, J. (1989). *Planen. Einführung in die Planerstellungsmethoden der Künstlichen Intelligenz*. B.I.-Wissenschaftsverlag, Mannheim u.a.
- [23] Hertzberg, J., editor (1991). *European Workshop on Planning. EWSP'91*, number 522 in LNAI, Berlin. Sankt Augustin, FRG, Springer.
- [24] Hoffmann, J. und Nebel, B. (2001). The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302.
- [25] Kunze, L., Dolha, M. E., Guzman, E., und Beetz, M. (2011). Simulation-based temporal projection of everyday robot object manipulation. In Yolum, Tumer, Stone, und Sonnen-

- berg, editors, *Proc. of the 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*. IFAAMAS.
- [26] Manna, Z. und Waldinger, R. (1986). A theory of plans. In *Proc. of the Workshop on Reasoning about Actions and Plans*, San Mateo, CA. Morgan Kaufmann.
 - [27] McAllester, D. und Rosenblitt, D. (1991). Systematic nonlinear planning. In *Proc. AAAI-91*, pages 634–639.
 - [28] McCarthy, J. und Hayes, P. J. (1969). Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4. Nachgedruckt in: [2].
 - [29] McDermott, D. (1993). A reactive plan language. Technical report, Yale University, Computer Science Dept.
 - [30] Müller, A., Kirsch, A., und Beetz, M. (2007). Transformational planning for everyday activity. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS'07)*, pages 248–255, Providence, USA.
 - [31] Newell, A. und Simon, H. A. (1963). GPS, a program that simulates human thought. In Feigenbaum, E. A. und Feldman, J., editors, *Computers and Thought*, pages 279–293. R. Oldenbourg KG. Nachgedruckt in: [2].
 - [32] Nilsson, N. J. (1989). Action networks. In [43], page 37.
 - [33] Penberthy, J. S. und Weld, D. (1992). UCPOP: A Sound, Complete, Partial-Order Planner for ADL. In *Third International Conference on Knowledge Representation and Reasoning (KR-92)*, Cambridge, MA.
 - [34] Peot, M. und Smith, D. (1992). Conditional nonlinear planning. In [21], pages 189–197.
 - [35] Russell, S. J. und Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Pearson Education, third edition.
 - [36] Sacerdoti, E. D. (1975). The nonlinear nature of plans. In *Proc. IJCAI-75*, pages 206–214. Nachgedruckt in: [2].
 - [37] Sacerdoti, E. D. (1977). *A Structure for Plans and Behavior*. American Elsevier, New York.
 - [38] Schattenberg, B., Bidot, J., Geßler, S., und Biundo, S. (2009). A Framework For Interactive Hybrid Planning. In Mertsching, B., Hund, M., und Aziz, Z., editors, *KI 2008: Advances in Artificial Intelligence, Proceedings of the 32nd Annual German Conference on AI*, volume 5803 of *Lecture Notes in Artificial Intelligence*, pages 17–24, Paderborn, Germany. Springer.
 - [39] Schoppers, M. J. (1987). Universal plans for reactive robots in unpredictable environments. In *Proc. IJCAI-87*, page 1039.
 - [40] Schoppers, M. J. (1989). *Representation and Automatic Synthesis of Reaction Plans*. PhD thesis, Univ. of Illinois, Dept. of Comp. Sci.
 - [41] Sussman, G. J. (1974). The Virtuous Nature of Bugs. In Allen, J., Hendler, J., und Tate, A., editors, *Readings in Planning*, pages 111–117. Morgan Kaufmann, San Mateo, CA.
 - [42] Tate, A. (1977). Generating project networks. In *Proc. IJCAI-77*, pages 888–893. Morgan Kaufmann Publishers. Nachgedruckt in: [2].
 - [43] Tenenberg, J., Weber, J., und Allen, J. (1989). Proc. from the Rochester Planning Workshop: From Formal Systems to Practical Systems. Technical Report 284, University of Rochester, Computer Science.

- [44] Thiébaux, S. und Hertzberg, J. (1992). A semi-reactive planner based on a possible models action formalization. In [21], pages 228–235.
- [45] Waldinger, R. (1977). Achieving several goals simultaneously. In *Machine Intelligence 8*. American Elsvier. Nachgedruckt in: [2].
- [46] Warren, D. (1976). Generating conditional plans and programs. In *AISB Summer Conference, Edinburgh*.
- [47] Weld, D. (1994). An introduction to least commitment planning. *AI Magazine*.
- [48] Weld, D. (1999). Recent advances in ai planning. *AI Magazine*.
- [49] Wilkins, D. E. (1988). *Practical Planning. Extending the Classical AI Planning Paradigm*. Morgan Kaufmann, San Mateo, CA.
- [50] Wilkins, D. E. (1990). Can ai planners solve practical problems? *Computational Intelligence*, 6(4):232–246.
- [51] Zilberstein, S. (1996). Using Anytime Algorithms in Intelligent Systems. *AI Magazine*, 17(3):73–83.

11 Neuronale Netze

Hanspeter A. Mallot und Wolfgang Hübner

11.1 Motivation

Schon vom Namen her nimmt die *Künstliche Intelligenz* Bezug auf die Leistungen der biologischen Informationsverarbeitung, die ihr als Vorbild, als Existenzbeweis und als Beispiel für Teillösungen dient. In diesem Kapitel soll auf einen Aspekt biologischer Informationsverarbeitung näher eingegangen werden, der insbesondere auch die Entwicklung maschineller Datenverarbeitung sehr stark beeinflusst hat, die *Neuronalen Netze*. Die Neuronalen Netze bilden ein Teilgebiet der Neuroinformatik, deren Gegenstand die Erforschung biologischer Informationsverarbeitung mit den Methoden der Informatik und Informationstechnologie ist (vgl. [39]).

Biologische Informationsverarbeitung dient der Organisation von Verhalten. Der Zweck biologischer Informationsverarbeitung ist es, Verhalten (und interne Regulationsprozesse) in Abhängigkeit von der jeweiligen Umweltsituation so zu organisieren, dass das Lebewesen sich behauptet. Die Ausstattung mit sensorischen Fähigkeiten und Verhaltensrepertoire wirkt dabei auf die Komplexität dieser Aufgabe zurück. So hat zum Beispiel die Entwicklung der *Hand* als eines *Manipulationsorgans* und die damit verbundene Erweiterung des Verhaltensrepertoires weitreichende Konsequenzen auch für die Hirnentwicklung der Primaten gehabt. Biologische Informationsverarbeitung kann geradezu als die umweltbezogene Organisation von Verhalten definiert werden; Umwelt meint dabei im Sinne von Uexkülls den Teil der Außenwelt, mit dem der Organismus interagieren (sich auseinandersetzen) kann [61].

Der Computer als Modell für das Gehirn. Vergleicht man die Funktionsweise neuronaler Systeme mit Modellen aus der Berechenbarkeitstheorie der Informatik so kann man zeigen, dass neuronale Systeme im Vergleich zur Turingmaschine den gleichen, im Fall rekurrenter Netzwerke sogar einen grösseren Funktionsumfang besitzen können [58]. Gehirne lassen sich allerdings nicht mittels formaler Sprachen programmieren; ihre Flexibilität gewinnen sie durch Anpassungen der Struktur („Plastizität“). Auf der anderen Seite zeigt ein Blick auf Abb. 11.1, dass die Probleme, zu denen man sich von der Neuroinformatik einen Beitrag erwarten kann (Sehen, Sprache, Manipulation, Lernen), große Bedeutung auch für die technische Informationsverarbeitung und die künstliche Intelligenz haben.

Die Arbeitsweise des Gehirns zeigt sich in seiner Struktur. Im Gegensatz zu *universellen* Rechnern sind Gehirne an ihre Funktionen angepasst. Nur aus diesem Grund ist es überhaupt möglich, im Sinne der neuronalen Netze von der Struktur („hardware“) auf die Funktion zu schließen. Im Ansatz der künstlichen Intelligenz ist diese „bottom-up“ Schlussweise explizit verboten (z.B. bei [42]). Umgekehrt folgt aus dieser Überlegung, dass künstliche neuronale Netze nur dann funktionieren können, wenn sie auf solchen Anpassungen aufbauen. Tun sie das nicht, so verlieren sie nicht nur die durch das Wort *neuronal* bezeichnete Motivation, sondern

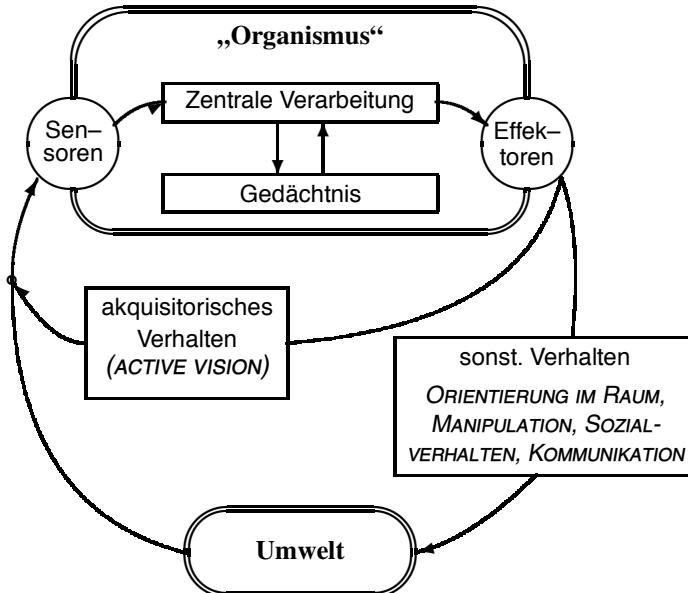


Abbildung 11.1: Biologische Informationsverarbeitung kann als „Informationswechsel“ aufgefasst werden, bei dem Verhalten in einer der jeweiligen Umweltsituation angepassten Weise erzeugt wird. Rückkopplungen des Verhaltens auf die wahrnehmbaren Sinnesreize finden entweder direkt („akquisitorisches Verhalten“, z.B. Augenbewegungen) oder über Wechselwirkungen mit der Umwelt statt. Zur „Umwelt“ gehören auch andere Organismen.

höchstwahrscheinlich auch ihre Anwendbarkeit: man kann vom Gehirn nur das lernen, was es selbst auch mit einiger Wahrscheinlichkeit tut. Die Abschnitte 11.2 und 11.4 fassen einige wichtige anatomische und physiologische Eigenschaften von Nervensystemen zusammen.

Informationsverarbeitungsprobleme für neuronale Netze. Da das Gehirn kein universeller Rechner ist, können auch nicht für alle Informationsverarbeitungsprobleme gute „neuronale“ Implementierungen existieren. Welche Probleme treten nun bei der Aufgabe „umweltbezogene Verhaltensorganisation“ auf? Interessante Fälle sind sicher die sog. *early vision* Probleme (Bildsegmentierung, Bewegungssehen, Tiefensehen, optische Navigation etc.), die sich biologischen Organismen tatsächlich stellen und die neurobiologisch z.T. auch schon sehr gut untersucht sind (insb. das Bewegungssehen). Ähnliche Probleme in anderen Sinnesmodalitäten sind z.B. das Richtungshören, die Unterscheidung verschiedener Schallquellen („Party-Effekt“) oder das Erkennen von Formen und Oberflächeneigenschaften. Im motorischen Bereich sind etwa Bewegungskoordination oder Manipulation zu nennen. Probleme, die primär auf das Erstellen von Repräsentationen abzielen oder für sich genommen bedeutungslose logische Probleme (wie z.B. das XOR-„Problem“) sind unter dem Gesichtspunkt der Verhaltensorganisation kaum gute Anwendungen oder auch nur Spielprobleme für die neuronale Informationsverarbeitung.

11.2 Natürliche neuronale Netze

In diesem Abschnitt werden einige wichtige Eigenschaften von natürlichen Nervensystemen kurz dargestellt, soweit sie für die Modellbildung bedeutsam sind. Dabei soll nicht bloß eine Motivation für einen „neuronalen Stil“ gegeben werden; vielmehr ist es gerade diese Anatomie und Physiologie, von der man für die Informationsverarbeitung lernen will. Für ein tiefergehendes Studium sei auf die zitierten Lehrbücher verwiesen.

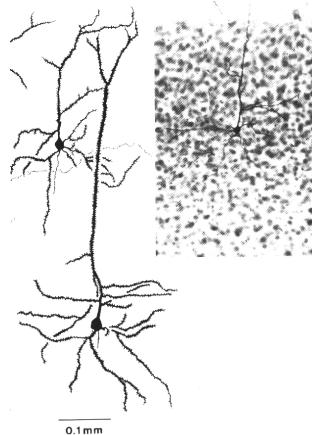


Abbildung 11.2: Nervenzellen (Pyramidenzellen) aus dem visuellen Cortex. Die dicken, mit Dornen („spines“) besetzten Fasern sind Dendriten, die dünnen Fasern im Bereich der linken Zelle Axone. Das Einschaltbild rechts zeigt ein Foto einer weiteren Pyramidenzelle vor dem Hintergrund der hier ebenfalls sichtbaren Somata der übrigen Nervenzellen; es vermittelt einen Eindruck von der Dichte des Gewebes. Aus [12].

11.2.1 Das Nervensystem besteht aus diskreten Zellen

Neuronentypen. Abb. 11.2 zeigt einige Nervenzellen (sog. Pyramidenzellen) aus dem visuellen Cortex einer Maus [12]. Sie bestehen aus folgenden Elementen: (1.) Der *Zellkörper (Soma)* enthält den Zellkern und wickelt die meisten biochemischen Synthesen ab. (2.) *Dendriten* sind relativ kurze und stark verzweigte Fortsätze des Somas mit (meist) nicht erregbarer Membran (s.u.). (3.) Das *Axon* ist die eigentliche leitende Nervenfaser, ein Fortsatz des Somas mit erregbarer Membran (Aktionspotentiale). Es kann bis zu mehreren Metern lang werden und ist wenig verzweigt. (4.) Die *Synapsen* sind die (gerichteten) Verbindungsstellen zwischen den Nervenfasern, genauer zwischen den axonalen Enden einer „präsynaptischen“ und den Dendriten einer „postsynaptischen“ Zelle. Durch den Erregungsfluss Dendrit → Soma → Axon erhält die Nervenzelle eine polare (gerichtete) Organisation.

Membranpotential. Wichtig für das Verständnis der Erregbarkeit von Nervenzellen ist die Tatsache, dass über der Membran der ganzen Zelle eine Spannungsdifferenz von ca. -70 mV (innen negativ) besteht, die auf die ungleiche Verteilung von Na^+ , K^+ , Ca^{2+} und Cl^- -Ionen sowie der Anionen saurer Proteine zurückgeht. Jeder dieser Ionenverteilungen lässt sich über die Nernstsche Gleichung ein Gleichgewichtspotential zuordnen. Insgesamt befindet sich das

System jedoch nicht im thermodynamischen Gleichgewicht; vielmehr werden die Ionenverteilungen von der sog. Kalium-Natrium-Pumpe unter Energieverbrauch aktiv erzeugt. Die Pumpe arbeitet gegen einen Leckstrom, der auf das Durchtreten einzelner Ionen durch die Membran zurückgeht. Man kann die elektrischen Verhältnisse am besten anhand eines Ersatzschaltbildes der Membran verstehen, in dem die Ionenverteilungen als Spannungsquellen, die Ionendurchtritte durch die Membran als widerstandsbehaftete (ohmsche) Ströme, und die Membran selbst als Kapazität dargestellt werden (Abb. 11.3a).

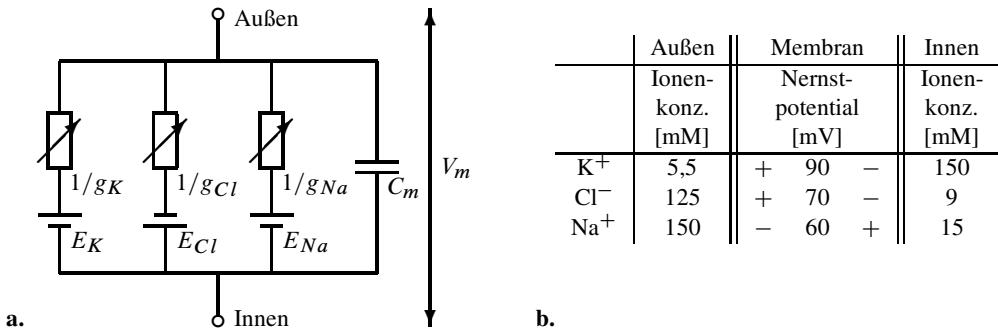


Abbildung 11.3: a. Ersatzschaltbild der Membran der Nervenzelle. Nach [28] b. Konzentrationen der drei wichtigsten Ionenarten und damit verbundene Gleichgewichts- (Nernst-)potentiale. Nach [45].

11.2.2 Nervenzellen sind erregbar

Die wichtigste Eigenschaft von Nervenzellen ist die Erregbarkeit. Diese Erregung (Aktivierung) stellt das eigentliche Signal bei der neuronalen Informationsverarbeitung dar.

Spannungsabhängige Kanäle. Ausgelöst durch bestimmte Änderungen des Membranpotentials ändern sich bei der Aktivierung die Leitfähigkeiten der Membran (vgl. Abb. 11.3), was natürlich wieder auf das Membranpotential zurückwirkt. Die Ionenleitfähigkeit der Membran beruht auf bestimmten, in die Membran integrierten Proteinkomplexen (den „Kanälen“), deren Konformation von der Membranspannung abhängt. Obwohl die dynamischen Prozesse etwa bei der Informationsverarbeitung in der Retina oder bei der dendritischen Summation eine große Rolle spielen, soll hier nicht auf die mathematische Beschreibung (Hodgkin-Huxley-Theorie) eingegangen werden. Eine einfache Einführung findet sich etwa bei [28].

Aktionspotential. Ändert sich das Membranpotential etwa durch einen von außen injizierten Strom, so relaxiert die Membran nach kleinen Reizen wieder zum Ruhepotential. Bei großen Depolarisationen (d.h. Abschwächungen des Ruhepotentials) steigt zunächst die Natrium-Leitfähigkeit sprunghaft an, wodurch die Depolarisation weiter verstärkt wird. Das Membranpotential steigt bis etwa +40 mV an. Durch die Schließung der Natrium-Kanäle und die verzögerte Öffnung von Kalium-Kanälen wird die Membran repolarisiert. Die (relativ geringen) Konzentrationsänderungen werden durch die K-Na-Pumpe wieder ausgeglichen. Den gesamten Zyklus bezeichnet man als Aktionspotential oder *spike*. Es handelt sich um einen binären („alles-oder-nichts“) Prozess, der entweder gar nicht oder, bei überschwelliger Reizung (Depolarisation), vollständig und dann mit stets etwa gleicher Maximalamplitude abläuft. Das Aktionspotential dauert ca. 1 ms. Daraus ergibt sich eine maximale Spikefrequenz in der Größenordnung von

1000 Hz. In der Regel ist der Wert aber kleiner, da die Membran nach einem Aktionspotential für einige Millisekunden *refraktär* ist.

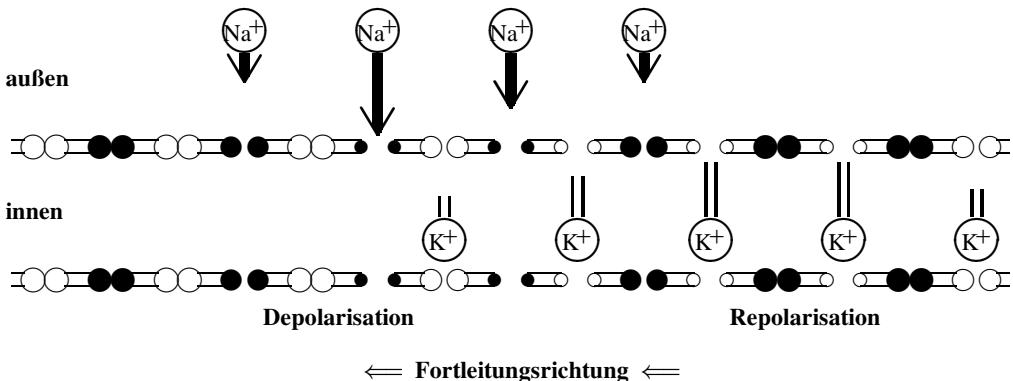


Abbildung 11.4: Schematische Darstellung der Leitfähigkeitsänderungen in der erregbaren Membran eines Axons während eines Aktionspotentials. Zu Beginn (links) werden durch die passiv vorauselende Depolarisierung die sonst geschlossenen Na^+ -Kanäle (•) geöffnet, so dass das Membranpotential schnell ansteigt (Depolarisation). Dadurch werden mit charakteristischer Verzögerung die K^+ -Kanäle (○) geöffnet, wodurch (wegen der umgekehrten Verteilung von Na^+ und K^+ -Ionen, vgl. Abb. 11.3b) das Membranpotential wieder absinkt (Repolarisation). Damit werden dann auch die spannungsabhängigen Kanäle wieder geschlossen. Die Länge der Pfeile deutet die Leitfähigkeiten g_{Na} bzw. g_{K} an.

Erregungsleitung. Die bisherige Betrachtung gilt für ein Membranstück an einem festen Ort. In der Nachbarschaft eines erregten Bereichs setzt nun durch passive Leitung (Kabelgleichung) ebenfalls eine Depolarisation ein, die zu einer Ausbreitung der Erregung führt. Im Prinzip erfolgt diese Ausbreitung in beide Richtungen der Nervenfaser; durch die Refraktärzeit wird jedoch erreicht, dass die Erregung stets gerichtet vom Ursprung wegläuft. Eine anschauliche Darstellung der Verhältnisse gibt Abb. 11.4. Durch geschickte Kombination aktiver und passiver Leitung werden Leitungsgeschwindigkeiten von bis zu 100 m/s erreicht.

11.2.3 Synaptische Übertragung

Synapsen. Die Signalübertragung von den axonalen Endverzweigungen eines Neurons auf den Dendriten eines anderen Neurons erfolgt in der Regel durch *chemische Synapsen* (vgl. Abb. 11.5). Diese bestehen aus einer *präsynaptischen Membran* am axonalen Ende der „sendenden“ Zelle, dem synaptischen Spalt sowie der *postsynaptischen Membran* der Empfängerzelle. Durch ein Aktionspotential kommt es in der präsynaptischen Endigung zur Ausschüttung eines *Neurotransmitters*, der auf der postsynaptischen Membran durch entsprechende *Rezeptoren* gebunden wird und in der Regel Änderungen der Membranleitfähigkeit bewirkt. Durch diesen chemischen Übertragungsmechanismus entsteht eine Vielzahl von Interaktionsmöglichkeiten zwischen den Signalen verschiedener Zellen, die für die neuronale Informationsverarbeitung von großer Bedeutung sind.

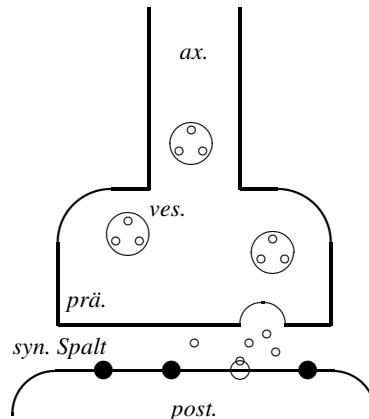


Abbildung 11.5: Schema einer chemischen Synapse. Das Axon der präsynaptischen Zelle (ax.) liefert die synaptischen Vesikel (ves.) mit dem Neurotransmitter an. Eintreffende Aktionspotentiale führen zu einer Verschmelzung dieser Vesikel mit der präsynaptischen Membran, wobei sich die Transmittermoleküle in den synaptischen Spalt (syn. Spalt) ergießen und zur postsynaptischen Membran diffundieren. Durch das Binden des Transmitters an spezifische Rezeptormoleküle (•, ○) werden postsynaptische Ionenkanäle geöffnet oder kompliziertere biochemische Reaktionen ausgelöst.

Postsynaptische Potentiale. Im einfachsten Fall werden durch die Transmittersubstanzen Ionenkanäle in der postsynaptischen (dendritischen) Membran geöffnet. Handelt es sich dabei um Na^+ - oder Ca^{2+} -Kanäle, so ist die Wirkung erregend (depolarisierend); handelt es sich um K^+ - oder Cl^- -Kanäle, so entsteht eine Hemmung (Hyperpolarisation). Postsynaptische Potentiale halten für einige Millisekunden an (bis der Transmitter abgebaut oder resorbiert ist) und breiten sich auf dem Dendriten in der Regel nur passiv aus.

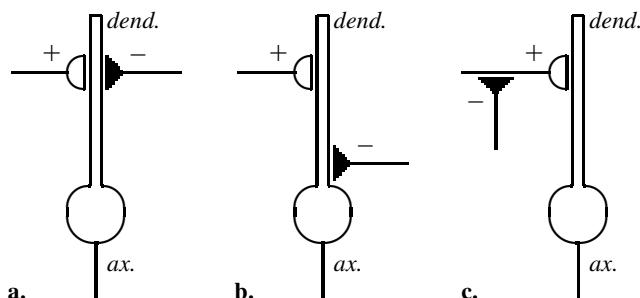


Abbildung 11.6: Dendritische Summation. **a.** Lineare Superposition bei benachbarten Synapsen. **b.** Proximale Hemmung. **c.** Präsynaptische Hemmung. dend.: Dendrit, ax.: Axon der postsynaptischen Zelle. „-“ hemmende, „+“ erregende Synapsen.

Dendritische Summation. Typische Cortexzellen empfangen in der Größenordnung von 10^4 Synapsen, deren postsynaptische Potentiale in komplexer Weise orts-zeitlich integriert werden. Ein Aktionspotential wird nach Maßgabe dieser Summation am Anfang des Axons ausgelöst, wenn das aus allen synaptischen Einflüssen resultierende „Generatorpotential“ die notwendige Schwelle übersteigt. Das Generatorpotential ist in erster Näherung eine mit dem Ab-

stand der Synapsen gewichtete Summe des ortszeitlichen Musters der postsynaptischen Potentiale; realistischere Modelle, die die dreidimensionale Geometrie des Dendritenbaumes berücksichtigen, erhält man in Form von partiellen Differentialgleichungen aus der Hodgkin-Huxley-Theorie. Zusätzlich wird die Situation dadurch kompliziert, dass sich mehr als zwei Fasern zu einem synaptischen Komplex zusammenschließen können, dass es dendro-dendritische Synapsen unter Umgehung des Somas gibt, dass lokal im Dendriten erregbare Membranbereiche auftreten können etc. Qualitativ kann man u.a. folgende Interaktionen finden:

1. *Lineare Summation.* Bei benachbarten Synapsen kann man davon ausgehen, dass die beteiligten postsynaptischen Membranen *äquipotentiell* sind. Die postsynaptischen Potentiale werden dann annähernd addiert.
2. *Proximale Hemmung.* Hemmende Synapsen am Ursprung eines größeren Dendriten-Astes können dessen Einfluss vollständig abschalten („Veto“).
3. *Präsynaptische Hemmung.* Zuweilen wirken hemmende Synapsen bereits präsynaptisch, d.h. sie sitzen einem Axon auf. Offensichtlich beteiligen sich solche Synapsen nicht direkt an der dendritischen Summation. Sie werden nur indirekt wirksam, indem sie einkommende Aktionspotentiale „abfangen“ (sog. *silent* oder *shunting inhibition*). Hierbei handelt es sich um eine multiplikative Nichtlinearität.

11.2.4 Lernen und synaptische Plastizität

Lernen ist Verhaltensänderung. Wenn Informationsverarbeitung die situationsgerechte Organisation von Verhalten ist, dann ist Lernen eine Veränderung dieser Organisation, sichtbar letztlich als eine Änderung des Verhaltens selbst. Dem Begriff des Lernens (auf der Verhaltenseite) steht auf der Seite der Struktur der *Plastizität*, d.h. der Strukturänderung, gegenüber. Im Sinne der Rückführung der Funktion des Gehirns auf seine Struktur sollte Plastizität die Grundlage des Lernens sein. In künstlichen neuronalen Netzen wird jedoch der Begriff der Plastizität meist vorschnell mit dem Lernen (und oft auch noch mit dem Gedächtnis) gleichgesetzt. Diese reduktionistische Identifizierung des Phänomens Lernen mit dem vermuteten Mechanismus ist einem wirklichen Verständnis der Zusammenhänge eher hinderlich.

Habituation (Gewöhnung). Ein Modell für synaptische Plastizität ist die Habituation (Gewöhnung) des Kiemen-Rückziehreflexes des Seehasen *Aplysia*, einer marinen Schnecke. Es handelt sich um einen monosynaptischen Reflex, bei dem die Reizung des Siphos (Einstromöffnung) mit dem Rückzug der empfindlichen Kieme beantwortet wird. Wiederholt man diese Reizung jedoch zu oft, so ändert sich die Effizienz der beteiligten Synapse(n) und die Reaktion lässt nach. Umgekehrt kann der Reflex durch schmerzhafte Reize am Fuss des Tieres verstärkt werden. Die Neurone und Synapsen, die an diesen Vorgängen beteiligt sind, sind sehr genau bekannt. Durch häufige Reizung werden in der Präsynapse Ionenkanäle abgebaut, die normalerweise den Einstrom positiver Ionen beim Eintreffen eines Aktionspotentials ermöglichen. Dadurch wird die Wirkung dieses Aktionspotentials und letztlich die Transmitterausschüttung abgeschwächt. Durch präsynaptische Kontakte anderer Neurone kann der Abbau der Kanäle rückgängig gemacht werden. Die Postsynapse ist an diesen Vorgängen nicht beteiligt.

Assoziative Modifikation von Synapsen. Eine Grundidee der künstlichen neuronalen Netze besteht in der Modifikation der synaptischen Übertragungsstärke durch (erfolgreichen) Gebrauch (Klassische Konditionierung, Hebb-Regel). Dabei wird eine Synapse dann verstärkt,

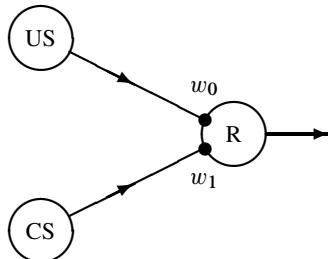


Abbildung 11.7: Modell der klassischen Konditionierung mit Hilfe von Hebb'schen Synapsen. CS: conditioned stimulus, US: unconditioned stimulus, R: Response, w_0, w_1 : synaptische Übertragungsgewichte. Vor der Konditionierung wird die Reaktion R nur durch den unkonditionierten Reiz hervorgerufen. Bietet man nun hinreichend oft CS und US gemeinsam, so wird w_1 verstärkt, da CS und R gleichzeitig aktiv sind. Schließlich kann CS die Zelle R alleine treiben. Dieses Modell setzt voraus, dass eine Verbindung zwischen CS und R bereits existiert.

wenn sie (1.) eine Erregung übertragen hat und (2.) die postsynaptische Zelle daraufhin aktiv wird („feuert“). Es ist instruktiv, sich die Verhältnisse anhand der *klassischen Konditionierung* zu überlegen (Abb. 11.7). In diesem Beispiel wird die „Synapse“ w_1 verstärkt, wenn (1.) der konditionierte Reiz CS geboten und (2.) die Zelle R (durch gleichzeitige Darbietung von US) aktiv ist. Man bezeichnet diese Regel für die synaptische Modifikation als Hebb-Regel.¹

Ähnliches Verhalten einzelner Synapsen ist in Form der sog. *Long-Term Potentiation, LTP* im Hippocampus und anderen Teilen der Großhirnrinde beschrieben worden (vgl. etwa [6]). Hierbei steigt die Effizienz der beteiligten Synapse(n) an, wenn einlaufende Aktionspotentiale die Aktivierung der postsynaptischen Zelle nach sich ziehen. Erregt man die Zelle elektrisch, ohne präsynaptischen Reiz, oder unterbindet man die Erregung trotz präsynaptischer Aktivität, indem man die Spannung mit einer geeigneten elektronischen Schaltung („*voltage clamp*“) konstant hält, so bleibt die Bahnung (d.h. die Verstärkung der Synapse) aus. Die *long-term potentiation* hält über einige Stunden an und klingt dann wieder ab.

Physiologie der Hebb-Synapse. Die Hebb'schen Synapsenregel besagt, dass eine Verstärkung der Synapse dann erfolgt, wenn gleichzeitig oder mit geeignetem Abstand prä- und postsynaptische Zelle erregt werden. Hierfür werden zwei Typen von Mechanismen diskutiert. Zum einen könnte die postsynaptische Zelle bei Erregung einen Botenstoff entgegen der synaptischen Übertragungsrichtung an die präsynaptische Zelle zurücksenden. Diese Funktion wird zumindest in manchen Fällen von Stickoxid, NO wahrgenommen. Zum anderen gibt es in vielen postsynaptischen Membranen des Gehirns Kanäle, die sich nur dann öffnen, wenn zum Zeitpunkt der Transmitterbindung an den Rezeptor die postsynaptische Zelle depolarisiert, d.h. vorerregt ist (sog. NMDA Rezeptor). In diesem Fall strömt Ca^{2+} in die postsynaptische Zelle ein, wodurch weitreichende Signalkaskaden in Gang gesetzt werden können. Letztlich können auf diesem Wege Gene in der postsynaptischen Zelle aktiviert werden, deren Expression zu einem Wachstum der Postsynapse und damit zu einem erhöhten Verstärkungsfaktor führen können.

¹ Donald O. Hebb (1949). Kanadischer Psychologe.

11.3 Künstliche neuronale Netze

11.3.1 Elemente neuronaler Netze

Grundgrößen für die Modellierung neuronaler Netze sind die Aktivität (Erregung) mit der zugehörigen Erregungsdynamik, die Übertragungsgewichte mit der zugehörigen Gewichtsdynamik und die Netzwerktopologie.

1. Erregung und Erregungsdynamik

1. *Erregungszustand (Aktivität):* Für ein Neuron i bezeichne e_i den Erregungszustand. Er kann z.B. die diskreten Werte $\{-1, 1\}$, $\{0, 1\}$ oder kontinuierliche Werte aus $[0, 1]$ oder \mathbb{R} annehmen. Der Vektor $\mathbf{e} := (e_1, e_2, \dots, e_I)^\top$ bezeichnet den Erregungszustand des ganzen Netzes. Im zeitkontinuierlichen Fall sind e_i und \mathbf{e} Funktionen von t . Im zeit- und ortskontinuierlichen Fall geht der Vektor \mathbf{e} in die globale Erregungsfunktion $e(\mathbf{x}; t)$ über.
2. *Aktivierungsfunktion (Übertragungsfunktion):* Die Aktivierungsfunktion $\alpha_i(\mathbf{e})$ beschreibt den zukünftigen Erregungszustand der Zelle i lokal durch die Zustände der benachbarten (= verbundenen) Zellen $\{e_j\}$. Im kontinuierlichen Fall entsteht ein „Aktivierungsfunktional“.
3. *Globale Erregungsdynamik:* Für bestimmte Probleme (z.B. Stabilitätsanalysen) kann man sogenannte „Netzwerkenergien“ definieren, die die Erregungsdynamik global beschreiben.

2. Gewichte und Gewichtsdynamik

1. *Übertragungsgewichte:* Die Verbindung zwischen zwei Neuronen e_i, e_j wird mit einem Gewicht w_{ij} versehen, das in die entsprechende Aktivierungsfunktion eingeht. Da die Verbindungen gerichtet sind, gilt im Allgemeinen $w_{ij} \neq w_{ji}$ (unsymmetrische Netze). Wir wollen w_{ij} als Gewicht des Übergangs $i \leftarrow j$ auffassen, d.h. in Richtung des „Einsammelns“ von Erregung. Im diskreten Fall bilden die w_{ij} eine quadratische Matrix W ; im kontinuierlichen Fall bilden die Gewichte den Kern $W(\mathbf{x}, \mathbf{x}'; t, t')$ eines Aktivierungsfunktionalen.
2. *Lernregel:* Lokale Regeln für die Verstellung von Gewichten bezeichnet man als „Lernregeln“. Sie hängen nur vom gegenwärtigen Gewicht der Verbindung sowie den Erregungszuständen der beteiligten Zellen ab (unüberwachtes Lernen).
3. *Globale Gewichtsdynamik:* Für manche Anwendungen können globale Anforderungen an das Netzwerk formuliert werden, die dann zur Einstellung der Gewichte benutzt werden. Solche Anforderungen werden häufig als Lehrer-Signal bezeichnet (überwachtes Lernen). Daneben gibt es Zwischenformen wie Wettbewerbslernen und Verstärklern.

3. Netzwerktopologie

Neuronale Netze sind in der Regel nicht vollständig verknüpft, sondern weisen komplexere Strukturen (z.B. Schichten u.a. Subpopulationen) auf. Ein Beispiel hierfür sind hierarchische Netzwerke. Neben der Gewichtsdynamik kann auch die Topologie während der Lernphase verändert werden. Beispiele hierfür sind Regularisierungsnetzwerke oder wachsende Netzwerkstrukturen.

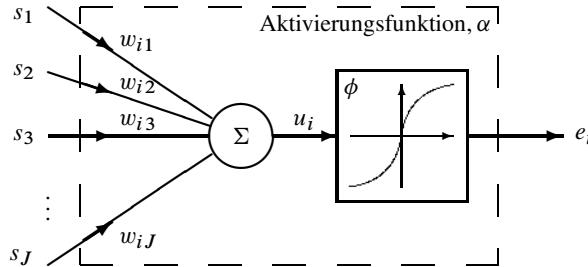


Abbildung 11.8: Einfaches Modellneuron für neuronale Netze. s_j : Eingänge (Stimuli), w_{ij} Übertragungsgewichte, Σ Summation, u_i Potential ($u_i = \sum_{j=1}^J w_{ij} s_j$), ϕ statische Nichtlinearität („Kennlinie“), e Erregung ($e_i = \phi(u_i)$).

11.3.2 Erregungsdynamik

Aktivierungsfunktionen

Eine Aktivierungsfunktion α ordnet einem Satz von Eingangssignalen s_j eine Erregung e_i zu. Dabei gehen in der Regel Übertragungsgewichte w_{ij} sowie eine nichtlineare Kennlinie ϕ ein, seltener auch nichtlineare Interaktionen der Inputs. Im Allgemeinen sind die s_j gerade die Erregungen der übrigen Zellen des Netzwerkes, im zeitdiskreten Fall etwa zum vorherigen Zeitpunkt.

Logische und semilineare Neurone. [44] führen die als „logisches Neuron“ bekannte Regel

$$e_i = \phi \left(\sum_j w_{ij} s_j - \theta \right) \text{ mit } \phi(u) := \begin{cases} 0 & \text{für } u < 0 \\ 1 & \text{für } u \geq 0 \end{cases} \quad (11.1)$$

ein, wobei e_i , die w_{ij} und θ diskret sind ($e_i \in \{0, 1\}$, $w_{ij} \in \{-1, 1\}$, $\theta \in \mathbb{N}$). Widrow & Hoff lassen kontinuierliche Gewichte zu („Adaptive Linear Neuron, ADALINE“). Der Schwellwert wird wie ein konstanter Eingang mit Gewicht $w_{i0} = -\theta$ behandelt, so dass eine einheitliche Behandlung der Adaptivität möglich wird [62].

Kontinuierliche Werte für alle beteiligten Größen und stetige oder differenzierbare Nichtlinearitäten charakterisieren das sogenannte „semilineare Neuron“. Die Erregung $e_i = \phi(\sum_j w_{ij} s_j) = \phi(\mathbf{w}_i \cdot \mathbf{s})$ eines Neurons wird dabei durch die Kennlinie $\phi : \mathbb{R} \rightarrow \mathbb{R}$, sowie die Projektion (Skalarprodukt) des Stimulus (\mathbf{s}) auf den Gewichtsvektor (\mathbf{w}_i) definiert² (s. Abb. 11.8). Beispiele der am häufigsten verwendeten Aktivierungsfunktionen sind

$$\phi(x, s) = \frac{2}{\pi} \arctan(sx) \quad \text{oder} \quad \phi(x, s) = \frac{1}{1 + \exp(-sx)} . \quad (11.2)$$

Während diskrete Erregungsgrößen dem Alles-oder-Nichts Charakter der Aktionspotentiale entsprechen, werden kontinuierliche Größen als momentane spike-Raten interpretiert.

Radialbasisfunktionen. Bisher wurde die Erregung e_i über die Projektion des Reizes auf den Gewichtsvektor definiert. Zuweilen ist es jedoch günstiger, die Erregung als eine Art

² \mathbf{w}_i ist der Zeilenvektor aus der Gewichtsmatrix $W = \{w_{ij}\}$.

Tabelle 11.1: Kennlinien die zur Definition von Radialbasisfunktionen verwendet werden können.

$\phi : \mathbb{R} \rightarrow \mathbb{R}_0^+$	
Gaußfunktion:	$\phi(x, r) = \exp(-(xr)^2)$
Multiquadrik	$\phi(x, r) = \sqrt{1 + (sr)^2}$
Inverse-Quadrik	$\phi(x, r) = \frac{1}{1+(sr)^2}$
Inverse-Multiquadrik	$\phi(x, r) = \frac{1}{\sqrt{(1+(sr)^2)}}$
Polyharmonic Spline	$\phi(x) = x^k, k = 1, 3, 5, \dots$ $\phi(x) = x^k \ln(x), k = 2, 4, 6, \dots$
Thin-Plate Spline	$\phi(x) = x^2 \ln(x)$

Ähnlichkeit des Reizvektors mit einem für die Zelle charakteristischen Merkmalsvektor $\mathbf{c}_i = (c_{i1}, \dots, c_{iJ})^\top$ zu definieren. Hierzu kann man Aktivierungsfunktionen der Form

$$e_i = \phi((\mathbf{c} - \mathbf{s})^\top \Sigma^{-1}(\mathbf{c} - \mathbf{s})) \quad \text{mit} \quad \Sigma \in \mathbb{R}^{J \times J} \quad (11.3)$$

verwenden. Diese sind aus einer Kennlinie, in diesem Fall einer positiven Halbfunktion $\phi : \mathbb{R} \rightarrow \mathbb{R}_0^+$, und dem Abstand zwischen dem Merkmalsvektor $\mathbf{c} \in \mathbb{R}^J$ und dem Stimulusvektor $\mathbf{s} \in \mathbb{R}^J$, zusammengesetzt. Im allgemeinen Fall kann hierbei die Mahalanobis-Distanz mit Kovarianzmatrix Σ verwendet werden. In vielen Anwendungen wird aber aus Komplexitätsgründen lediglich die euklidische Norm und eine uniforme Skalierung verwendet (s. auch Gl. 11.19). Aufgrund der verwendeten Norm ist die Aktivierungsfunktion rotationssymmetrisch, bzw. rotationsinvariant. Der Merkmalsvektor \mathbf{c} bildet dabei das Rotationszentrum. Aufgrund der Symmetrie und der Definition von ϕ , antwortet das Neuron immer dann maximal oder minimal, wenn $\mathbf{s} \approx \mathbf{c}_i$ gilt. Aus diesem Grund spricht man hierbei auch von *abgestimmten Neuronen*.

Die Bestimmung möglicher Kennlinien (s. Tabelle 11.1) geht dabei im Wesentlichen auf die Interpolationstheorie zurück (\rightarrow Theorem von Micchelli, [20, 51]). Von besonderem Interesse sind Kennlinien mit einem endlichen, zusammenhängenden Trägerbereich („local support“-Funktionen). Wie in Abschnitt 11.4.2 noch gezeigt wird, bilden diese Funktionen, insbesondere die Gaußfunktion, die Basis für die Modellierung *rezeptiver Felder* [13, 65]. Des weiteren besitzen die Gaußfunktion starke Ähnlichkeit mit Kernelfunktionen, weshalb Radialbasissysteme auch in den Bereich der Kernelmethoden einzuordnen sind (s. z.B. [20] S. 258ff).

Unter bestimmten Umständen kann der Gaußkern als Tiefpassfilter interpretieren werden. In diesem Fall definiert die Skalierung des Kerns dessen Bandbreite im Frequenzraum, weshalb in Zusammenhang mit Radialbasisfunktionen oft auch von der *Bandbreite* eines Kerns die Rede ist.

Explizite Repräsentation von Ort und Zeit. Zur Beschreibung dynamischer Vorgänge in kontinuierlicher Zeit eignen sich Aktivierungsfunktionen der Form

$$\frac{du_i}{dt}(t) = -cu_i(t) + \sum_j w_{ij}s_j(t - \tau_j) - \theta; \quad e_i(t) = \phi(u_i(t)). \quad (11.4)$$

Dabei bezeichnen c die Zeitkonstante und $\tau_j \in \mathbb{R}_0^+$ Latenzzzeiten (vgl. [23]). Will man überdies auch den Ort explizit berücksichtigen (indem man die Neurone nicht mehr durchnummiert, sondern mit einem Punkt im Raum identifiziert), so erhält man Integralgleichungen der

Form

$$e(\mathbf{x}) = \int W(\mathbf{x}, \mathbf{x}') s(\mathbf{x}') d\mathbf{x}'. \quad (11.5)$$

Durch Kombination mit Gl. 11.4 entstehen Integro-Differential-Gleichungen, wie sie etwa von [63] oder [40] verwendet werden.

Probabilistische Modelle. Betrachtet man die Stimuli und die Erregung eines Neurons als Zufallsvariable σ_j und η_i mit diskreten Werten in $\{0, 1\}$, so erhält man stochastische Aktivierungsfunktionen. In Anlehnung an Ansätze aus der statistischen Physik wählt man etwa:

$$P\{\eta_i(t) = 1\} = \phi_T \left(\sum_j w_{ij} \sigma_j - \theta \right); \quad \phi_T(u) := \frac{1}{1 + e^{-u/T}}. \quad (11.6)$$

Der Parameter T der Verteilung wird dann häufig als „Temperatur“ bezeichnet ; ϕ_T entspricht einer sigmoiden Kennlinie im deterministischen Fall.

Nichtlineare Interaktion der Eingänge. Aktivierungsfunktionen mit nichtlinearer Interaktion der Eingänge s_j sind z.B. die sogenannte Sigma-Pi Regel (multilineare Neurone) oder präsynaptische Hemmung (Abs. 11.2.3), modelliert als Division eines erregenden und eines hemmenden Eingangsterms.

Konsensfunktion und Netzwerkenergie. Bisher haben wir Erregungsdynamiken betrachtet, die lokal mit Hilfe von Aktivierungsfunktionen beschrieben werden. Die Zustandsänderung eines Neurons hängt dabei lediglich vom momentanen Zustand der Neurone ab, von denen es Eingänge erhält. Oft ist man jedoch an der Entwicklung globaler Zustandsgrößen interessiert (z.B. Energien oder Liapunov-Funktionen), z.B. bei der Untersuchung von Konvergenzverhalten oder bei der Konstruktion von Netzen für bestimmte Optimierungsprobleme. Als ein solches globales Maß der Netzwerkaktivität werden Funktionen der Form

$$k(\mathbf{e}) := \frac{1}{2} \sum_{i,j} w_{ij} e_i e_j + \sum_{j=1} w_{j0} e_j = \frac{1}{2} \mathbf{e}^\top W \mathbf{e} + \mathbf{w}_0^\top \mathbf{e} \quad (11.7)$$

untersucht. Dabei bezeichnet $-w_{j0}$ die Schwellen. Diese Funktion nimmt dann große Werte an, wenn zwei Zellen, die durch starke Gewichte w_{ij}, w_{ji} miteinander verbunden sind, auch tatsächlich gleichzeitig aktiv sind. (Man beachte, dass k jeweils nur von den Mittelwerten $(w_{ij} + w_{ji})/2$ abhängt. Daher auch der Faktor $1/2$ in Gl. 11.7.) Sind die Schwellen $w_{j0} = 0$, so folgt aus der Cauchy-Schwartzschen Ungleichung sofort, dass k bei gegebener Norm von \mathbf{e} durch die Eigenvektoren von W maximiert wird.

Eine andere Interpretation von k ergibt sich aus der Überlegung, dass für die Aktivierungsfunktion $\dot{\mathbf{e}} = W\mathbf{e} + \mathbf{w}_0$ (vgl. Gl. 11.4) mit symmetrischer Gewichtsmatrix $W = W^\top$ die Beziehung

$$\text{grad } k(\mathbf{e}) = \dot{\mathbf{e}} \quad (11.8)$$

gilt. In diesem Fall ist k also das Potential der Netzwerkdynamik. Für unsymmetrische W ist k eine Liapunov-Funktion (d.h. $k(\mathbf{e}(t))$ ist monoton fallend und nach unten beschränkt für jede Lösung $\mathbf{e}(t)$). Anwendung finden solche globalen Funktionen bei der Boltzmann-Maschine oder in Spin-Glas Modellen (vgl. [1]).

11.3.3 Grundtypen von neuronalen Netzen

Je nach der Dimensionalität von Input und Output kann man zwischen folgenden Netzwerktypen unterscheiden:

1. *Klassifikatoren* teilen den Raum der Inputvektoren in Klassen ein; im einfachsten Fall detektiert ein einzelnes Output-Neuron ein Muster einer bestimmten Klasse. $\mathbb{R}^J \rightarrow \{0, 1\}$
2. *Assoziatoren* ordnen bestimmten Inputvektoren bestimmte Outputvektoren zu. Dabei ist die Anzahl der assoziierten Paare relativ klein (z.B. gleich der Dimension des Inputraums) und jedenfalls endlich. Nicht gespeicherte Inputvektoren können mit Zwischenwerten oder mit einem der gespeicherten Outputvektoren beantwortet werden. Im zweiten Fall spricht man von einem klassifizierenden Assoziator. $\{\mathbf{s}^{(1)}, \dots, \mathbf{s}^{(P)}\} \subset \mathbb{R}^J \rightarrow \{\mathbf{e}^{(1)}, \dots, \mathbf{e}^{(P)}\} \subset \mathbb{R}^I$.
3. *Abbildende Netzwerke* implementieren eine Zuordnung zwischen beliebigen Inputvektoren und zugehörigen Outputvektoren. Sie werden benutzt, um Funktionen bzw. Operatoren zu approximieren. $\mathbb{R}^I \rightarrow \mathbb{R}^J$.

Assoziierende Netzwerke: Auto- und Heteroassoziator

Der Heteroassoziator besteht aus zwei Zellpopulationen. Wir betrachten ein Netzwerk aus zwei Teilpopulationen (Schichten) \mathcal{S}, \mathcal{E} von Neuronen mit den Erregungsvektoren $\mathbf{s} = (s_1, \dots, s_J)^\top$ und $\mathbf{e} = (e_1, \dots, e_I)^\top$. Innerhalb der Subpopulationen bestehen keine Verbindungen, während jede Zelle aus \mathcal{S} mit jeder Zelle aus \mathcal{E} verbunden ist; das zugehörige Gewicht sei c_{ij} . (Beachte, dass diese Definition von der bisherigen abweicht: c_{ii} meint nicht die Kopplung einer Zelle auf sich selbst, sondern die Kopplung zweier Zellen mit gleicher Nummer in \mathcal{S} und \mathcal{E} .) Mit der linearen Aktivierungsfunktion (ohne Schwelle und Kennlinie) hat man dann:

$$e_i = \sum_{j=1}^J c_{ij} s_j, \text{ bzw. } \mathbf{e} = C \mathbf{s} \text{ mit } \{c_{ij}\} = C. \quad (11.9)$$

Bezeichnet man mit W die Verknüpfungsmatrix auf der Menge aller Neurone $\mathcal{F} := \mathcal{S} \cup \mathcal{E}$ (d.h. $\mathbf{f} := (s_1, s_2, \dots, s_J; e_1, e_2, \dots, e_I)^\top$), so erhält man folgenden Zusammenhang:

$$\mathbf{f} \mapsto W\mathbf{f} = \begin{pmatrix} 0_{JJ} & 0_{JI} \\ C & 0_{II} \end{pmatrix} \mathbf{f}. \quad (11.10)$$

Dabei bezeichnet 0_{IJ} eine $I \times J$ -Matrix, deren sämtliche Koeffizienten verschwinden.

Verknüpfungsmatrix und äußeres Produkt. Es sei \mathbf{s} ein spezieller Stimulusvektor auf \mathcal{S} , der mit einem Erregungsvektor \mathbf{e} auf der Ausgangspopulation \mathcal{E} beantwortet oder „assoziiert“ werden soll. Gesucht ist also eine Matrix C mit der Eigenschaft $\mathbf{e} = C\mathbf{s}$. Eine solche Matrix erhält man aus der Kovarianzmatrix (oder dem äußeren Produkt) von Input- und Outputvektor. Wir setzen $C := \mathbf{es}^\top$ bzw. $c_{ij} = e_i s_j$ und erhalten

$$C\mathbf{s} = (\mathbf{e} \cdot \mathbf{s}^\top) \mathbf{s} = \mathbf{e}(\mathbf{s}^\top \mathbf{s}) = \mathbf{e}\|\mathbf{s}\|^2. \quad (11.11)$$

Die Koeffizienten $c_{ij} = e_i s_j \|\mathbf{s}\|^{-2}$ haben also die gewünschte Eigenschaft.

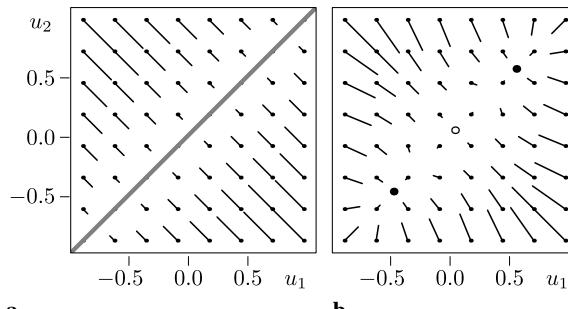


Abbildung 11.9: Dynamik des Autoassoziators aus Gl. 11.15. **a.** linearer Fall: Alle Punkte auf der Diagonalen sind Fixpunkte. Eingänge werden entlang der Linien projiziert. **b.** nichtlinearer Fall: $(0,0)$ ist instabiler Fixpunkt. Die zugehörige Projektionslinie trennt die Attraktionsgebiete der beiden stabilen Attraktoren (\bullet). Die „Nadeln“ deuten die Richtungen der $\dot{\mathbf{u}}$ an.

Speicherung mehrerer Assoziationen. Wir betrachten nun einen orthonormalen Satz von Inputvektoren $\mathbf{s}^{(p)}$ (d.h. $\mathbf{s}^{(p)} \cdot \mathbf{s}^{(q)} = \delta_{pq}$),³ die jeweils mit einem gegebenen Outputvektor assoziiert werden sollen. Wir wählen $C := \sum_q \mathbf{e}^{(q)} \cdot \mathbf{s}^{(q)\top}$ und erhalten:

$$\begin{aligned} C \mathbf{s}^{(p)} &= \left(\sum_q \mathbf{e}^{(q)} \cdot \mathbf{s}^{(q)\top} \right) \mathbf{s}^{(p)} = \sum_q \mathbf{e}^{(q)} \left(\mathbf{s}^{(q)\top} \mathbf{s}^{(p)} \right) \\ &= \sum_q \mathbf{e}^{(q)} \delta_{pq} = \mathbf{e}^{(p)}. \end{aligned} \tag{11.12}$$

Wegen der vorausgesetzten Orthonormalität kann dieses Verfahren auch bei geeigneter Vorkodierung nur funktionieren, solange die Anzahl der Musterpaare kleiner oder gleich J (Anzahl der Inputneurone = Dimension des Merkmalsraumes) ist. Im Falle P zu assoziierender Paare kann man allgemein so vorgehen: Man bezeichnet mit $S := [\mathbf{s}^{(1)}; \mathbf{s}^{(2)}; \dots; \mathbf{s}^{(P)}]$ und $E := [\mathbf{e}^{(1)}; \mathbf{e}^{(2)}; \dots; \mathbf{e}^{(P)}]$ die aus den P Input- bzw. Outputvektoren spaltenweise zusammengesetzten Matrizen. Dann lässt sich zeigen, dass die Verknüpfungsmatrix C die Form

$$C = E S^\top (S S^\top)^{-1} \tag{11.13}$$

annehmen muss, um die gewünschten Assoziationen mit minimalem quadratischen Fehler zu liefern. Ist $P \leq J$ und sind die $\mathbf{s}^{(q)}$ linear unabhängig, so liefert C sogar exakte Lösungen. $S S^\top$ ist die Kovarianzmatrix der Inputmenge, von der wir der Einfachheit halber annehmen wollen, dass sie regulär ist. Die Matrix $S^\top (S S^\top)^{-1}$ heißt *Moore-Penrose Pseudo-Inverse* von S .

Autoassoziator. Beim Autoassoziator (Abb. 11.10b) wird nicht zwischen einer Input- und einer Outputschicht unterschieden; vielmehr betrachtet man eine vollständig verknüpfte Population von N Zellen. Das Input-Muster stellt dann eine Anfangsbedingung dar, von der ausgehend der Assoziator durch seine intrinsische Dynamik einen Endzustand (Attraktor) anstrebt, der das

³ Hochgestellte Indizes numerieren die Mustervektoren (Trainingssatz).

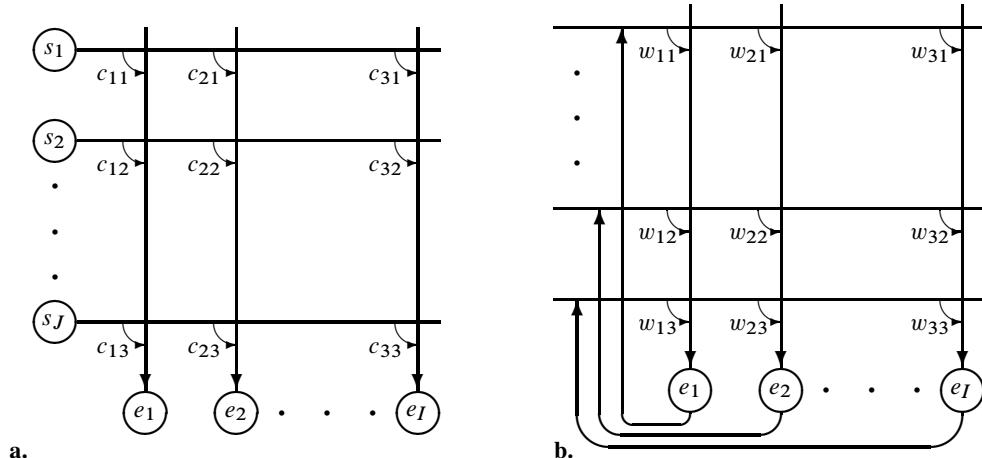


Abbildung 11.10: a. Heteroassoziator. b. Autoassoziator.

gewünschte Outputmuster ist. Zu jedem Attraktor gehört ein Attraktionsgebiet, d.h. eine Teilmenge des Inputraumes, mit der Eigenschaft, dass das Netzwerk Inputs aus diesem Gebiet mit dem zugehörigen Attraktor beantwortet.

Im zeitdiskreten, linearen Fall ergibt sich durch Iteration der Gleichung $\mathbf{e}(t+1) = W\mathbf{e}(t)$, ($\mathbf{e}(0) := \mathbf{s}$) im Endergebnis eine Projektion⁴, mit der Gleichung

$$\mathbf{e}^{end.} = V\mathbf{s}, \text{ mit } V := \lim_{t \rightarrow \infty} W^t. \quad (11.14)$$

Interessant sind nur die Fälle, in denen der größte Eigenwert von W eins ist; andernfalls konvergiert die Folge W^t nicht oder gegen null. Ist 1 etwa k -facher Eigenwert von W ($0 < k \leq N$), so spannen die zugehörigen Eigenvektoren einen k -dimensionalen linearen Unterraum auf, auf den die Abbildung V den Eingangsraum projiziert. Jeder Punkt im Bild von V , $\{V\mathbf{e} : \mathbf{e} \in \mathbb{R}^N\}$, ist ein Fixpunkt. Alle Punkte des Eingangsraumes, die auf ihn projiziert werden, liegen in einer $(N - k)$ -dimensionalen Hyperebene durch den Fixpunkt (vgl. Abb. 11.9). Man kann diese Hyperebene als eine Art „Attraktionsgebiet“ auffassen, doch handelt es sich im Gegensatz zu echten Attraktionsgebieten nicht um eine offene Menge. Benachbarte Startwerte können daher immer auf verschiedene Fixpunkte laufen.

Im nichtlinearen Fall kann es mehrere isolierte Attraktoren mit echten Attraktionsgebieten geben. Anschaulich entsprechen die Attraktoren z.B. den Minima eines Potentialgebirges und die Grenzen der Attraktionsgebiete den Wasserscheiden in diesem Gebirge. Die Klassifikationsfähigkeit nichtlinearer Autoassoziatoren ist also stärker als im linearen Fall: Ergebnisse „zwischen“ den Attraktoren können nicht auftreten.

Beispiel. Abb. 11.9 zeigt die Dynamik des aus zwei Zellen gebildeten Autoassoziators

$$\begin{pmatrix} \dot{u}_1 \\ \dot{u}_2 \end{pmatrix} = - \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + \frac{1}{4} \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} \phi(u_1) \\ \phi(u_2) \end{pmatrix}, \quad \mathbf{u}(0) = \mathbf{s} \quad (11.15)$$

⁴ Eine lineare Abbildung $P : \mathbb{R}^n \rightarrow \mathbb{R}^n$ mit der Eigenschaft $P^2 = P$ heißt (lineare) Projektion. Anschaulich ist das Bild von P der Unterraum, auf den projiziert wird. Der Kern von P markiert die Projektionsrichtung.

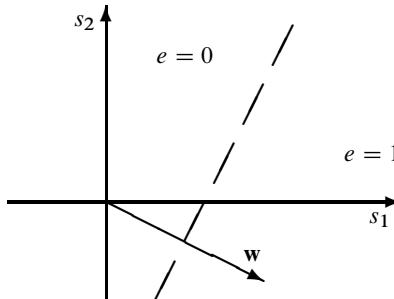


Abbildung 11.11: Interpretation eines Modellneurons mit einfacher Schwelle ($e \in \{0, 1\}$) als linearer Klassifikator. Die Trennfläche (gestrichelt) ist durch den Gewichtsvektor \mathbf{w} (ihren Normalenvektor) und die Schwelle $-w_0$ gegeben. Für Inputvektoren \mathbf{s} links dieser Trennlinie ist die Erregung 0, für Inputs rechts der Trennlinie 1.

(vgl. Gl. 11.4). Die Gewichtsmatrix hat die Eigenwerte 1 (Eigenvektor $(1, 1)^\top$) und 0.5 (Eigenvektor $(-1, 1)^\top$). Im linearen Fall (Abb. 11.9a) ist $\phi(u) \equiv u$ und alle Punkte auf der Diagonalen $u_1 = u_2 =: u$ sind Fixpunkte, auf die das System jeweils von den Startwerten $(u - \lambda, u + \lambda)^\top$ für alle $\lambda \in \mathbb{R}$ zuläuft. Im nichtlinearen Fall (Abb. 11.9b) wurde $\phi(u) = \frac{2}{\pi} \arctan(2u)$ gewählt. Man erhält dann einen instabilen Fixpunkt $\mathbf{u}_0 = (0, 0)$ sowie zwei stabile Attraktoren $\mathbf{u}_{1,2} = (\pm 0.5, \pm 0.5)$ mit den echten Attraktionsgebieten $\{u_1 > u_2\}$ und $\{u_1 < u_2\}$.

Klassifizierende Netzwerke: Das Perzeptron

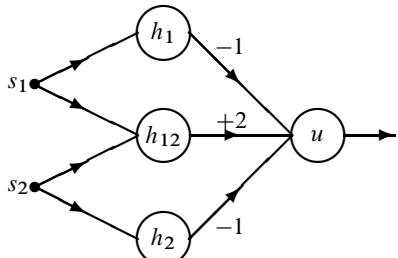
Semilineare Neurone sind lineare Klassifikatoren. Ein einzelnes Modellneuron mit der Aktivierungsfunktion

$$e := \begin{cases} 1 & \text{für } \sum_{j=0}^J w_j s_j \geq 0 \\ 0 & \text{für } \sum_{j=0}^J w_j s_j < 0 \end{cases} \quad (11.16)$$

kann als linearer Klassifikator auf dem Raum der Inputvektoren $\mathbf{s} = (s_1, \dots, s_J)^\top$ aufgefasst werden (Abb. 11.11). s_0 ist konstant 1, so dass $-w_0$ die Schwelle des Neurons ist. Das Potential u spielt hier die Rolle einer „Diskriminanzfunktion“ (vgl. Abs. 11.5); sie hängt (bis auf die Schwelle) linear vom Input \mathbf{s} ab. Wegen $u = \sum_{j=1}^J w_j s_j = \mathbf{w}^\top \mathbf{s}$ ⁵ ist die Trennfläche der Klassifikation durch die Hyperebene $w_0 + \mathbf{w}^\top \mathbf{s} = 0$ gegeben.

Die lineare Diskriminanzfunktion teilt den Eingaberaum in zwei Hälften, weshalb man lediglich zwei Klassen unterscheiden kann. Um den Ansatz auf eine beliebige Anzahl von Klassen zu erweitern, kann man für jede der K Klassen eine eigene Aktivierungsfunktion $y_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + w_{k0}$ einführen. Die Hyperebene die Klasse j von Klasse i trennt ist durch $(\mathbf{w}_k - \mathbf{w}_j)^\top \mathbf{x} + (w_{k0} - w_{j0}) = 0$ gegeben. Trennt man die Klasse i von allen anderen Klassen so entsteht für die Klasse i eine Entscheidungsregion. Man kann zeigen das diese Regionen immer konvex und zusammenhängend sind (vgl. etwa [10]).

⁵ Da alle Vektoren als Spaltenvektoren aufgefasst werden, ist $\mathbf{w}^\top \mathbf{s}$ das Skalarprodukt von \mathbf{w} und \mathbf{s} .



Input $s_1 \quad s_2$	Assoziation			Ausgang	
	h_1	h_2	h_{12}	u	$\phi(u)$
(w_i) 0 0	0	0	0	0	0
1 0	1	0	1	1	1
0 1	0	1	1	1	1
1 1	1	1	1	0	0

Abbildung 11.12: Einfaches dreischichtiges Perzeptron für die Lösung des XOR-Problems. Rechts die „Wahrheitstafel“ für die beteiligten logischen Neurone.

Lineare Klassifikatoren können kein „exklusives Oder“ implementieren. Klassen, deren Trennflächen keine Hyperebenen sind, können so nicht gebildet werden. Will man z.B. das „exklusive Oder“ (XOR) zweier Eingänge implementieren, so sollten die Inputvektoren $(0, 0)$ und $(1, 1)$ in eine, die Vektoren $(1, 0)$ und $(0, 1)$ jedoch in die andere Klasse fallen. Eine solche Klassifikation ist mit einer Trennebene aus geometrischen Gründen nicht möglich. Obwohl dies ein eher akademisches Problem ist, das sich etwa in Zusammenhang mit der Frage der Universalität einer Rechenmaschine stellt, zeigt sich bereits hier eine prinzipielle Limitierung des Klassifikationsansatzes. Mögliche Auswege sind (a.) die Verwendung mehrschichtiger Klassifikationsnetze, welchen eine verallgemeinerte Diskriminanzfunktion zugrunde liegt (\rightarrow feature-Extraktion), (b.) verbesserte Vorverarbeitung der untersuchten Daten (z.B. Dekorrelation und Dimensionsreduktion mittels Hauptachsentransformation) oder (c.) die Verwendung nichtlinearer Inputinteraktionen (z.B. Sigma-Pi-Neuronen). Die Verwendung nichtlinearer Aktivierungsfunktionen erinnert an den Polynom-Klassifikator $u = w_0 + \sum_{i=1}^n w_i s_i + \sum_{i,j=1}^n w_{ij} s_i s_j + \dots$ der konventionellen Mustererkennung.

Verallgemeinerte Diskriminanzfunktionen. Eine Möglichkeit linear nicht separierbare Klassen zu trennen besteht darin die Eingangsdaten in einen neuen Merkmalsraum zu transformieren,

$$\mathbf{L}(\mathbf{s}) = [\phi_1(\mathbf{s}), \dots, \phi_m(\mathbf{s})]^\top. \quad (11.17)$$

Die Dimension des Merkmalsraum ist in der Regel deutlich höher als die Dimension des Datensatzes. Diese Methode geht zurück auf das Theorem über die Separierbarkeit von Datensätzen nach Cover [14], welches im Wesentlichen aussagt, dass die Wahrscheinlichkeit einen Datensatz linear zu trennen mit der Dimension des Raumes zunimmt. Die verallgemeinerte Diskriminanzfunktion⁶,

$$y_k(\mathbf{s}) = \sum_{j=1}^m w_{kj} \phi_j(\mathbf{s}) + w_{k0} = \sum_{j=0}^m w_{kj} \phi_j(\mathbf{s}) = \mathbf{w}_k^\top \mathbf{L} \quad (11.18)$$

ist demnach eine Linearkombination aus nicht-linearen Transformations- oder Basisfunktionen. Wie im folgenden Abschnitt noch gezeigt wird, ist Gl. 11.18 unter bestimmten Umständen dazu geeignet jede beliebige Funktion zu approximieren. Im Fall mehrschichtiger neuronaler Netze werden die Merkmalstransformationen innerhalb der „verborgenen“ Schicht (oder auch

⁶ Hierbei wird angenommen, dass $\phi_0 := 1$ ist. Somit treten die Schwellwerte (Bias) wiederum als Gewichte w_{k0} in der Summierung auf.

Assoziationsschicht) realisiert, d.h. die Merkmalstransformation ist im Gegensatz zu anderen Klassifikationsansätzen, das Ergebnis eines Lernprozesses.

Abbildende Netzwerke

Viele Anwendungen in der Signalverarbeitung erfordern eine weitgehend stetige Transformation eines Erregungsvektors in einen anderen. Neuronale Netzwerke können hier eingesetzt werden, um solche hochdimensionalen Abbildungen (diskret: Funktionen, kontinuierlich: Operatoren) zu approximieren. Der Übergang zu den assoziativen Netzwerken ist fließend: beim Autoassoziator ist die Abbildung des Inputraumes auf die Outputs (Attraktoren) bereits stückweise stetig.

Ein Beispiel für eine allgemeine Funktionsapproximation wurde bereits in Form der verallgemeinerten Diskriminanzfunktion (vgl. Gl. 11.18) gegeben. Im Folgenden soll nun näher auf die in Gl. 11.18 verwendeten Basisfunktionen eingegangen werden.

Mehrschichtiges Perzeptron. Das mehrschichtige Perzeptron (MLP) verwendet semilineare Neurone als Basisfunktionen. Als Anwendung eines Satzes von Kolmogorov [31] kann man zeigen, dass jede stetige Abbildung $f : [0, 1]^n \rightarrow \mathbb{R}^m$ durch ein dreischichtiges vorwärts gekoppeltes Netzwerk mit n Inputzellen, m Outputzellen und $2n + 1$ Zellen in der mittleren Schicht exakt implementiert werden kann. Dabei müssen für alle Zellen (fast) beliebige nichtlineare Kennlinien zugelassen werden. Leider ist der Beweis nicht konstruktiv (vgl. [21]). Ein Verfahren zur Approximation von solchen Funktionen in Netzen mit differenzierbaren Nichtlinearitäten ist die sogenannte *back-propagation*.

Radialbasis Netzwerke. Im Gegensatz zu den MLPs bestehen Radialbasis-Funktions-Netzwerke (RBFN) neben einer Eingangs- und einer Ausgangsschicht in der Regel aus lediglich einer „verborgenen“ Schicht,

$$f(\mathbf{s}) = \sum_{i=0}^N w_i \Phi(\sigma^{-1} \|\mathbf{c}_i - \mathbf{s}\|_2) \quad , \quad (11.19)$$

wobei keine Übertragungsgewichte enthalten sind. Zu den Freiheitsgraden gehören neben den Gewichten auch Ort und Skalierung (Bandbreite) der Neurone. Nicht in allen Fällen werden auch alle potentiellen Freiheitsgrade verwendet. Neben der Beschränkung auf die euklidische Norm (s. auch Gl. 11.3), ist es z.B. möglich einen einheitlichen Skalenfaktor für alle im Netzwerk enthaltenen Knoten zu verwenden. In der Regel wird das Lernproblem durch solche Maßnahmen auf Kosten der Approximationsgüte vereinfacht.

Filteroperationen. Ein kontinuierliches, einschichtiges Netzwerk mit Inputverteilung $s(\mathbf{x})$ und Erregungsverteilung $e(\mathbf{x})$ (vgl. Gl. 11.5) kann als lineares Filter im Sinne der Bildverarbeitung aufgefasst werden. Hängt die Koppelstärke zweier Neuronen an den Positionen \mathbf{x} und \mathbf{x}' nur von ihrem (gerichteten) Abstand $\mathbf{x} - \mathbf{x}'$ ab, so ergibt sich z.B. eine einfache Faltungsoperation:

$$e(\mathbf{x}) = \int w(\mathbf{x} - \mathbf{x}') s(\mathbf{x}') d\mathbf{x}' . \quad (11.20)$$

In der Neurobiologie entspricht w dem rezeptiven Feld der Zelle an der Position \mathbf{x} (vgl. Gl. 11.60). Die Idee, dass rezeptive Felder mit den Merkmalsdetektoren der Bildverarbeitung in Verbindung gebracht werden können, hat großen Einfluss auf beide Disziplinen ausgeübt.

Selbstorganisierende Merkmalskarten. Während über die Selbstorganisation noch zu reden sein wird (\rightarrow Gewichtsdynamik), soll hier die Idee der „Merkmalskarte“ (*feature-map*) bereits kurz skizziert werden. Im Unterschied zur Filteroperation ist hier $W(\mathbf{x}, \mathbf{x}')$ aus Gleichung 11.5 nicht von der Form $w(\mathbf{x} - \mathbf{x}')$. Vielmehr werden von jeder „Zelle“ \mathbf{x} andere Reizeigenschaften betrachtet. Ein Beispiel ist die Selbstorganisation orientierungsspezifischer Kantendetektoren, die mit stetig variierender Vorzugsorientierung in der neuronalen Schicht angeordnet sind (vgl. Abs. 11.3.6). Allgemeiner kann auf diese Weise der für eine gegebene Reizklasse jeweils optimale Filtersatz gefunden werden (etwa im Sinne der Hauptachsentransformation). Dieser Typ neuronaler Netze erzeugt also Datenrepräsentationen.

11.3.4 Gewichts- und Strukturdynamik

Lernregeln für die Gewichtsdynamik. Für die Informationsverarbeitung mit neuronalen Netzen ist die Auswahl der Gewichte w_{ij} von großer Bedeutung. Zuweilen kann man sich für ein gegebenes Problem geeignete Gewichte analytisch überlegen, doch ist man in der Regel an „selbstorganisierten“ oder „adaptiven“ Mechanismen interessiert, nach denen diese Gewichte aufgabenabhängig gelernt werden können. Wichtige Unterscheidungskriterien für Lernregeln sind (vgl. Tabelle 11.2):

1. *Lernen mit und ohne Lehrer:* Beim überwachten Lernen kann man das gewünschte Netzwerkverhalten zumindest für einen vorgegebenen Satz von „Trainings-Mustern“ angeben.

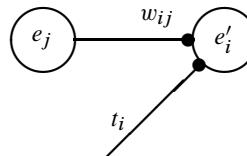


Abbildung 11.13: Komponenten von Lernregeln. e_j sei die Erregung des präsynaptischen Neurons zur Zeit t , e'_i die des postsynaptischen Neurons zur Zeit $t + 1$. w_{ij} ist das aktuelle Übertragungsgewicht und t_i ein „Lehrersignal“, das den gewünschten Zustand der postsynaptischen Zelle zur Zeit $t + 1$ wiedergibt. Regeln mit Lehrersignal heißen „überwacht“.

Tabelle 11.2: Typen von Lernregeln. λ : Lernrate. Weitere Bezeichnungen vgl. Abb. 11.13.

	unüberwacht	\longleftrightarrow	überwacht
lokal	Hebb-Regel: $\Delta w_{ij} = \lambda e'_i e_j$ dto. mit „Vergessen“ $\Delta w_{ij} = \lambda e'_i (e_j - w_{ij})$ Kovarianz-Lernen: $\Delta w_{ij} = \lambda (e'_i - \bar{e'})(e_j - \bar{e})$	Verstärkungslernen: $\Delta w_{ij} > 0$ falls gilt 1. e'_i war erregt 2. e_j hat beigetragen 3. Ergebnis wird besser	Widrow-Hoff: $\Delta w_{ij} = \lambda (t_i - e'_i) e_j$
	Wettbewerbslernen: $\Delta w_{ij} = \lambda e'_i e_j / (\sum w_{ij}^2)$		Back-Propagation
global	Vorbesetzung der Gewichte aufgrund analytischer Betrachtungen (z.B. Pseudo-Inverse)		

Dieses „Lehrersignal“ t liegt entweder für alle Zellen, für einen Teil (Outputzellen) oder nur als globale Bewertung (Verstärkungslernen) vor. Überwachte Lernregeln mit detailliertem Lehrersignal sind biologisch nicht begründet; eine Regel für nicht überwachtes Lernen ist die von Hebb.

2. *Lokale vs. nichtlokale Lernregeln:* Die Änderung des Übertragungsgewichtes w_{ij} hängt im einfachsten Fall nur von den Erregungen der beteiligten Zellen, e_i, e_j ab (strenge Lokalität). Darüberhinaus können die Erregungen anderer Zellen, weitere Übertragungsgewichte sowie Lehrersignale eingehen. Wir bezeichnen hier auch solche Regeln als lokal, bei denen pro Zelle ein Lehrersignal t_i eingeht.
3. *Relevante Erregungsfunktionen:* Neben der Erregung selbst können nichtlineare Funktionen oder Zeitableitungen der Erregung in die Lernregel eingehen.

Strukturdynamik. Neben den Netzgewichten ist auch die Struktur eines Netzwerks für dessen Verhalten von entscheidender Bedeutung. Zur Netzwerkstruktur zählen insbesondere die Anzahl der Neurone und deren Anordnung. Allgemeiner können aber alle freien Parameter eines Netzes, die vom Anwender festgelegt werden müssen, als Strukturmerkmale bezeichnet werden. Im Fall der Radialbasis-Netze kann dies z.B. die Breite der Gaußfunktionen sein. Ähnlich wie die Netzgewichte lassen sich auch die Strukturmerkmale automatisch für eine gegebene Aufgabe adaptieren. Hierzu zählt insbesondere die dynamische Anpassung der Neuronenzahl in den verborgenen Schichten. Varianten dynamischer Netzwerkstrukturen sind für die meisten Netzwerktypen definiert worden (s. z.B. [16, 24, 47]).

11.3.5 Überwachtes Lernen als Fehlerminimierung

Der Konvergenzsatz für Perzeptrons. Da Erregung und Gewichte der logischen Neurone eines Perzeptrons nur diskrete Werte annehmen können, ist das Perzeptron ein endliches System. Von „Konvergenz“ (d.h. dem Zulaufen auf ein funktionierendes Perzeptron für eine vorgegebene Aufgabe) kann daher eigentlich keine Rede sein, da man im Prinzip alle Zustände einfach durchprobieren könnte. Eine bessere Strategie gibt der sog. Konvergenzsatz für Perzeptrons. Wir betrachten ein dreischichtiges Perzeptron mit einer Eingangserregung s und einem gegebenen Satz von Assoziatoren mit Erregungsverteilung $(h_1, \dots, h_I)^\top = h$ (Abb. 11.12). Die Assoziatoren erregen eine Entscheidungszelle e :

$$e(s) := \phi\left(\sum_j w_j h_j(s) - \theta\right); \quad \phi(u) = \begin{cases} 1 & \text{für } u \geq 0 \\ 0 & \text{für } u < 0 \end{cases} \quad (11.21)$$

Es seien nun zwei Mengen von Reizmustern, S^+, S^- vorgegeben, die das Perzeptron unterscheiden soll. Man hat also ein Lehrersignal

$$t(s) = \begin{cases} 1 & \text{für } s \in S^+ \\ 0 & \text{für } s \in S^- \end{cases}. \quad (11.22)$$

Wir zeigen nun Bilder aus beiden Mengen und aktualisieren die Übertragungsgewichte nach folgendem Spezialfall der Widrow-Hoff-Regel (siehe unten):

$$\Delta w_j = h_j(t - e). \quad (11.23)$$

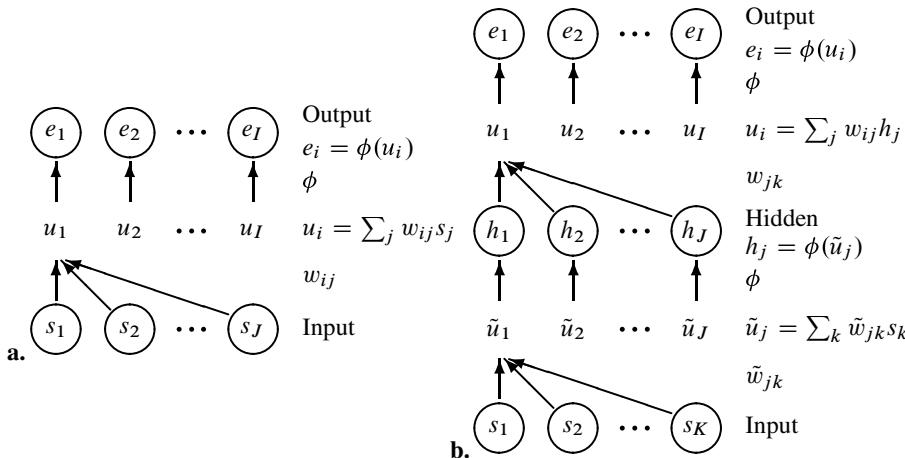


Abbildung 11.14: Zur Herleitung von überwachten Lernregeln aus Fehlerfunktionen. **a.** Widrow–Hoff Regel (= δ -Regel) für einen Heteroassoziator mit J Input- und I Output-Zellen. Für alle Zellen e_i liegt ein Lehrersignal vor. **b.** Back-Propagation Netzwerk (verallgemeinerte δ -Regel) mit K Input-, I Output- und J sogenannten verborgenen Zellen. Nur für die eigentlichen Outputzellen liegt ein Lehrersignal vor.

Nimmt man nun an, dass ein Gewichtsvektor \mathbf{w} existiert, mit dem $e \equiv t$ wird, so besagt der *Konvergenzsatz*, dass, ausgehend von einem beliebigen Startvektor \mathbf{w}^0 , dieser Vektor durch die Perzeptron-Lernregel nur endlich oft verändert wird. Mit anderen Worten, wenn eine Lösung existiert, wird sie mit dem skizzierten überwachten Lernalgorithmus auch gefunden [46]. Im nächsten Abschnitt wird gezeigt, wie die δ -Lernregel, von der die Perzeptron-Lernregel ja ein Spezialfall ist, aus einer Konvergenzforderung hergeleitet werden kann. Diese Herleitung lässt sich jedoch nicht direkt auf das Perzeptron übertragen, da die verwendeten Ableitungen für logische Neurone nicht definiert sind.

Widrow-Hoff oder δ -Regel. Wir betrachten einen Heteroassoziator mit Inputneuronen s_j , $j = 1, \dots, J$ und Outputneuronen e_i , $i = 1, \dots, I$ (Abb. 11.14a). Für jedes Inputmuster kann eine gewünschte Outputverteilung (t_1, t_2, \dots, t_I) angegeben werden (Lehrersignal). Man kann dann eine Fehlerfunktion

$$E := \frac{1}{2} \sum_{i=1}^I \left[e_i - t_i \right]^2 = \frac{1}{2} \sum_{i=1}^I \left[\phi \left(\sum_{j=1}^J w_{ij} s_j \right) - t_i \right]^2 \quad (11.24)$$

definieren, die durch eine geeignete Lernregel minimiert werden soll. Ein naheliegender Ansatz ist die Minimierung von E in den Gewichten w_{pq} durch Gradientenabstieg. Man bestimmt dazu die partiellen Ableitungen von E nach der Kettenregel:

$$\begin{aligned} \frac{\partial E}{\partial w_{pq}} &= \left[\phi \left(\sum_{j=1}^J w_{pj} s_j \right) - t_p \right] \phi' \left(\sum_{j=1}^J w_{pj} s_j \right) s_q \\ &= \underbrace{[e_p - t_p] \phi'(u_p)}_{-\delta_p} s_q. \end{aligned} \quad (11.25)$$

Daraus ergibt sich die Lernregel $\Delta w_{pq} = \lambda \delta_p s_q$, die (bei geeigneter Wahl der Schrittweite λ) mit der in Tabelle 11.2 angegebenen Widrow–Hoff–Regel identisch ist.

Back-Propagation. Die δ -Lernregel kann man für dreischichtige Netzwerke (Abb. 11.14b) folgendermassen verallgemeinern: Ausgehend von Gl. 11.24 überlegt man sich leicht, dass für die Gewichte der zweiten Schicht, w_{ij} , Gl. 11.25 entsprechend gilt. Für die Gewichte der ersten („verborgenen“) Schicht erhält man durch mehrfache Anwendung der Kettenregel:

$$\begin{aligned}\frac{\partial E}{\partial \tilde{w}_{pq}} &= \sum_{i=1}^I [e_i - t_i] \frac{\partial}{\partial \tilde{w}_{pq}} \phi(u_i) \\ &= \sum_{i=1}^I [e_i - t_i] \phi'(u_i) \frac{\partial}{\partial \tilde{w}_{pq}} \left(\sum_j w_{ij} h_j \right).\end{aligned}\tag{11.26}$$

Durch Einsetzen der Aktivierungsfunktion $h_j = \phi(\sum_k \tilde{w}_{jk} s_k)$ und erneute Anwendung der Kettenregel folgt:

$$\frac{\partial E}{\partial \tilde{w}_{pq}} = \sum_{i=1}^I [e_i - t_i] \phi'(u_i) w_{ip} \phi'(\tilde{u}_p) s_q = - \underbrace{\sum_{i=1}^I \delta_i w_{ip} \phi'(\tilde{u}_p)}_{\tilde{\delta}_p} s_q.\tag{11.27}$$

Dabei ist $\delta_i = -[e_i - t_i] \phi'(u_i)$ aus Gl. 11.25 benutzt worden. Mit dem neu definierten Fehlersignal $\tilde{\delta}$ erhält man für die \tilde{w}_{pq} formal wieder die δ -Regel. Aus Gl. 11.27 liest man ab, dass das Fehlersignal für die vorletzte Schicht aus dem der letzten Schicht errechnet werden kann, indem man „rückwärts“, d.h. über die post-synaptischen Zellen summiert: $\tilde{\delta}_p = \phi'(\tilde{u}_p) \sum_i w_{ip} \delta_i$. (Bei Aktivierungsfunktionen wird demgegenüber über den zweiten Index der Übertragungsgewichte summiert.)

Der Name „back-propagation“ röhrt von dieser rückwärts schreitenden Berechnung des Fehlersignals her, die übrigens auf etwaige weitere Schichten fortgesetzt werden kann. Eine biologische Implementierung dieses überwachten Verfahrens ist kaum vorstellbar, doch kann man mit *back-propagation* feststellen, ob bestimmte Leistungen prinzipiell in entsprechenden vorwärts gekoppelten Netzen erbracht werden können.

Die Herleitung der (verallgemeinerten) δ -Regel setzt differenzierbare Nichtlinearitäten ϕ voraus. In der Praxis geht man meist so vor, dass man jeweils ein Muster zeigt und nachfolgend die Gewichte aktualisiert („online–learning“). Im Sinne der Herleitung wäre es richtiger, jeweils den ganzen Trainingssatz („batch–learning“) vorzuführen und dann den mittleren Fehler zu verkleinern. Durch die Präsentation jeweils eines einzelnen Musters wird zusätzlich ein zufälliges Element in den Lernprozess eingebracht, weshalb die sequentielle Methode oft auch besser konvergiert.

Lernen in Radialbasisnetzwerken. Radialbasisnetzwerke der Form

$$t_k = \sum_i w_i \phi(\sigma_i^{-1} \|\mathbf{c}_i - \mathbf{s}_k\|_2), \quad \text{bzw. } \mathbf{t} = \Phi \mathbf{w}\tag{11.28}$$

besitzen neben den Gewichten auch noch die Skalierung⁷ (Bandbreite) und die Zentren der Basen als Freiheitsgrade. Je nachdem welche Beschränkungen man über die Freiheitsgrade oder die Verteilung der Daten machen kann, lassen sich folgende Varianten von Lernproblemen unterscheiden:

1. *Interpolation.* Ist das Ziel des Lernens die exakte Interpolation der Trainingsdaten, so kann man auf jeden Datenpunkt ein Zentrum legen ($\mathbf{c}_i = \mathbf{s}_i$). In diesem Fall ist die Größe der verborgenen Schicht und die Größe des Datensatzes gleich. Der Gewichtsvektor kann dabei durch Lösung des linearen Gleichungssystems (11.28) erfolgen: $\mathbf{w} = \Phi^{-1}\mathbf{t}$. Die Bandbreite der Basen ist dabei einheitlich. Sind alle Messpunkte paarweise verschieden, so ist die Matrix Φ aufgrund der Konstruktion der Radialbasisfunktionen (s. Tabelle 11.1) invertierbar.
2. *Linearer Ansatz.* In vielen anwendungsrelevanten Fällen möchte man die Trainingsdaten nicht interpolieren sondern mit minimalem Fehler approximieren. In diesem Fall ist die verborgene Schicht deutlich kleiner als die Trainingsmenge. Die Gewichte können dabei über die Pseudo-Inverse bestimmt werden: $\mathbf{w} = \Phi^\top(\Phi\Phi^\top)^{-1}\mathbf{t}$. Auch hier besitzen alle Zentren eine einheitliche Bandbreite. Dabei gilt das sowohl die Anzahl, als auch die Lage der Zentren bekannt sein müssen. Eine Möglichkeit besteht darin die Lage der Zentren vorab durch selbstorganisierte Lernmethoden wie „k-Means-Clustering“ zu bestimmen (s. [55], [20] S. 268 ff.).
3. *Nicht-lineare Optimierung.* Eine Alternative zu den bisherigen Methoden besteht darin die Bandbreite und den Ort mit in die Optimierung einzubeziehen. Dafür muss die Fehlergleichung

$$E := \sum_j [t_j - \sum_i w_i \phi(\sigma_i^{-1} \|\mathbf{c}_i - \mathbf{s}_j\|_2)]^2 \quad (11.29)$$

in allen Freiheitsgraden minimiert werden. Ausgehend von einer initialen Startlösung kann dann durch Gradientenabstieg eine verbesserte Lösung gefunden werden. In der Regel können durch die Verwendung variabler Zentren und Skalierungen deutlich bessere Ergebnisse erzielt werden (s. auch [51, 55]). Anzahl und initiale Lage der Basen werden dabei adaptiv in den Lernprozess integriert (s. z.B. [24]), so dass sich die Struktur des Netzwerks während des Lernens verändern kann.

4. *Hierarchische Anordnung.* Ausgehend von Gl. 11.28 kann man mehrere verborgene Schichten, bzw. Hierarchieebenen, hintereinander anordnen:

$$t(\mathbf{s}) = \sum_{j=0}^L \sum_{i=0}^{N_L} w_{i,j} \phi(\sigma_j^{-1} \|\mathbf{c}_i - \mathbf{s}\|_2) \quad . \quad (11.30)$$

Hierbei besitzen alle Neurone einer Ebene die gleiche Bandbreite, wobei die Ebenen in Bezug auf die Bandbreite absteigend geordnet sind ($j' < j : \sigma_{j'} > \sigma_j$). Strukturen dieser Art finden insbesondere Anwendung in der *Multiskalen-Signalrepräsentation*, welche auch die Basis für viele Modelle des visuellen Systems bilden (s. [2] [53]).

Angewandt auf das Approximationsproblem kann man das mehrschichtige Netzwerk dazu nutzen jeweils nur den Approximationsfehler an die nachfolgende Schicht weiterzugeben.

⁷ Im Allgemeinen kann hier die Mahalanobis-Distanz verwenden werden. Hierdurch können die Ergebnisse verbessert werden. Aufgrund der hohen Anzahl an Freiheitsgraden wird allerdings in der Regel darauf verzichtet.

Auf diese Weise kann man zunehmend feinere Details des Datensatzes approximieren (z.B. [15]).

5. *Kernelbasierte Dichteschätzung.* Geht man davon aus, dass die Messpunkte s_i unregelmäßig im \mathbb{R}^d verteilt sind, so ist durch,

$$\rho(\mathbf{x}) = \frac{1}{N(2\pi)^{\frac{d}{2}}} \sum_i^N \frac{1}{\sigma_i} e^{-\frac{1}{2\sigma_i^2} \|s_i - \mathbf{x}\|_2^2} \quad (11.31)$$

eine Schätzung der Dichte der Messpunkte gegeben. Verfahren dieser Art werden auch als „Kernel–Density Estimator“ bezeichnet. Die Bandbreite des Kerns definiert den lokalen Glättungsgrad, welcher je nach Anwendung mit aus den Daten geschätzt werden kann (vgl. [59]). Bis auf die Selektion der Bandbreite ist hierbei kein weiterer Lernschritt notwendig. In Gl. 11.31 wird für alle Basen ein einheitliches Gewicht verwendet, was der Normierung der Gesamtaktivität entspricht. Die Lage der Zentren ist bereits durch den Datensatz definiert. In der Regel ist dies eine sehr redundante Darstellung der Dichtefunktion, welche allerdings durch weitere Lernschritte (*Dichtekompression*) vereinfacht werden kann [32].

Allgemeine Anmerkungen. Zum Abschluss des Bereichs „überwachtes Lernen“ sei noch darauf hingewiesen das die hier vorgestellten Lernregeln alle Eigenschaften die mit gradientenbasierten Verfahren zusammenhängen besitzen. Hierzu zählen u.A. Probleme der Schrittweitensteuerung, Auswahl des Abbruchkriteriums, sowie der Umgang mit lokalen Minima. In gleicher Weise können hier aber auch viele der aus der Numerik bekannten Lösungen angewandt werden. In [9] wird hierzu ein Überblick über die Anwendung numerischer Optimierungsverfahren auf neuronale Netze gegeben.

11.3.6 Unüberwachtes Lernen

Unüberwachtes Lernen bedeutet das die Gewichtsdynamik ausschließlich durch die Daten und etwaige Vorannahmen bestimmt wird. Es wird daher kein Lehrersignal benötigt, bzw. das Lehrersignal wird selbst generiert oder ist durch die Vorgabe von Selbstorganisationsmechanismen bestimmt. Das Lernergebnis spiegelt hierbei inherente Eigenschaften oder Strukturen der Daten wieder. Aus diesem Grund ist unüberwachtes Lernen auch eine Möglichkeit zur Datenanalyse und -repräsentation.

Verborgene Variablen

Eine (statistische) Sichtweise auf das Lernproblem ist die Annahme, dass direkt beobachtbare (messbare) Daten $S = [s_1, \dots, s_n]$ durch einen Prozess generiert (generative Modelle) werden, welcher im Wesentlichen von nicht beobachtbaren (latenten) Variablen abhängt. In diesem Zusammenhang besteht das Lernproblem in der Bestimmung, bzw. Rekonstruktion der *latenten Variablen* allein aus den Beobachtungen und Vorannahmen über die statistische Natur der Daten.

Die Vorgehensweise ist dabei in allen Fällen ähnlich. Die Gewichte werden durch Optimierung einer Bestimmungsgleichung ermittelt. Um sicher zu stellen das alle Neurone hinreichend Informationen tragen, d.h. die Neurone sollten auf unterschiedliche Lösungen hin konvergieren, müssen noch zusätzliche Randbedingungen eingeführt werden. Als Randbedingung kommt hierbei

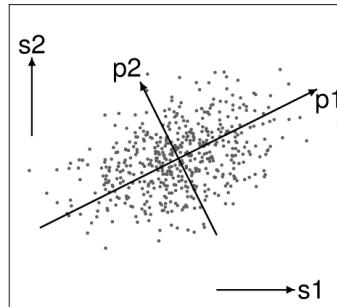


Abbildung 11.15: Hauptachsentransformation. Eine gegebene Menge von Inputvektoren $s^{(p)}$ eines Netzwerkes mit N Inputzellen bildet eine Punktwolke im N -dimensionalen Raum. Eine für viele Zwecke nützliche Umkodierung der $s^{(p)}$ erhält man, wenn man als neues Koordinatensystem die Eigenvektoren p_i der Kovarianzmatrix $\sum_p s^{(p)} s^{(p)\top}$ benutzt. Diese Eigenvektoren heißen Hauptachsen oder principal components des Datensatzes $\{s^{(p)}\}_p$.

die Annahme paarweise Dekorrelation, bzw. linearer Unabhängigkeit in Frage. Im Folgenden werden einige dieser Modelle, und deren Zusammenhang mit neuronalen Netzen, beschrieben.

Komponenten maximaler Varianz. Betrachtet man die Projektion der Eingangsdaten S auf einen Gewichtsvektor w , so kann man als Lernregel für die Gewichtsdynamik die Forderung stellen, dass die Varianz des auf w projizierten Datensatzes maximal wird. Unter der Annahme⁸ $E[S] = \mathbf{0}$ ergibt sich für den Gewichtsvektor die Bestimmungsgleichung,

$$\begin{aligned} E(w) &= \text{Var}(w^\top S) + \lambda(w^\top w - 1) \\ &= \mathbb{E}[(w^\top S)^2] + \lambda(w^\top w - 1) \\ &= \mathbb{E}[(w^\top S)(S^\top w)] + \lambda(w^\top w - 1) \\ &= w^\top \mathbb{E}[SS^\top] w + \lambda(w^\top w - 1) . \end{aligned} \tag{11.32}$$

Einzelne Komponenten können nun durch Maximierung von Gl. 11.32 ermittelt werden: $w_j = \arg \max_w E(w)$. Dies entspricht einer Optimierung unter der Randbedingung $\|w\| = 1$, wobei λ der zugehörige Lagrange-Multiplikator ist. Die Matrix $C = \mathbb{E}[SS^\top]$ ist die Kovarianzmatrix des Datensatzes. Das Ziel besteht nun darin Lösungen zu generieren, welche entlang ihrer Varianzen aufsteigend geordnet sind: $j < k : \text{Var}(w_j^\top S) > \text{Var}(w_k^\top S)$. Um sicher zu stellen das alle Lösungen auch unterschiedlichen Informationsgehalt haben, muss hierfür noch die Forderung paarweiser Unkorreliertheit gestellt werden: $j \neq k : \langle w_j, w_k \rangle = 0$. Insgesamt bedeutet dies, dass die Gewichtsvektoren ein Orthonormalsystem bilden. Man kann zeigen, dass sich das Problem auf ein Eigenwertproblem der Form $Cw = \lambda w$ zurückführen lässt. Hierbei treten die Lagrange-Multiplikatoren als Eigenwerte auf, welche wiederum der Varianz in Richtung des Eingenvektors entsprechen (vgl. auch [60]).

⁸ Die Annahme kann immer erfüllt werden indem man den Mittelwert von dem Datensatz abzieht, bzw. bei der Rücktransformation wieder hinzufügt. Wird S als Zufallsvektor interpretiert, so bezeichnet $\mathbb{E}[S]$ dessen Erwartungswert.

Statistisch unabhängige Komponenten. Im Gegensatz zur Hauptachsentransformation wird bei der Unabhängigkeitsanalyse („*Independent Component Analysis*“, ICA) die weitaus stärke Annahme gemacht, dass die latenten Variablen statistisch unabhängig sind. Die beobachtbaren Variablen (\mathbf{s}) entstehen dabei durch lineare Vermischung, d.h. durch Multiplikation der latenten Variablen (\mathbf{t}) mit einer nicht singulären Mischmatrix: $\mathbf{s} = \mathbf{A}\mathbf{t}$. Zur Rekonstruktion (Schätzung) der latenten Variablen muss daher die Abbildung umgekehrt werden: $\mathbf{t} = \mathbf{W}\mathbf{s}$. Die Bestimmung der Gewichtsmatrix \mathbf{W} erfolgt dabei ausschließlich aufgrund der beobachteten Variablen und der Unabhängigkeitsannahme. Die daraus entstehende Lösung ist bis auf Skalierung und Permutation der Komponenten eindeutig.

Je nachdem wie Unabhängigkeit definiert wird, lassen sich ICA-Algorithmen grob in zwei Klassen teilen: 1) Minimierung der wechselseitigen Information („Transinformation“, „Mutual Information“) ([3, 7, 50]) und 2) Maximierung der Abweichung von einer normalverteilten Zufallsvariablen. Aufgrund seiner Effizienz wird im Folgenden der zweite Weg im Rahmen des „FastICA“-Verfahrens ausführlicher beschrieben ([26]). Für eine Realisierung innerhalb einer neuronalen Architektur, sowie für die Beschreibung alternativer Methoden sei hier auf weiterführende Literatur verwiesen: [20, 27].

Als Maß für die Abweichung einer Zufallsvariablen von einer Normalverteilung mit gleicher Kovarianzmatrix kann die *negative Entropie* (Negentropie)

$$N(S) = H(S_{\text{gauss}}) - H(S) \quad \text{mit} \quad H(S) = - \int_{-\infty}^{\infty} p_S(s) \log(p_S(s)) ds \quad (11.33)$$

verwendet werden. Im Gegensatz zur Kurtosis ist die negative Entropie für alle nicht normalverteilten Zufallsvariablen positiv und nur im Fall einer Normalverteilung identisch 0. Anstatt zu fordern, dass die Kovarianzen der Normalverteilung in Gl. 11.33 gleich den Kovarianzen der Zufallsvariablen X sind, kann die negative Entropie effizienter durch

$$N(V) = \mathbb{E}[\Phi(V)] - \underbrace{\mathbb{E}[\Phi(U)]}_{\text{const.}} \quad (11.34)$$

angenähert werden. Hierbei vergleicht man eine um den Mittelwert 0 verteilte, varianznormalisierte Zufallsvariable V (d.h. $\mathbb{E}[V] = 0$ und $\text{Var}(V) = 1$) gegen eine normalverteilte Zufallsvariable U mit gleicher Varianz und Mittelwert. Vernachlässigt man in Gl. 11.34 den konstanten Term, welcher auf die Maximierung keinen Einfluss hat, so entsteht folgende Bestimmungsgleichung,

$$E(\mathbf{w}) = \mathbb{E}[\Phi(\mathbf{w}^\top \mathbf{s})] + \lambda(\mathbf{w}^\top \mathbf{w} - 1) \quad (11.35)$$

mit

$$\Phi(v) = \log(\cosh(v)) \quad \text{oder} \quad \Phi(v) = e^{-\frac{1}{2}v^2} \quad . \quad (11.36)$$

Die Robustheit der Gl. 11.36 wurde im Rahmen des „FastICA“-Algorithmus experimentell nachgewiesen [26], im Prinzip können hier aber auch andere Nichtlinearitäten verwendet werden. Hierbei sei noch anzumerken das aus der Randbedingung $\|\mathbf{w}\| = 1$ automatisch auch die Varianznormalisierung (vgl. auch Gl. 11.32) folgt:

$$\text{Var}(\mathbf{w}^\top \mathbf{S}) = \mathbf{w}^\top \underbrace{\mathbb{E}[\mathbf{S}\mathbf{S}^\top]}_{\text{Id}} \mathbf{w} = \|\mathbf{w}\|^2 = 1 \quad . \quad (11.37)$$

Die Bedingung $\mathbb{E}[\mathbf{SS}^\top] = \text{Id}$ kann durch entsprechende Normalisierung von S („Whitening“) immer erfüllt werden. Eine erste Komponente \mathbf{w}_0 erhält man nun durch Minimierung der Gl. 11.35 mittels Fixpunktiteration (z.B. Newton-Verfahren). Alle weiteren ($n - 1$) Komponenten werden durch die gleiche Optimierung bestimmt, wobei nach jedem Iterationsschritt noch eine Orthogonalisierung (nach dem Gram–Schmidt–Verfahren) durchgeführt wird,

$$\tilde{\mathbf{w}}_{i+1} = \mathbf{v}_{i+1} - \sum_{j=1}^i (\mathbf{v}_{i+1}^\top \mathbf{w}_j) \mathbf{w}_j \quad i = 1, 2, \dots, n. \quad (11.38)$$

Hierbei bezeichnet \mathbf{v}_i die aktuelle Zwischenlösung der i -ten Komponente innerhalb des Iterationsverfahrens. Die normalisierte Lösung ist dann: $\mathbf{w}_i = \frac{\tilde{\mathbf{w}}_i}{\|\tilde{\mathbf{w}}_i\|}$.

Langsam variierende Komponenten. Der Analyse langsam variierender Komponenten („Slow Feature Analysis“, SFA) liegt die Annahme zugrunde, dass sich die Ursache (Stimuli, Signalquellen) einer Wahrnehmung in der Regel nicht beliebig schnell verändert, während gleichzeitig die neuronale Repräsentation, d.h. die Erregung einzelner Zellen, sehr viel schneller variieren kann (*Prinzip der Langsamkeit*).

Formal lässt sich das Prinzip der Langsamkeit als Optimierungs-, bzw. Lernproblem beschreiben. Betrachtet man ein mehrdimensionales Signal $\mathbf{s}(t) = [s_1(t), \dots, s_n(t)]^\top$ im Zeitintervall $t \in [t_0, t_1]$, so kann man nach einer Abbildung suchen welche das Eingangssignal in ein neues Zeitsignal $\mathbf{y}(t) = [g_1(\mathbf{s}), g_2(\mathbf{s}), \dots, g_k(\mathbf{s})]^\top$ transformiert, für dessen Komponenten die zeitliche Varianz $\langle \dot{y}_j^2 \rangle$ minimal wird. Der Ausdruck $\langle f \rangle = \frac{1}{\Delta t} \int_{t_0}^{t_1} f(t) dt$ beschreibt dabei die zeitliche Mittelung. In den Funktionen $g_j(t)$ selbst erfolgt aber keine zeitliche Integration, so dass die Filterantworten ohne Verzögerung berechnet werden können ([64]).

Um die Lösung zu vereinfachen kann man die Funktionen g als lineare Überlagerungen nichtlinearer Funktionen $\mathbf{h} = (h_1(\mathbf{s}), \dots, h_K(\mathbf{s}))$ beschreiben: $g_j = \sum_k^K w_{j,k} h_k(\mathbf{s})$. Mit Hilfe der Merkmalstransformation⁹ $\mathbf{z}(t) = h(s(t))$ entsteht das lineare Optimierungsproblem:

$$\tilde{\mathbf{w}} = \arg \min_{\mathbf{w}} \langle \dot{y}_j \rangle^2 = \mathbf{w}_j^\top \langle \mathbf{z} \mathbf{z}^\top \rangle \mathbf{w}_j \quad . \quad (11.39)$$

Wählt man die Merkmalstransformation so, dass wiederum $\langle \mathbf{z} \rangle = 0$ und $\langle \mathbf{z} \mathbf{z}^\top \rangle = 1$ gilt, so ergibt sich:

$$\begin{aligned} \langle y_j \rangle &= \mathbf{w}_j \underbrace{\langle \mathbf{z} \rangle}_{=0} = 0 && \text{Mittelwertfrei} \\ \langle y_j^2 \rangle &= \mathbf{w}_j^\top \underbrace{\langle \mathbf{z} \mathbf{z}^\top \rangle}_{=\text{Id}} \mathbf{w}_j = \mathbf{w}_j^\top \mathbf{w}_j = 1 && \text{Varianznormalisiert} \\ \forall k < j \quad \langle y_k, y_j \rangle &= \mathbf{w}_k^\top \underbrace{\langle \mathbf{z} \mathbf{z}^\top \rangle}_{=\text{Id}} \mathbf{w}_j = \mathbf{w}_k^\top \mathbf{w}_j = 0 && \text{Dekorrelation} \end{aligned}$$

Die ersten beiden Bedingungen verhindern das triviale Lösungen der Form $g_j = \text{const.}$ entstehen. Die Forderung der Orthogonalität stellt wiederum sicher, dass hinreichend unterschiedliche Lösungen entstehen. Unter den gegebenen Voraussetzungen sind alle drei Bedingungen erfüllt wenn die Gewichtsvektoren ein Orthonormalsystem bilden.

⁹ Als Transformationen können z.B. Radialbasisfunktionen (s. Abs. 11.3.5) oder Monome verwendet werden.

Werden die Eingangsdaten entsprechend der Vorannahmen normalisiert, so reduziert sich das Lernproblem aus Gl. 11.39 wiederum auf ein Eigenwertproblem

$$\langle \mathbf{z} \mathbf{z}^\top \rangle \mathbf{w}_j = \lambda_j \mathbf{w}_j, \quad (11.40)$$

welches mittels Hauptachsentransformation gelöst werden kann. Hierbei liefern die betragskleinsten Eigenwerte die jeweils langsamsten Komponenten.

Selbstorganisation

Während im vorherigen Abschnitt statistische Grundannahmen über die Beschaffenheit der Eingangsdaten vorausgesetzt wurden, wird nun eine Klasse von Verfahren beschrieben, bei welchen sich die Gewichtsvektoren („reziproken Felder“) selbstorganisierend im Eingangsraum verteilen („Selforganised Featuremaps“, SOM). Hierbei wird angenommen das Zellen der Ausgangsschicht sich entsprechend verschiedener Selbstorganisationsmechanismen verhalten. Hierzu zählen vor allem das Prinzip des Wettbewerbs und das Prinzip der Kooperation (s. auch [20] S. 453 ff.). Obwohl diese Verfahren keine Vorannahmen über die Beschaffenheit der Daten machen, kann man dennoch zeigen das in einigen Fällen selbstorganisierte Mechanismen äquivalent zu statistischen Methoden sind (s.u.).

Wettbewerbslernen (Competitive Learning). Eine Strategie für nicht überwachtes Lernen, die oft für Selbstorganisationsprobleme eingesetzt wird, ist das Wettbewerbslernen. Hierbei benutzt man eine Variante der Hebb-Regel, die jedoch nicht auf alle Zellen wirkt, sondern nur auf die, die durch den gebotenen Reiz gerade maximal erregt wurde. Dies erreicht man z.B. dadurch, dass man die Zellen in einer Nachbarschaft inhibitorisch koppelt, so dass nachher überhaupt nur noch eine Zelle erregt ist („winner take all“). Eine andere Möglichkeit besteht in der Bilanzierung oder Normierung der Übertragungsgewichte (z.B. $\sum w_{ij}^2 = const$). Dadurch werden für jede verstärkte Synapse alle anderen unspezifisch abgeschwächt.

Ein Beispiel ist die von [48] untersuchte Lernregel, die sich von der ursprünglich von [41] eingeführten durch die Bilanzierung der Norm (statt der Summe) der Gewichte unterscheidet. Wir betrachten eine Menge von Inputvektoren $\mathbf{s}(t) := (s_1(t), \dots, s_J(t))^\top$ eines linearen Neurons mit Aktivierungsfunktion:

$$e(t) := \sum_{j=1}^J w_j s_j(t) = \mathbf{w}^\top \mathbf{s}(t). \quad (11.41)$$

Die Gewichte w_j werden nun nach einer bilanzierten Hebb-Regel verstellt:

$$\begin{aligned} w_j(t+1) &= \frac{w_j(t) + \lambda e(t)s_j(t)}{\sqrt{\sum_{j=1}^J (w_j(t) + \lambda e(t)s_j(t))^2}}; \\ \mathbf{w}(t+1) &= \frac{\mathbf{w}(t) + \lambda e(t)\mathbf{s}(t)}{\|\mathbf{w}(t) + \lambda e(t)\mathbf{s}(t)\|}. \end{aligned} \quad (11.42)$$

Für kleine λ geht diese Lernregel durch Taylor-Entwicklung um $\lambda = 0$ in

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \lambda e(t)[\mathbf{s}(t) - e(t)\mathbf{w}(t)] \quad (11.43)$$

über. In dieser Lernregel wird die Bilanzierung der Synapsengewichte also durch einen quadratischen Vergessensterm $e^2 \mathbf{w}$ erreicht. Man prüft leicht nach, dass die Änderung des (normierten) Gewichtsvektors stets orthogonal zum aktuellen Gewichtsvektor erfolgt, so dass die Norm von \mathbf{w} durch Gl. 11.43 nicht verändert wird.

Schreibt man in Gl. 11.43 $\mathbf{w}(t+1) - \mathbf{w}(t) = \Delta \mathbf{w}$ und lässt das Argument t weg, so erhält man durch Einsetzen von (11.41):

$$\Delta \mathbf{w} = \lambda [\underbrace{\mathbf{s}}_e \underbrace{\mathbf{s}^\top \mathbf{w}}_e - \underbrace{\mathbf{w}^\top \mathbf{s}}_e \underbrace{\mathbf{s}^\top \mathbf{w}}_e \mathbf{w}]. \quad (11.44)$$

Wegen der Assoziativität der Matrixmultiplikation kann man die äußeren Produkte $\mathbf{s}\mathbf{s}^\top$ des Trainingssatzes zusammenfassen; sie bestimmen hier die Gewichtsdynamik. Es sei daran erinnert, dass diese äußeren Produkte (Kovarianzmatrix) beim linearen Assoziator für die Konstruktion einer Gewichtsmatrix explizit angesetzt werden (Abschnitt 11.3.3).

Durch Mittelung gehen die äußeren Produkte in die Kovarianzmatrix $C := \mathbb{E}\{\mathbf{s}\mathbf{s}^\top\}$ ¹⁰ der Inputvektoren über. Man kann dann die Gewichtsdynamik so formulieren:

$$\frac{d\mathbf{w}}{dt} = C\mathbf{w} - (\mathbf{w}^\top C\mathbf{w})\mathbf{w}. \quad (11.45)$$

Man überlegt sich leicht, dass jeder (normierte) Eigenvektor von C ein Fixpunkt von (11.45) ist. Weiterhin gilt sogar, dass \mathbf{w} gegen den Eigenvektor von C mit dem größten Eigenwert konvergiert (vgl. [48]). D.h., der Gewichtsvektor \mathbf{w} der Zelle (ihr rezeptives Feld) konvergiert gegen die erste Hauptachse der Inputmenge.

Dieses Ergebnis kann in unterschiedlicher Weise auf Netze mit mehreren Outputzellen verallgemeinert werden. Zunächst ist klar, dass mehrere im Sinne eines Heteroassoziators angekoppelte Outputzellen nach der Ojaschen Lernregel alle gegen den gleichen Eigenvektor von C , nämlich gegen den mit dem größten Eigenwert, konvergieren müssten. Durch hemmende Kopplung innerhalb der Outputschicht kann man erzwingen, dass die Gewichtsvektoren orthogonal [54] oder wenigstens linear unabhängig [49] werden. In beiden Fällen spannen die Gewichtsvektoren der I Outputzellen einen I -dimensionalen Unterraum des J -dimensionalen Inputraumes auf, der mit dem von den ersten I Hauptachsen aufgespannten Raum identisch ist.

Topologie-erhaltende Merkmalskarten. Wir betrachten den linearen Assoziator, d.h. die *feed-forward* Projektion einer Inputschicht mit Erregungen $\mathbf{s} = (s_1, s_2, \dots, s_J)^\top$ auf eine Outputschicht mit Erregungsverteilung $\mathbf{e} = (e_1, e_2, \dots, e_I)^\top$. Innerhalb der Schichten bestehen zwar keine Verbindungen, doch ist eine Nachbarschaftsbeziehung (Topologie) auf der Outputschicht definiert. Die Aktivierungsfunktion ist durch die Input-Output-Gewichtsmatrix $C = \{c_{ij}\}_{i \leq I, j \leq J}$ gegeben:

$$e_i = \sum_{j=1}^J c_{ij} s_j := \mathbf{c}_i^\top \mathbf{s}. \quad (11.46)$$

Dabei bezeichnet $\mathbf{c}_i = (c_{i1}, c_{i2}, \dots, c_{iJ})^\top$ den Vektor aller Inputgewichte der Zelle i , d.h. ihr rezeptives Feld. [30] führt nun folgende competitive Gewichtsdynamik ein: Es sei

$$e_k := \max_{1 \leq i \leq I} \{e_i\} \quad (11.47)$$

¹⁰ \mathbb{E} bezeichnet hier den Erwartungswert der als Zufallsvariable aufgefassten Matrix $\mathbf{s}\mathbf{s}^\top$.

das Erregungsmaximum in der Outputschicht. Es wird an der Zelle erreicht, deren Gewichte dem aktuellen Inputvektor am ähnlichsten sind.¹¹ Nach jeder Reizpräsentation werden nun in einer Nachbarschaft \mathcal{G}_k der maximal erregten Zelle k die Gewichte dem Reizvektor angenähert:

$$\text{für alle } i \in \mathcal{G}_k: \mathbf{c}_i(t+1) := \frac{\mathbf{c}_i(t) + \alpha(t)\mathbf{s}(t)}{\|\mathbf{c}_i(t) + \alpha(t)\mathbf{s}(t)\|}; \quad \alpha(t) := \frac{\alpha_0}{t} \in \mathbb{R}. \quad (11.48)$$

Die Gewichte der übrigen Zellen bleiben unverändert. Kohonen diskutiert eine neuronal plausible Implementierung dieser Lernregel, bei der durch laterale Kopplungen in der Ausgangsschicht die Erregung auf die maximal erregte Zelle „zusammengezogen“ wird; danach kann man dann eine Variante der Hebb-Regel anwenden.

Wie die meisten anderen Selbstorganisationsmodelle auch, führt dieser Algorithmus zur Entstehung von reizangepassten *Gewichtsvektoren* (rezeptive Felder, Merkmalsdetektoren), d.h. zu Datenrepräsentationen, von denen man annimmt, dass sie in einem noch näher zu spezifizierenden Sinn „optimal“ sind. Die Modelle von [48, 49, 54] geben das Optimalitätskriterium genau an: die Netzwerke implementieren eine Hauptachsentransformation (vgl. Abs. 11.3.6). Bei Kohonen ist das Kriterium komplizierter. Durch die \mathcal{G}_k wird eine Nachbarschaftsbeziehung in der Ebene definiert, die dazu führt, dass die Gewichtsvektoren benachbarter Zellen ähnlich sind; bei der Hauptachsentransformation sind demgegenüber die Gewichtsvektoren aller Zellen paarweise unkorreliert (orthogonal). Da andererseits für jeden Reizvektor $\mathbf{s}^{(p)}$ eine mehr oder weniger passende Zelle existiert, wird im Endeffekt die Reizmenge $S := \{\mathbf{s}^{(p)} : 1 \leq p \leq P\}$ unter möglichster Erhaltung ihrer Topologie auf die Ebene der Ausgangsneurone abgebildet. Während nun die Hauptachsentransformation nur dann sinnvoll anwendbar ist, wenn die Inputvektoren S in etwa ein Ellipsoid im Merkmalsraum \mathbb{R}^I ausfüllen, kann man Kohonen-Karten auch auf komplexere Datensätze anwenden. Der entscheidende Punkt des Verfahrens liegt darin, dass funktionelle Ähnlichkeit auf räumliche Nähe abgebildet wird. Interessante Anwendungen und eine umfassende Diskussion des Verfahrens geben [19].

Abschlussbemerkungen

Ein Prinzip, welches im Bereich der neuronalen Netze (bzw. in dem gesamten Gebiet des maschinellen Lernens) eingesetzt wird, ist die Verallgemeinerung bestehender Methoden durch Einführung nicht linearer Merkmalstransformationen. Der große Vorteil besteht darin, dass die Netzgewichte über lineare Methoden bestimmt werden können und dass obwohl die Merkmalstransformationen selbst nicht linear sind. Als Beispiel hierfür wurde bereits die SFA, oder die verallgemeinerte Diskriminanzfunktion (vgl. Gl. 11.18) besprochen. Auf die gleiche Weise wurden in den letzten Jahren sehr viele der in den vorherigen Abschnitten vorgestellten Verfahren verallgemeinert. Hierzu zählen u.A.: *Kernel-PCA*, *Kernel-ICA*, *Kernel-SOM*, sowie die verallgemeinerte Hebb-Regel. Eine ausführliche Zusammenfassung dieser Methoden findet sich in [20].

11.3.7 Generalisierung und Komplexität

Wir gehen nun nochmals auf den Fall des überwachten Lernens zurück, d.h. zu einem gegebenen Trainingssatz $S = \{\mathbf{s}_1, \dots, \mathbf{s}_n\}$, $\mathbf{t} = \{t_1, \dots, t_n\}$ und einem gegebenen Netzwerk $h(\mathbf{s})$ sollen

¹¹ Dies folgt für semilineare Neurone aus der *Cauchy-Schwarzschen Ungleichung*, die besagt, dass das Skalarprodukt $\mathbf{a}^\top \mathbf{b}$ zweier Einheitsvektoren ($\|\mathbf{a}\| = \|\mathbf{b}\| = 1$) dann maximal wird, wenn $\mathbf{a} = \mathbf{b}$ ist. Sind \mathbf{a} und \mathbf{b} Zufallsvektoren, so ist durch den Erwartungswert des Skalarproduktes $\mathbf{a}^\top \mathbf{b}$ gerade ihre Kovarianz gegeben.

die Netzgewichte durch Minimierung der Fehlerfunktion

$$E(\mathbf{w}) = \sum_{i=1}^N [t_i - h(\mathbf{s}_i, \mathbf{w})]^2 \quad \text{mit} \quad \mathbf{w}_{\text{opt}} = \operatorname{argmin}_{\mathbf{w}} E(\mathbf{w}) \quad (11.49)$$

bestimmt werden. Das in Gl. 11.49 verwendete Netzwerk gehört z.B. einer der in den vorhergehenden Abschnitten beschriebenen Netzwerkklassen an. In der Regel hat man aber eine Vielzahl von Möglichkeiten. Es können Netzwerke unterschiedlicher Größe oder auch Netzwerke aus verschiedenen Klassen verwendet werden. Mit anderen Worten: man kann in der Regel aus einer größeren (diskreten) Menge von Modellen $H = \{h_1, \dots, h_m\}$ auswählen. In diesem Zusammenhang stellt sich nun automatisch die Frage welches der vorhandenen Modelle ein gegebenes Problem am besten lösen kann. Hierbei ist der Restfehler (Residuum, „Goodness-of-Fit“) kein aussagekräftiges Kriterium, da das Residuum nur Aussagen über die verwendeten Lerndaten zu lässt. Aussagen über die Generalisierungsleistung, d.h. die Vorhersage von Funktionswerten an unbekannten Stellen sind damit nicht ohne weiteres möglich. Im Allgemeinen müssen folgende Probleme¹² beachtet werden:

1. Die Eingangsdaten sind durch Messfehler (Rauschen) verfälscht, was dazu führen kann, dass sich das Netzwerk an das Rauschen anpasst. Vorwissen über die Art des Rauschens ist nicht immer verfügbar und muss deshalb auch mit aus den Daten geschätzt werden.
2. Es sind im Verhältnis zur Dimension des Eingaberaums zu wenig Trainingsdaten vorhanden. Dies kann dazu führen, dass die Trainingsdaten exakt interpoliert werden und damit die Generalisierungsleistung verloren geht.
3. Das Lernproblem ist inherent unterbestimmt und daher ohne Vorannahmen über die grobe Struktur der Funktion nicht lösbar. Diese Situation tritt häufig bei der Lösung inverser Probleme auf.

Ein Weg diesen Problemen zu begegnen ist die *Modellselektion*. Neben der Bestimmung optimaler Parameter wird damit der Lernprozess über den Raum möglicher Netzwerke ausgedehnt. Im Folgenden werden nun einige Ansätze hierzu beschrieben.

Kreuzvalidierung ist eine Methode den Generalisierungsfehler numerisch zu bestimmen. Im einfachsten Fall wird die Menge der Trainingsmuster in zwei Hälften geteilt. Die eine Hälfte wird zur Bestimmung der Netzparameter verwendet, die andere Hälfte zur Validierung des Lernfortschritts. Das Training wird dann abgebrochen wenn der Validierungsfehler, und nicht der Trainingsfehler, sein Minimum erreicht hat. Diese, auch als „early-stopping“ bezeichnete Lernstrategie, ist durch die Beobachtung begründet, dass bei einem Gradientenabstiegsverfahren (\rightarrow back-propagation) am Anfang sehr große Lernfortschritte erzielt werden, diese aber mit der Zeit immer kleiner werden. In diesem Fall kann man davon ausgehen, dass sich das Netzwerk an sehr feine Details anpasst welche eher spezielle Eigenschaften des Lerndatensatzes widerspiegeln.

In vielen Fällen ist die Bereitstellung hinreichend großer Lerndaten mit einem erheblichen Aufwand verbunden. Eine Möglichkeit den Verifikationsfehler ohne zusätzliche Daten zu schätzen besteht darin den Lerndatensatz in K gleich große Teile zu teilen („ k -fold cross-validation“).

¹² Punkt 1 und 2 werden meist unter dem Begriff der Überanpassung („overfitting“) zusammengefasst.

Das Training wird nun K-mal wiederholt, wobei jeweils ein anderer Datenblock als Verifikationsmuster verwendet wird. Der Verifikationsfehler wird dann durch Mittlung der Einzelergebnisse bestimmt. Die „leave-one-out“ (LOO) Methode ist dabei die extremste Form, da jeweils nur ein Element zur Verifikation herangezogen wird.

Regularisierungsnetzwerke. Für eine Vielzahl von Anwendungen, insbesondere bei unterbestimmten, bzw. inversen Problemen, führt auch eine Aufteilung der Lerndaten zu keiner Lösung. Um dennoch brauchbare Lösungen generieren zu können, kann man den Lösungsraum, d.h. die Menge darstellbarer Funktionen, geeignet einschränken. Hierzu betrachten wir die modifizierte Fehlerfunktion,

$$\tilde{E}(\mathbf{w}) = \sum_i^N [t_i - h(\mathbf{s}_i)]^2 + v\Omega[h] = E(\mathbf{w}) + v\Omega[h] \quad . \quad (11.50)$$

Der Ausdruck $\Omega[h]$ ist ein Strafterm, welcher mit Gewichtung v überkomplexe Lösungen bestraft. Im Gegensatz zu den Lagrange-Multiplikatoren muss diese Bedingung nicht exakt erfüllt sein. Der Ausdruck $\Omega[h]$ ist ein Funktional, d.h. eine Abbildung die einer Funktion eine reelle Zahl zuordnet. In vielen Fällen entspricht $\Omega[h]$ einem Differentialoperator aus der Klasse der Tikhonov-Regularisierer, d.h. der Stabilisierer bewertet die Variation der Funktion h auf dem gesamten Wertebereich. Alternativ können hier auch Stabilisierer verwendet werden, die auf der Fourier-Transformierten von h basieren. Durch die Wahl von $\Omega[h]$ wird die Form der Funktion h festgelegt. So kann man z.B. zeigen das im Rahmen von Radialbasisfunktionen Regularisierungsnetzwerke entstehen welche selbst wieder auf Radialbasen beruhen (s. [51]).

Eine, aufgrund ihrer Einfachheit, häufig verwendete Methode ist der s.g. „weight-decay“-Regularisierer,

$$\Omega[h] = \frac{1}{2} \sum_i w_i^2 = \frac{1}{2} \|\mathbf{w}\|^2 \quad . \quad (11.51)$$

Dieser Stabilisierungsterm wird, in Kombination mit Gl. 11.50, auch als *Ridge-Regression* bezeichnet. Um ein besseres Verständnis für die Wirkungsweise des Stabilisierers zu erhalten kann man die Fehlerfunktion bis zur zweiten Ordnung in ihre Taylor-Reihe entwickeln. Durch anschließendes Nullsetzen der Ableitung erhält man dann eine verbesserte Lösung. Angewandt auf die Gleichungen 11.49 und 11.50 ergibt sich damit:

$$\nabla E + H\mathbf{w}^* = 0 \quad \text{und} \quad \nabla E + H\tilde{\mathbf{w}} + v\tilde{\mathbf{w}} = 0 \quad .$$

Expandiert man nun die beiden neuen Gewichtsvektoren mittels der Eigenvektoren der Hessematrix H ($H\mathbf{u}_j = \lambda_j \mathbf{u}_j$) zu $\mathbf{w}^* = \sum_j w_j^* \mathbf{u}_j$ und $\tilde{\mathbf{w}} = \sum_j \tilde{w}_j \mathbf{u}_j$, so erhält man:

$$\tilde{w}_j = \frac{\lambda_j}{\lambda_j + v} w_j^* \quad . \quad (11.52)$$

Gl. 11.52 verdeutlicht, dass die Wirkungsweise des Stabilisierers im wesentlichen durch die lokale Krümmung der Fehlerfunktion bestimmt wird. Für den Fall $\lambda_j \gg v$ gilt $\tilde{w}_j \approx w_j^*$, d.h. die Lösung entspricht weitgehend dem unregulierten Fall. Für $\lambda_j \ll v$ gilt $|\tilde{w}_j| \ll |w_j^*|$, d.h. die entsprechende Komponente des Gewichtsvektors wird weitgehend unterdrückt. Man muss

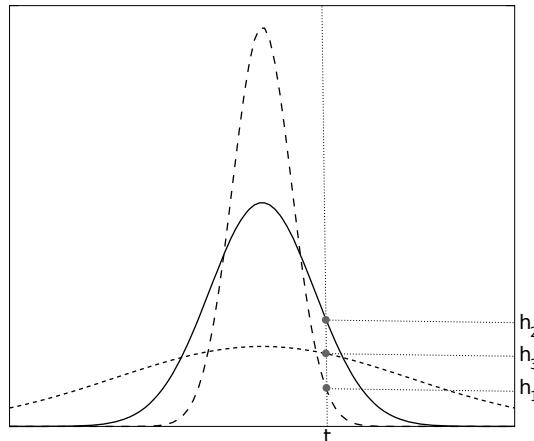


Abbildung 11.16: Schematische Darstellung der Evidenz für einen Datensatz \mathbf{t} (vgl. Gl. 11.55). Aufgrund der Normalisierung der Verteilung wird das komplexe Modell (h_3), welches sich an sehr unterschiedliche Daten anpassen kann, sowie das einfache Modell (h_1), welches nur einige der Daten korrekt repräsentieren kann, schlechter bewertet als das Modell mittlerer Komplexität (h_2). Auf der x-Achse sind, zumindest hypothetisch, alle möglichen Datensätze aufgelistet.

hierbei allerdings beachten das diese Abschätzung im konkreten Anwendungsfall stark von den verwendeten Hyperparametern , in diesem Fall dem Faktor v , abhängt. Im Folgenden wird nun ein Weg zur Schätzung solcher Hyperparameter aus den vorhandenen Trainingsdaten skizziert.

Bayes'sche Modellselektion. Im Folgenden wird eine wahrscheinlichkeitstheoretische Sichtweise auf das Problem der Parameterbestimmung, der Regularisierung, und letztlich der Modellselektion beschrieben. Diese Abfolge repräsentiert eine Hierarchie, welche sich direkt auf die Bayes'sche Inferenz abbilden lässt (vgl. [35]). Hierdurch wird eine Sichtweise auf das Lernproblem ermöglicht, die bis zu einem gewissen Grad von konkreten Netzwerktypen losgelöst ist.

Die A-posteriori Verteilung der Parameter \mathbf{w} eines Netzwerks h_i ist nach Bayes durch

$$\underbrace{p(\mathbf{w}|\mathbf{t}, \mathcal{S}, \Theta, h_i)}_{\text{Posterior}} = \frac{\underbrace{p(\mathbf{t}|\mathcal{S}, \mathbf{w}, h_i)}_{\text{Likelihood}} \underbrace{p(\mathbf{w}|\Theta, h_i)}_{\text{Prior}}}{\underbrace{p(\mathbf{t}|\mathcal{S}, \Theta, h_i)}_{\text{Evidenz}}} \quad (11.53)$$

gegeben. Die A-posteriori Verteilung beschreibt dabei wie sich das Vorwissen über die Netzwerkparameter, die A-priori Verteilung¹³, durch die Beobachtung der Lerndaten verändert hat. Es ist zu beachten das in den Konditionalteil auch das Modell (h_i) und die Hyperparameter (Θ) eingehen. Wendet man nun die Bayes–Formel erneut an, um die A-posteriori Verteilung der

¹³ Die A-priori Verteilung beschreibt Vorannahmen über die Gewichte, wie sie beispielsweise im vorherigen Abschnitt durch den Regularisierungsterm definiert wurden.

Hyperparameter zu bestimmen, so erhält man:

$$p(\Theta|\mathbf{t}, S, h_i) = \frac{p(\mathbf{t}|\Theta, S, h_i)p(\Theta|h_i)}{p(\mathbf{t}|S, h_i)} . \quad (11.54)$$

Hierbei ist zu beachten, dass die *Evidenz* (oder „Marginal–Likelihood“)

$$p(\mathbf{t}|S, \Theta, h_i) = \int p(\mathbf{t}|S, \mathbf{w}, h_i)p(\mathbf{w}|\Theta, h_i)d\mathbf{w} \quad (11.55)$$

aus Gl. 11.53 in Gl. 11.54 wiederum als „Likelihood“ eingeht. Im Gegensatz zu punktweisen Auswertungen, z.B. die Fehlerminimierungen aus den vorherigen Abschnitten, geht in Gl. 11.55 die vollständige Information über alle Netzgewichte ein. Die Evidenz ist die wahrscheinlichkeitstheoretische Umsetzung des Prinzips von „Occams–Razor“ in Bezug auf das Problem der Modellselektion¹⁴. Die durch Gl. 11.55 repräsentierte Verteilung bewertet Modelle mittlerer Komplexität automatisch besser als zu einfache oder zu komplexe Modelle (s. Abb. 11.16). Im Gegensatz zur Kreuzvalidierung kann hierbei allerdings der vollständige Lerndatensatz genutzt werden.

Durch Marginalisierung über die Hyperparameter,

$$p(\mathbf{t}|S, h_i) = \int p(\mathbf{t}|\Theta, S, h_i)p(\Theta|h_i)d\Theta , \quad (11.56)$$

ergibt sich nun

$$p(h_i|\mathbf{t}, S) = \frac{p(\mathbf{t}|S, h_i)p(h_i)}{p(\mathbf{t}|S)} \quad \text{mit} \quad p(\mathbf{t}|S) = \sum_i p(\mathbf{t}|S, h_i)p(h_i) . \quad (11.57)$$

Die Gl. 11.53, 11.54 und 11.57 verdeutlichen dabei nochmals den hierarchischen Ansatz: Gl. 11.53 ermöglicht Aussagen über die Verteilung der Netzwerkgewichte, Gl. 11.54 ermöglicht Aussagen über die Hyperparameter (d.h. Rauschmodell und Gewichtung des Regularisierungsterms), Gl. 11.57 ermöglicht Aussagen über unterschiedliche Modelle hinweg.

Für eine konkrete Anwendung des hier skizzierten Weges ist es allerdings notwendig komplexe Integrale zu lösen. Unter der Annahme das die Verteilung, insbesondere die Evidenz, nur ein einzelnes Extremum besitzt kann man die Integrale durch lokale Näherungsmethoden lösen (Verfahren der Laplace–Approximation, s. [36]). Für einige vereinfachte Fälle existieren auch analytische Lösungen (s. [9]). Des weiteren wurde die Methode der Bayes’schen Modellselektion auch im Rahmen von Kernelmethoden angewandt (s. hierzu [52]).

11.4 Modellierung biologischer Systeme

11.4.1 Neuroanatomie des visuellen Systems

Die Sehbahn. Die ersten Stationen der visuellen Informationsverarbeitung, bei denen die Erregungen an den Somata der beteiligten Nervenzellen umgeschaltet werden, sind die Retina,

¹⁴ In diesem Zusammenhang wird der Quotient aus „Likelihood“ und „Evidenz“ auch als *Occam–Faktor* bezeichnet.

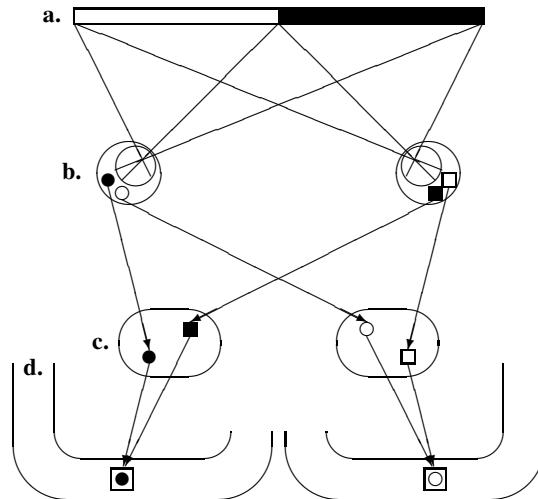


Abbildung 11.17: Schema der Sehbahn eines Säugetieres. Das Gesichtsfeld (a.) wird auf die Retinae der beiden Augen (b.) abgebildet. In jedem Auge sind zwei Ganglionzellen angedeutet: •, ○: linkes Auge. ■, □: rechtes Auge. ○, □: linkes Gesichtsfeld. •, ■: rechtes Gesichtsfeld. Die Axone der nasal gelegenen Zellen (○, ■) kreuzen im Chiasma opticum zur gegenüberliegenden Hirnhälfte, wo sie in das Corpus geniculatum laterale (LGN, c.) eintreten. Die Zellen des LGN werden jeweils nur von einem Auge innerviert. Erst bei der folgenden Projektion auf den visuellen Cortex (d.) konvergieren die Signale beider Augen auf binokulare Zellen.

das *Corpus geniculatum laterale (LGN)*¹⁵ im Zwischenhirn und der primäre visuelle Cortex im Großhirn. Dazwischen liegen Faserverbindungen, die aus den Axonen des jeweils vorhergehenden Kerngebietes gebildet werden: Sehnerv, Chiasma und optischer Trakt zwischen Retina und LGN sowie die optische Radiation zwischen LGN und Cortex. Eine Übersicht gibt Abb. 11.17.

Zelluläre Organisation des visuellen Cortex. Der Cortex ist ein etwa 1–2 mm dickes, flächig ausgedehntes Netzwerk mit einer Zelldichte von etwa 10000 pro mm³. Insgesamt besteht er aus etwa 10¹¹ Neuronen. Während die tangentiale Organisation mehr oder weniger uniform ist, gibt es in der Vertikalen deutliche Variationen der Netzwerkparameter. Mit neuroanatomischen Methoden lassen sich typischerweise sechs Schichten unterscheiden, die sich durch unterschiedliche Verteilung von Zelltypen und Konnektivitäten auszeichnen. Auch innerhalb der Schichten bestehen komplexe Verbindungsmuster; sie werden daher nur unzureichend durch die „Monolayer“ etwa eines mehrschichtigen Perzeptrons modelliert. Vgl. Abb. 11.2.

Columnäre Strukturen. Auf etwas größerem Niveau findet man eine Reihe von Ordnungsprinzipien, die unter dem Begriff der *Columne* zusammengefasst werden. Dabei variieren Verbindungsmuster oder Eigenschaften der rezeptiven Felder in mehr oder weniger regelmäßiger Weise über horizontale Abstände in der Größenordnung von 0,1 bis etwa 5 mm. Beispiele sind die Okulardominanzstreifen, die sich durch Entmischung der Eingänge aus den beiden Augen im visuellen Cortex bilden, die Orientierungskolumnen, die einer stetigen Variation der Vorzugs-

¹⁵ LGN ist die Abkürzung der englischen Bezeichnung „*lateral geniculate nucleus*“.

orientierung der rezeptiven Felder über dem corticalen Ort entsprechen, oder die Segregation des Outputs, bei der die Zellen bereichsweise in verschiedene Folgeareale projizieren.

Areale. In der Größenordnung von einigen Millimetern bis zu Zentimetern ist der Cortex in sog. *Areale* eingeteilt, die ursprünglich durch ihre zelluläre Organisation definiert wurden. Die meisten der (bei Primaten) ca. 20 bekannten visuellen Areale enthalten jeweils eine mehr oder weniger vollständige Repräsentation des Gesichtsfelds in Form einer *retinotopen Karte*. Das bedeutet, dass sich die Positionen der rezeptiven Felder im Gesichtsfeld stetig mit der corticalen Position der zugehörigen Zelle verschieben. Umgekehrt kann man sagen, dass benachbarte Punkte des Gesichtsfelds auf benachbarte Punkte des Cortex abgebildet werden. Zwischen den Arealen gibt es viele (aber nicht ausschließlich) vollparallele Projektionen, meist in beiden Richtungen. Die topographische Organisation entsteht im Wesentlichen durch die regelmäßige Anordnung der Inputfasern. Die intrinsische Organisation ist auf dem Zentimeter-Niveau in etwa uniform.

11.4.2 Rezeptive Felder beschreiben das Reiz-Reaktions-Verhältnis

Nervenzellen in den sensorischen Teilen des Gehirns beantworten bestimmte Sinnesreize stärker als andere. Im Allgemeinen reagieren sie nur auf Reize einer Modalität, wir beschränken uns hier auf das Sehen.

Spezifität. Im primären visuellen Areal (V1 = Area 17) der Großhirnrinde (*Cortex*) findet man z.B. Zellen, die spezifisch auf ganz bestimmte Eigenschaften oder Teilespektre der Sinnesreize reagieren. Typischerweise findet man folgende Spezifitäten:

1. *Position:* Die Zellen können jeweils nur von bestimmten Bereichen des Gesichtsfeldes erregt werden. Dieser Bereich ist das „rezeptive Feld“ in der ursprünglichen Wortbedeutung.
2. *Orientierung:* Zur Reizung verwendet man häufig helle Rechtecke („Balken“) auf dunklem Hintergrund. In diesem Fall hängt die Reaktion der Zelle von der Orientierung dieses Balkens ab.
3. *Ortsfrequenz:* Als Reizmuster können auch sinusförmig modulierte Grauwertverteilungen (Streifenmuster) verwendet werden. In diesem Fall hängt die Reaktion von der Ortsfrequenz und der Orientierung ab.
4. *Bewegung:* Die Reaktion praktisch aller Zellen hängt vom Zeitverlauf der Reizung ab. So werden etwa bewegte Reize bestimmter Geschwindigkeiten oder Richtung bevorzugt oder das Ein- und Ausschalten eines Reizmusters unterschiedlich beantwortet.
5. *Farbe:* Schließlich hängt die Reaktion häufig von der Farbe des Reizes ab.

Diese Liste ist natürlich nicht vollständig. Für alle diese Reizparameter kann man sogenannte *tuning*-Kurven messen, bei denen die Reaktion als Funktion des Reizparameters angegeben wird. Diese Abstimmkurven sind in der Regel nicht sehr scharf, so kann die Halbwertsbreite bei der Orientierungsspezifität durchaus 45° und mehr betragen. Es wird daher diskutiert, ob die Spezifitäten tatsächlich einzelnen Zellen oder eher größeren „Populationen“ zukommen (Populationskodierung).

Modelle rezeptiver Felder. Als Ansatz einer vereinheitlichenden Theorie rezeptiver Felder in den ersten Stufen der visuellen Informationsverarbeitung wurde von Adelson und Bergen ([2])

das Konzept der *plenoptischen* Funktion¹⁶ vorgeschlagen. Die plenoptische Funktion

$$P := P(\mathbf{x}, \lambda, t, \mathbf{v}) \quad (11.58)$$

ist ein hypothetisches Konstrukt, das die vollständige visuelle Information einer Szene in Abhängigkeit von der Blickrichtung $\mathbf{v} = (v_x, v_y, v_z)^\top$, der Position auf der Bildebene $\mathbf{x} = (x, y)^\top$, der Wellenlänge λ und dem Zeitpunkt t beschreibt. Die oben beschriebenen Spezifitäten entsprechen dabei Unterräumen der plenotischen Funktion, z.B. dem Ortsraum oder dem Orts-Zeitraum. Die Analyse komplexer Muster in den Unterräumen lässt sich sehr stark vereinfachen, wenn man diese lokal (d.h. durch eine Apertur) betrachtet. Nimmt man die eindimensionale Ortsfunktion $f(x)$ als Beispiel, so kann man diese lokal, d.h. in einer kleinen Umgebung um einen Punkt x , vollständig durch ihre (partiellen) Ableitungen

$$f^n(x) = \frac{\partial^n}{\partial x^n} f(x) \quad n = 0, 1, 2, \dots \quad (11.59)$$

beschreiben (Taylor-Entwicklung). Durch Mittelung, z.B. Faltung mit einer Gaussfunktion $G(x, \sigma)$, kann man nun den punktell definierten Ableitungen eine Ausdehnung zuordnen:

$$f^n(x, \sigma) = \frac{\partial^n}{\partial x^n} [f(x) \otimes G(x, \sigma)] = \underbrace{\left[\frac{\partial^n}{\partial x^n} G(x, \sigma) \right]}_{\text{Ableitungsfilter}} \otimes f(x) \quad . \quad (11.60)$$

Die hieraus entstehenden Filter werden auch als unscharfe Ableitungen („fuzzy derivatives“) (vgl. [29]) bezeichnet. Hierbei wurde die Skalierung (σ) als neuer Freiheitsgrad eingeführt. Die Verwendung unterschiedlich skaliert Filter ist im wesentlichen dafür verantwortlich, dass Bildelemente verschiedener Größen detektiert werden können.

Durch den Ansatz der unscharfen Ableitungen lassen sich bereits eine Vielzahl von rezeptiven Feldern im visuellen System erklären. Eine ausführliche Übersicht hierzu geben Adelson und Bergen ([2]). Alternativ zu dem Konzept der unscharfen Ableitungen werden häufig auch Gabor-Funktionen, verwendet (vgl. auch [33]). Neben der Skalierung geht hierbei noch die Ortsfrequenz und im Fall mehrdimensionaler Filter die Orientierung ein. Bei den unscharfen Ableitungen ist die Ortsfrequenz hingegen an die Skalierung gebunden. Diese Konzepte finden auch in der maschinellen Bildverarbeitung Anwendung. In der Regel werden dann aber genäherter Ableitungsfilter verwendet, welche sich deutlich effizienter berechnen lassen. Beispiel hierfür sind Haar-Wavelets und Integralbilder.

Neuronale Kodierung: „spezifische Sinnesenergien.“ Da alle Aktionspotentiale in etwa gleich sind, kann in der Form der Erregung keine Information kodiert sein. So führt z.B. eine mechanische Reizung der Netzhaut zu einer Sehwahrnehmung („Sternchen“), weil sich die mechanisch ausgelöste Erregung nicht von der „adäquat“, d.h. optisch ausgelösten unterscheidet¹⁷. Kodierungsmöglichkeiten im Nervensystem ergeben sich aus den Spezifitäten rezeptiver Felder sowie vielfältigen orts-zeitlichen Mustern von Erregungen. So ist z.B. die Reizstärke häufig in der momentanen *spike-Rate* (Kehrwert des Zeitintervalls zwischen zwei Aktionspotentialen) kodiert.

¹⁶ Plenoptisch ist ein Kunstwort, zusammengesetzt aus *plenum* (vollständig) und optisch.

¹⁷ Dies ist das von Johannes Müller bereits im 19. Jahrhundert formulierte „Gesetz der spezifischen Sinnesenergien“. In der neueren Literatur findet man dafür die Bezeichnung „*labeled line coding*“.

11.4.3 Visuelle Informationsverarbeitung mit neuronalen Karten

In diesem Abschnitt werden einige Ansätze zusammengestellt, die von der Neurobiologie nahegelegten Datenstrukturen sowie einfache Erregungsdynamiken nutzen. Im Gegensatz zur Selbstorganisation neuronaler Karten steht hierbei die Frage im Vordergrund, was man mit Karten anfangen kann, wenn man sie denn hat. Die erste Gruppe von Beispielen behandelt visuelle Informationsverarbeitung mit neuronalen Karten. Es sei ausdrücklich darauf hingewiesen, dass Karten auch in anderen Sinnesmodalitäten und z.T. im motorischen Bereich eine große Rolle spielen.

Stereopsis und Erregungsdynamik auf funktionellen Karten. Ein Grundproblem des stereoskopischen Tiefensehens ist das sogenannte „Korrespondenzproblem“, d.h. die Zuordnung von Bildelementen (*features*) im linken und rechten Bild, die Abbild des gleichen Dings in der Welt sind. Hat man dieses Problem gelöst, kann man den Abstand der Objekte durch Triangulation bestimmen. Wir werden hier kurz zwei neuronal motivierte Ansätze zur Lösung des Korrespondenzproblems vorstellen. Für weitergehende Fragestellungen sei auf das Kapitel „Bildverstehen“ verwiesen.

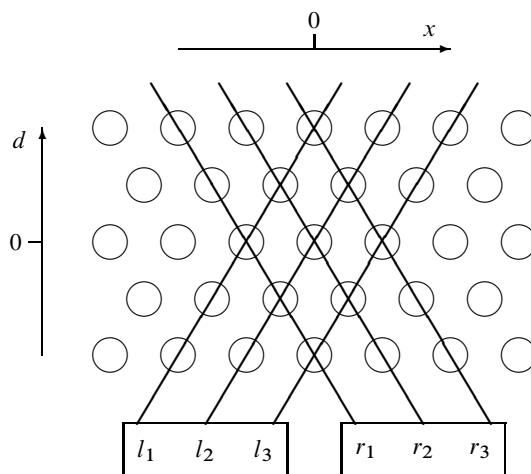


Abbildung 11.18: Ortskodierung der Disparität für kooperative Stereoalgorithmen. Die Bildmerkmale an den Positionen l_1, \dots, l_3 (linkes Halbbild) bzw. r_1, \dots, r_3 (rechtes Halbbild) erregen jeweils die Zellen mit $x - d/2 = r_i$ bzw. $x + d/2 = l_i$. Durch eine Erregungsdynamik wird ein Erregungsmuster erzeugt, bei dem zu jedem Bildmerkmal (möglichst) nur eine Zelle aktiv ist. Die intrinsischen Verbindungen sind nicht gezeigt. Zellen gleicher Disparität erregen sich gegenseitig, Zellen verschiedener Disparität am gleichen Ort hemmen sich.

Disparitätskarten. [43] kodieren Disparität d und Ort x eines Bildelementes in die Position einer Zelle in einer „Disparitätskarte“ (Abb. 11.18); die vertikale Komponente der Position kann hier weggelassen werden. Eine Zelle an der Position (x, d) wird durch Bildmerkmale an der Stelle $x - \frac{d}{2}$ im linken und $x + \frac{d}{2}$ im rechten Bild erregt. Das Korrespondenzproblem wird

dann auf folgende Dynamik abgebildet:

$$e_{x,d}^{t+1} = \phi \left(\sum_{(x',d') \in S(x,d)} e_{x',d'}^t - \epsilon \sum_{(x',d') \in O(x,d)} e_{x',d'}^t + e_{x,d}^0 \right). \quad (11.61)$$

Der Korrespondenzprozess benutzt zwei Ordnungsbedingungen, Eindeutigkeit und Stetigkeit. Die Eindeutigkeit (jedes Merkmal soll mit genau einem aus dem anderen Halbbild korrespondieren) wird durch die Hemmungsumgebung $O(x, d)$ implementiert. Ist etwa die Zelle (x, d) aktiv, so sollten die Zelle $(x \pm d', d - 2d')$ nicht gleichzeitig aktiv sein, da sie dasselbe Merkmal im linken bzw. rechten Halbbild verwenden. Man wählt daher z.B. $(x \pm d', d - 2d') \in O$. Die Stetigkeit wird durch die erregende Umgebung S berücksichtigt: benachbarte Zellen sollen gleiche Disparitäten haben. Die Konstante ϵ erlaubt eine Bewertung der beiden Bedingungen.

Der äußere Reiz $e_{x,d}^0$ ist eins, wenn sowohl an der Stelle $x - \frac{d}{2}$ im linken Bild als auch an der Stelle $x + \frac{d}{2}$ im rechten Bild ein Merkmal vorliegt. Diese Reizverteilung bleibt angelegt („clamped“ stimulus). ϕ ist eine sigmoide Nichtlinearität mit Wertebereich $[0, 1]$. Ohne sie kann sich die Kooperativität nicht entwickeln.

Eine Übersicht über ähnliche Verfahren und deren biologischen Erklärungswert geben [11].

Okulardominanzstreifen. Wir erinnern noch einmal an die Repräsentation der Bilder des linken und rechten Auges in den Okulardominanzstreifen des primären visuellen Cortexareals V1. Einen Vorschlag, wie man diese Datenstruktur zur Bestimmung von Disparitäten ausnutzen kann, machen [66]. Es handelt sich um einen intensitätsbasierten Ansatz, der im wesentlichen die Korrelation zweier in benachbarten Streifen abgelegter Bildteile ausnutzt. Es sei $s(x, y)$ das Ausgangsbild und

$$f(x, y) = s(x, y) + s(x - D, y) = s(x, y) * (\delta(x, y) + \delta(x - D, y)) \quad (11.62)$$

seine Verdopplung mit einer horizontalen Verschiebung D , die sowohl die Streifenbreite als auch etwaige Disparitäten enthält. Zur Bestimmung von D verwendet man nun statt der Autokorrelationsfunktion das sogenannte *Cepstrum* von f , das man erhält, wenn man das Leistungsdichtespektrum von f logarithmiert und davon dann noch einmal das Leistungsdichtespektrum¹⁸ bestimmt. Für diese spezielle Filteroperation fehlt eine biologische Motivation. Auf der anderen Seite zeigen psychophysische Ergebnisse, dass intensitätsbasierte Stereopsis möglich und ein Korrelationsmechanismus dafür wahrscheinlich ist.

Ortsvariante Bildverarbeitung: Erregungsdynamik auf topographischen Karten. Während Merkmalskarten stetige Anordnungen ganzer rezeptiver Felder in einer neuronalen Schicht darstellen, wird bei *topographischen* Karten lediglich der Zusammenhang zwischen dem Zentrum des rezeptiven Feldes und dem Ort in der Karte betrachtet. Es handelt sich also einfach um eine (stückweise) stetige und umkehrbare Funktion $\mathbb{R}^2 \rightarrow \mathbb{R}^2$. Kombiniert man eine solche Koordinatentransformation mit einer Filteroperation oder einer anderen translationsinvarianten Operation auf der Karte (intrinsische Verschaltung der kartierten Netzwerkschicht), so entstehen

¹⁸ Das Leistungsdichtespektrum ist die Fourier-Transformierte der Autokorrelationsfunktion. Im Vergleich zu gewöhnlichen Korrelationstechniken erweist sich das Cepstrum als weniger rauschempfindlich und robuster gegen Rotationen des Musters.

spezielle Operationen, die für eine Reihe von Problemen in der *ortsvarianten Bildverarbeitung* eingesetzt werden können (vgl. [38]).

Komplex logarithmische Karten. Die topographische Karte des primären visuellen Areals (V1) bei Affen wird in guter Näherung durch verschiedene Varianten der komplexen Logarithmusfunktion

$$\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \mathcal{L} \begin{pmatrix} x \\ y \end{pmatrix} := \begin{pmatrix} \frac{1}{2} \ln(x^2 + y^2) \\ \arctan \frac{y}{x} \end{pmatrix} \quad \text{für } x > 0 \quad (11.63)$$

beschrieben. Diese Funktion hat die interessante Eigenschaft, dass Drehstreckungen im Definitionsbereich („Retina“) in Translationen im Bildbereich überführt werden. Um das zu sehen, schreiben wir die Retinakoordinaten polar, d.h. $x = r \cos \varphi$, $y = r \sin \varphi$ und erhalten:

$$\begin{pmatrix} r \cos \varphi \\ r \sin \varphi \end{pmatrix} \mapsto \begin{pmatrix} \ln r \\ \varphi \end{pmatrix}. \quad (11.64)$$

Dreht man nun um einen Winkel α und streckt um einen Faktor λ , so wird das Bild um den Vektor $(\ln \lambda, \alpha)^T$ verschoben. Diese Eigenschaft des komplexen Logarithmus ist von vielen Autoren für die Konstruktion rotations- und größeninvarianter Musterklassifikatoren genutzt worden. Eine andere Anwendung ist die Transformation optischer Flussmuster, die bei reinen Translationsbewegungen des Beobachters entstehen. In diesem Fall weisen die projizierten Bewegungsvektoren im Bild radial von einem Expansionspunkt (*focus of expansion*) weg. Durch komplex logarithmische Kartierung um diesen Expansionspunkt herum werden die Flussvektoren parallel. Im nächsten Absatz werden wir eine Kartierung besprechen, die diese Eigenschaft ebenfalls aufweist, darüberhinaus aber auch noch die Längen der Flussvektoren vereinheitlicht.

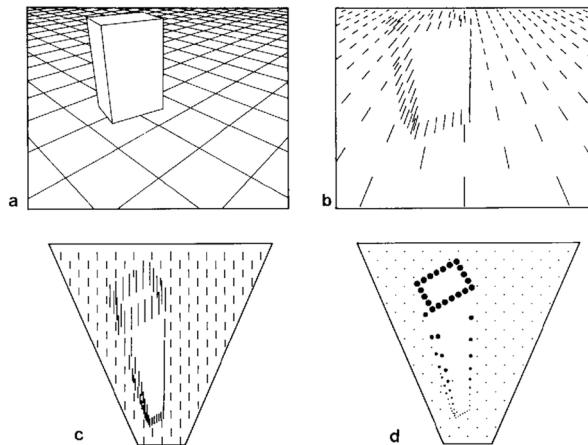


Abbildung 11.19: „Inverse Perspektive“ für visuelle Hinderniserkennung. **a.** Ausgangsbild. **b.** Projektion des Flussfelds für ein Hindernis in der Ebene. **c.** Flussfeld im invers perspektivisch kartierten Bild. **d.** Hindernisdetection durch Schwellwertbildung.

Hindernisvermeidung. Ein Beobachter bewege sich translatorisch auf einer Ebene mit der Geschwindigkeit $\mathbf{m} = (m_1, m_2, m_3)^T$. Das 3D-Geschwindigkeitsfeld relativ zum Beobachter

ist dann konstant. Die projizierten Geschwindigkeitsvektoren weisen daher alle auf den durch den Ausdruck $-\frac{1}{m_3}(m_1, m_2)^\top$ gegebenen Fluchtpunkt der Raumrichtung \mathbf{m} hin, der je nach Bewegungsrichtung als Expansions- bzw. Kontraktionspunkt bezeichnet wird.

Wir definieren jetzt ein *Hindernis* als etwas, das vom erwarteten Flussfeld der Ebene abweicht. Dies ist eine sehr allgemeine Definition eines Hindernisses, die man am besten mit den Bewegungsfreiheitsgraden des Beobachters begründet: angenommen, er kann sich nur in der Ebene bewegen, dann ist in der Tat alles, was aus der Ebene herausragt (oder über ihr schwebt), ein Hindernis. Im Sinne der in Abb. 11.20 gemachten Einteilung handelt es sich um eine verhaltensbezogene Klassifikation. Abb. 11.19 zeigt, dass ein solches Hindernis tatsächlich Abweichungen im Flussfeld erzeugt, doch sind diese schwierig zu detektieren, da die Ortsvarianz des Flussfeldes auf unterschiedliche Effekte zurückgehen kann:

- Die 3D Struktur der Umwelt. Das ist der gewünschte Effekt.
- Die perspektivische Verzerrung. Damit sind Effekte gemeint, die nur von der Position im Gesichtsfeld abhängen.

Inverse Perspektive und Optischer Fluss. Durch die Einführung einer geeigneten Koordinatentransformation vor der Bewegungsdetektion kann man die verschiedenen Ortsabhängigkeiten des Flussfeldes separieren (Abb. 11.19). Man erhält diese Transformationen, wenn man Bildpunkte $(x', y')^\top$ rückwärts in die durch die Bewegungsfreiheitsgrade des Beobachters aufgespannte Ebene hinausprojiziert. Das Flussfeld dieser Ebene wird dann konstant, während Hindernisse abweichende Verschiebungsvektoren aufweisen. Mit diesem Verfahren ist man in der Lage, natürliche Szenen schnell in Hindernisse und passierbare Bereiche zu segmentieren, ohne eine eigentliche Objekterkennung durchführen zu müssen (vgl. [37]).

11.5 Mustererkennung mit neuronalen Netzen

Optimale Klassifikatoren. Für bestimmte Standardprobleme der Mustererkennung liegen seit langem Optimallösungen vor, die auch durch den Einsatz neuronaler Netze nicht weiter verbessert werden können. Grundproblem ist der „Empfang“ eines statistisch gestörten („verrauschten“) Musters aus einer Klasse bekannter Muster ω_i (Abb. 11.20a). Man geht dann so vor, dass man für jedes empfangene Muster einen Satz von Merkmalen (*features*) bestimmt, die die Grundlage für die Entscheidung bilden. Sind die Muster z.B. Bilder, so können die Merkmale im einfachsten Fall die Grauwerte einzelner Pixel sein. In der Regel führt die Verwendung abgeleiteter Merkmale (z.B. Kantenelemente, Koeffizienten verschiedener Integraltransformationen oder der Hauptachsentransformation etc.) jedoch zu besseren Ergebnissen.

Der Musterklassifikator berechnet zunächst für jeden Eingangsvektor (Zufallsvariable \mathbf{s}) einen Satz von „Diskriminationsfunktionen“, zum Beispiel die *likelihood* Werte $g_i(\mathbf{s}) = p(\mathbf{s}|\omega_i)P(\omega_i)$. Der Klassifikator wählt dann die Hypothese i mit dem höchsten g_i -Wert aus. Die drei Komponenten – Inputvektor, Diskriminationsfunktion und Maximumsbildung – gleichen einem dreischichtigen Perzeptron (zur Theorie der Mustererkennung vgl. etwa [10]).

Im Zentrum der klassischen Testtheorie steht die Verteilung einer statistischen Störung. Zur Ermittlung dieser Verteilungen, aber auch zur Definition der Mustervektoren ω_i (*Konzeptbildung*)

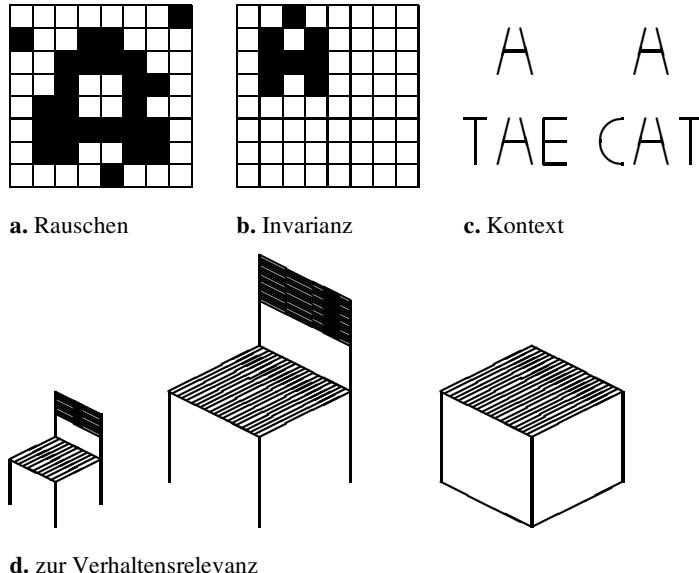


Abbildung 11.20: Typen von Mustererkennungsproblemen. **a.** Statistisch verrausches Muster (Buchstabe A). **b.** Größen- und Lageinvarianz. **c.** Gleiche Muster (oben) müssen je nach Kontext verschieden klassifiziert werden (unten). **d.** Objekte als „Utensilien“ für eine Verhaltensweise („Hinsetzen“). Größeninvarianzfähig zur Identifizierung der beiden linken Objekte (Puppenstuhl, Stuhl), während nur die beiden rechten (Stuhl, Kiste) zum Daraufsetzen geeignet sind.

werden dabei auch Lernverfahren benutzt. Neuronale Netzwerke sind vor allem bei solchen Problemen im Vorteil, wo die Grundidee „Signal = Muster + Rauschen“ nicht mehr zutrifft. Will man z.B. Objekte in Bildern erkennen, so ist die Verschiebung des Bildes um ein paar Pixel oder die Drehung des Objektes im Raum kaum als statistische Störung des Inputvektors zu beschreiben; eine Übersicht gibt Abb. 11.20. Man kann in solchen Fällen versuchen, das Problem in den Kodierungsschritt zu verlagern, indem man translations- und rotationsinvariante Merkmale (z.B. Zentralmomente) verwendet. In anderen Fällen können sich die statistischen Eigenschaften der Störung schnell ändern, so dass adaptive Verfahren wie z.B. neuronale Netze sinnvoll erscheinen.

Lösungsbeispiele für das „XOR“-Problems. Wie bereits in Abs. 11.3.3 beschrieben ist das „XOR“-Problem die einfachste Variante eines linear nicht separierbaren Datensatzes. Zur Lösung dieses Problems genügt allerdings bereits ein dreischichtiges Netzwerk, mit drei Neuronen in der verborgenen Schicht. Das zugehörige MLP ist in Abb. 11.12 dargestellt, die Trennfläche in Abb. 11.21a.

Ähnlich kann man im Fall der verallgemeinerten Diskriminationsfunktion verfahren. Verwendet man als Merkmalstransformation die Radialbasen,

$$\Phi(\mathbf{x}) = [1, e^{-\|\mathbf{x}-\mathbf{s}_1\|^2}, e^{-\|\mathbf{x}-\mathbf{s}_2\|^2}] \quad \text{mit } \mathbf{s}_1 = [1, 0]^\top \text{ und } \mathbf{s}_2 = [0, 1]^\top ,$$

sowie den Gewichtsvektor $\mathbf{w} = [0, 1, 1]^\top$, so ergibt sich eine Lösung bei der jedes Element der Trainingsmenge als Prototyp auftritt (vgl. Abb. 11.21b).

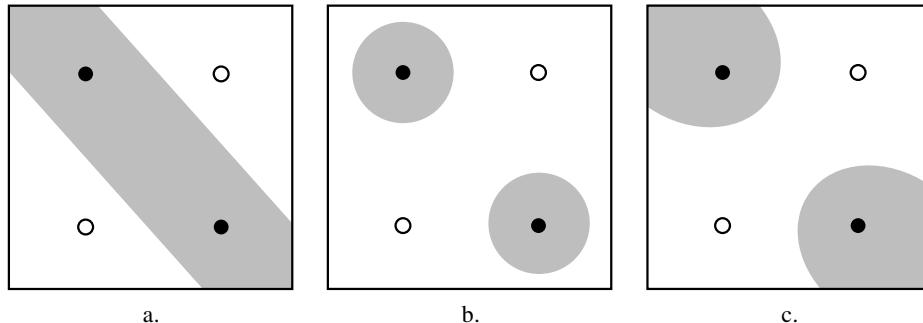


Abbildung 11.21: Lösungsmöglichkeiten für das „XOR“-Problem. a. Dreischichtiges Perzeptron b. Radialbasisnetzwerk mit zwei Komponenten c. Verallgemeinerte Diskriminationsfunktion auf Basis der Monome 2.ter Ordnung (quadratisch separierbar). Die Entscheidungsregionen für die Klasse (●) sind grau schattiert. In a. und c. sind die Entscheidungsregionen scharfbegrenzt; in b. entspricht der grau schattierte Bereich der ersten Standardabweichung einer Gaußfunktion mit Varianz Eins.

Eine weitere Variante erhält man, wenn man als Merkmalstransformation die Menge der Monome bis zum Grad zwei verwendet:

$$\Phi(\mathbf{x}) = [1, x_1, x_2, x_1x_2, x_1^2, x_2^2]^\top .$$

Eine hieraus resultierende Lösung ist in Abb. 11.21c schematisch dargestellt.

Hierarchische Netzwerke. In Analogie zur Bildverarbeitung werden rezeptive Felder häufig als Merkmalsdetektoren aufgefasst, die Informationen über lokale Bildelemente („features“) extrahieren (vgl. die klassischen Arbeiten von [25]). Durch Kopplung von Neuronen mit unterschiedlichen rezeptiven Feldern entstehen neue rezeptive Felder, welche dann komplexere Muster repräsentieren. Auf diese Weise kann man auch Detektoren konstruieren, die eine zunehmend stärkere Invarianz¹⁹ gegenüber geometrischen und photometrischen Transformationen aufweisen.

Setzt man die Kopplung über mehrere Ebenen hinweg fort, so entstehen hierarchische Systeme, welche insbesondere im Bereich der ansichtsbasierten Objekterkennung untersucht wurden. Eines der bekanntesten Modelle ist das *Neocognitron*, ein vielschichtiges Perzeptron, bestehend aus einer Input-Schicht und drei oder vier zweischichtigen Modulen mit jeweils ähnlichem Aufbau. Jedes Modul besteht aus einer S- und einer U-Schicht, die wiederum eine Anzahl von Zellgruppen (*planes*) enthalten. Die Zellen der S-Schichten erhalten über modifizierbare Synapsen Input von allen Zellen der C-Schicht des vorhergehenden Moduls (bzw. von der Inputschicht). Durch Wettbewerbslernen in den Gruppen oder durch überwachtes Lernen werden die Inputgewichte der S-Zellen so modifiziert, dass jede Gruppe einer S-Schicht auf ein bestimmtes Merkmal des Musters reagiert, wobei die Position in der Gruppe der Position des Merkmals in der Inputschicht entspricht. Beim Übergang auf die C-Schicht werden S-Zellen mit leicht verschiedener Position zusammengefasst, so dass jeweils eine gewisse Lageinvarianz erzeugt wird. Die Synapsen S→C sind nicht modifizierbar. Das Netzwerk kann trainiert werden, handgeschriebene Buchstaben zu klassifizieren. (Vgl. [17] u. [18]).

¹⁹ Der Begriff Invarianz ist eigentlich nicht ganz korrekt. Streng genommen handelt es sich hier um Robustheit und nicht um Invarianz im mathematischen Sinn.

In diesem Kontext wurden in jüngster Zeit Netzwerke untersucht, welche sich durch eine große Hierarchietiefe („Deep Computing“) auszeichnen. Wie bereits in Abschnitt 11.3.5 beschrieben, sind gradientenbasierte Lernverfahren in der Regel nicht für das Training komplexer Netzwerke geeignet. Eine Möglichkeit dieses Problem zu umgehen besteht darin eine Kombination aus unüberwachten und überwachten Lernmethoden zu verwenden. Dabei wird durch ein unüberwachtes Lernverfahren eine initiale, aber ungenaue Lösung bestimmt, welche dann mittels überwachten Lernen verfeinert wird. Basierend auf dieser Idee wurden z.B. Lernverfahren für komplexe, einschichtige Radialbasisnetzwerke entwickelt (s. hierzu [20]). Weiterhin lassen sich damit auch komplexe Netzwerke mit großer Hierarchietiefe trainieren, vorausgesetzt das Training erfolgt Ebene für Ebene [22]. Mit diesem Ansatz konnten sehr gute Ergebnisse bei klassischen Mustererkennungsaufgaben, wie beispielsweise der Handschriftenerkennung oder der Verarbeitung natürlicher Sprache erzielt werden. Eine ausführliche Einführung in das Thema wird in [8] gegeben.

Während der Ansatz hierarchischer Merkmalsintegration für frühe Verarbeitungsstufen durchaus angemessen erscheint, führt dieser Ansatz bei höheren Verarbeitungsstufen zu Schwierigkeiten. Werden die *features* einfach immer komplizierter, bis das Feuern einer Zelle für das Erkennen einer ganzen Szenerie steht? Für diese Idee spricht die Tatsache, dass die rezeptiven Felder tatsächlich größer werden und dass sog. Gesichterzellen, die hochspezifisch auf Gesichter reagieren, wirklich gefunden wurden. Dagegen spricht ein logisches Problem: je detaillierter die Reizbeschreibung wird, für die die Erregung einer Zelle steht, umso mehr solcher Zellen werden gebraucht. Wenn tatsächlich eine „Großmutterzelle“ dafür zuständig ist, die Gegenwart der alten Dame anzudeuten, was passiert, wenn sie den Hut abnimmt?

NETtalk: Phonem-Klassifikation aus Buchstabengruppen.

Ein besonders schwieriges Mustererkennungsproblem ist die Zuordnung von Graphemen zu Phonemen, die Voraussetzung für die Erzeugung gesprochener Sprache aus geschriebenen Texten ist. Insbesondere in der englischen Sprache benötigt man für diese Zuordnung ausgiebige Kontext-Information (Zum Kontext vgl. Abb. 11.20c). Ein Beispiel ist das Kunstwort „ghoti“, das man wie „fish“ aussprechen müsste wenn man etwa *gh* wie in *to cough*, *o* wie in *women* und *ti* wie in *nation* zusammensetzt.

[56] konstruieren ein *back-propagation* Netzwerk (Abs. 11.3.5), das aus drei vorwärts verkoppelten Schichten besteht:

1. Einer Inputschicht aus sieben Gruppen von jeweils 29 Zellen. Wird ein Reizmuster (sieben Buchstaben) präsentiert, so wird in jeder Gruppe die für den entsprechenden Buchstaben reservierte Zelle aktiviert (26 Buchstaben + 3 Satzzeichen = 29 Zellen). In jeder Gruppe ist also genau eine Zelle aktiv.
2. Einer Zwischenschicht aus 80 Assoziationszellen („*hidden units*“).
3. Einer Ausgangsschicht aus 23 Zellen, die bestimmte artikulatorische Merkmale der gewünschten Phoneme kodieren (Lautschrift). Hier können also mehrere Zellen gleichzeitig aktiv sein. Der Ausgang gibt die Aussprache des mittleren der sieben Eingangsbuchstaben wieder.

Mit einem nachfolgenden Klassifizierungsschritt lernt das Netzwerk, etwa 95 % der Worte aus einem 1000 Worte umfassenden Trainingssatz korrekt in eine vereinfachte Lautschrift zu übersetzen. Eine Vermehrung der *hidden units* bringt keine wesentliche Verbesserung mehr. Die

Phonem-Vektoren wurden an einen Spracherzeuger („DECtalk“) weitergegeben und können von Versuchsperson meist problemlos verstanden werden.

11.6 Schlussbemerkung

Neuronale Netze und biologische Informationsverarbeitung. Eine Übersicht über die Theorie und Anwendung neuronaler Netze wird dadurch erschwert, dass die vorliegenden Arbeiten (wie übrigens auch die Lehrbücher) eine ganze Reihe verschiedener und zum Teil unvereinbarer Ziele verfolgen, z.B.:

1. Modellierung neurophysiologischer Daten,
2. Übertragung der vom Nervensystem tatsächlich verwendeten Lösungen auf technische Anwendungen,
3. Entwicklung einer Theorie künstlicher neuronaler Netze,
4. Entwicklung eines neuronal motivierten, verteilten oder parallelen Programmierstils,
5. Übertragung vorhandener Theorien für Systeme mit vielen, miteinander wechselwirkenden Elementen auf Nervennetze (Spin-Glas Modelle).

Der Schwerpunkt der hier vorliegenden Einführung liegt deutlich in den Bereichen 2 und 3, die die Grundlage für weitergehende Anwendungen bilden. Anwendungen können nur dann funktionieren, wenn das, was angewendet werden soll, in diesem Fall also die neuronale Informationsverarbeitung, auch hinreichend gut verstanden ist. Umgekehrt kann man vom Gehirn sicher nicht lernen, wie die Lernregel für *back-propagation* optimiert werden kann oder wie man Muster in einem Spin-Glas speichert, weil es diese Verfahren selbst nicht benutzt. Wie überall setzen auch im Bereich der natürlichen Informationsverarbeitung funktionierende Anwendungen die empirische Erforschung der Vorbilder voraus.

11.7 Weiterführende Literatur

Gesamtdarstellungen. Die Leistungsfähigkeit künstlicher neuronaler Netze für realistische Informationsverarbeitungsprobleme ist umfassend in den Büchern [10, 20] dargestellt. Eine enzyklopädische Übersicht in vielen von Spezialisten verfassten Einzelartikeln versucht [34].

Aufsatzsammlungen. Viele ältere Originalarbeiten sind in den Sammlungen [Anderson & Rosenfeld 1988, Anderson *et al.* 1990, Shaw & Palm 1988] zusammengetragen.

Literaturverzeichnis

- [1] E.H. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines*. John Wiley & Sons, 1989.
- [2] E.H. Adelson and J.R. Bergen. *Computational Models of Visual Processing*, chapter The plenoptic function and the elements of early vision, pages 3–20. Cambridge, MA: MIT Press, 1991.

- [3] S. Amari. A new learning algorithm for blind signal separation. *Advances in Neural Information Processing Systems*, 1996.
- [4] J.A. Anderson and Hrsg. E. Rosenfeld. *Neurocomputing. Foundations of Research*. The MIT Press, Cambridge, Ma., 1988.
- [5] J.A. Anderson, A. Pellionisz, and Hrsg. E. Rosenfeld. *Neurocomputing 2. Directions for Research*. The MIT Press, Cambridge, Ma., 1990.
- [6] M.F. Bear, B.W. Connors, and M.A. Paradiso. *Neurowissenschaften. Ein grundlegendes Lehrbuch für Biologie, Medizin und Psychologie*. Spektrum Verlag, 3. Aufl., 2009.
- [7] A.J. Bell and T.J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, November 1995.
- [8] Y. Bengio. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2:1–127, 2009
- [9] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
- [10] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2009.
- [11] R. Blake and H.R. Wilson. Neural models of stereoscopic vision. *Trends in Neurosciences*, 14:445–452, 1991.
- [12] V. Braitenberg and A. Schüz. *Anatomy of the Cortex. Statistics and Geometry of neural connectivity*. 2. Edition, Springer Verlag, Berlin, 1998.
- [13] P.S. Churchland and T.J. Sejnowski. *The Computational Brain*. MA:MIT Press., 1999.
- [14] T.M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *Electronic Computers, IEEE Transactions on*, EC-14(3):326 –334, june 1965.
- [15] S. Ferrari, F. Bellocchio, V. Piuri, and N. A. Borghese. A hierarchical rbf online learning algorithm for real-time 3-d scanner. *IEEE Transactions on Neural Networks*, 21(2):275–285, 2010.
- [16] B. Fritzke. Growing cell structures – a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994.
- [17] K. Fukushima, S. Miyake, and T. Ito. Neocognitron: A neural network model for a mechanism of visual pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:826–834, 1983.
- [18] K. Fukushima. Neocognitron for handwritten digit recognition. *Neurocomputing*, 51(0):161–180, 2003.
- [19] K. Schulten, H. Ritter, and T. Martinetz. *Neuronale Netze. Eine Einführung in die Neuroinformatik selbstorganisierter Netzwerke*. Addison-Wesley, Bonn etc., 2. Aufl., 1991.
- [20] S. Haykin. *Neural Networks and Learning Machines*. Prentice Hall International, 3rd rev. ed. edition, 2009.
- [21] R. Hecht-Nielsen. *Neurocomputing*. Addison-Wesley, Reading, MA, 1990.
- [22] E.G. Hinton, S. Osindero, and Y. Teh. A Fast Learning Algorithm for Deep Belief Nets *Neural Computation*, 18:1527–1554, 2006
- [23] M. W. Hirsch. Convergent activation dynamics in continuous time networks. *Neural Networks*, 2(5):331–349, 1989.
- [24] G. B. Huang, P. Saratchandran, and N. Sundararajan. A generalized growing and pruning rbf (ggap-rbf) neural network for function approximation. *IEEE Transactions on Neural Networks*, 16(1):57–67, 2005.
- [25] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology (London)*, 160:106–154, 1962.

- [26] A. Hyvärinen and E. Oja. A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9(7):1483–1492, October 1997.
- [27] J. Karhunen, E. Oja, L. Wang, R. Vigario, and J. Joutsensalo. A class of neural networks for independent component analysis. *Neural Networks, IEEE Transactions on*, 8(3):486–504, may 1997.
- [28] B. Katz. *Nerv, Muskel und Synapse*. G. Thieme Verlag, Stuttgart, 1971.
- [29] J. J. Koenderink and A. J. van Doorn. Representation of local geometry in the visual system. *Biol. Cybern.*, 55:367–375, March 1987.
- [30] T. Kohonen. *Self-Organization and Associative Memory*. Springer Verlag, Berlin, 2. Aufl., 1988.
- [31] A. N. Kolmogorov. *On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables*. Providence, R.I.: American Mathematical Society, 1961.
- [32] M. Kristan, D. Skočaj, and A. Leonardis. Online kernel density estimation for interactive learning. *Image and Vision Computing*, 28(7):1106–1116, 2010.
- [33] T.S. Lee. Image representation using 2d gabor wavelets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10), 1996.
- [34] Hrsg. M.A. Arbib. *The Handbook of Brain Theory and Neural Networks*. The MIT Press, Cambridge, Ma., 1995.
- [35] D. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.
- [36] D. MacKay. Comparison of approximate methods for handling hyperparameters. *Neural Computation*, 11(5):1035–1068, July 1999.
- [37] H.A. Mallot, H.H. Bülfhoff, J.J. Little, and S. Bohrer. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological Cybernetics*, 64:177–185, 1991.
- [38] H.A. Mallot, W. von Seelen, and F. Giannakopoulos. Neural mapping and space-variant image processing. *Neural Networks*, 3:245–263, 1990.
- [39] H.A. Mallot, J. Kopecz, and W. von Seelen. *Neuroinformatik als empirische Wissenschaft*. 1992.
- [40] H.A. Mallot and F. Giannakopoulos. Population networks: a large-scale framework for modelling cortical neural networks. *Biological Cybernetics*, 75:441–452, 1996. 10.1007/s004220050309.
- [41] C. Malsburg. Self-organization of orientation sensitive cells in the striate cortex. *Biological Cybernetics*, 14:85–100, 1973.
- [42] D. Marr. *Vision*. W. H. Freeman, San Francisco, 1982.
- [43] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, 194:283–287, 1976.
- [44] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4):115–133, December 1943.
- [45] P.L. McGeer, J.C. Eccles, and E.G. McGeer. *Molecular Neurobiology of the Mammalian Brain*. Plenum Press, New York and London, 2. Aufl., 1987.
- [46] M.L. Minsky and S.A. Papert. *Perceptrons, expanded edition*. The MIT Press, Cambridge, MA., 1988.

- [47] P. L. Narasimha, W. H. Delashmit, M. T. Manry, J. Li, and F. Maldonado. An integrated growing-pruning method for feedforward network training. *Neurocomputing*, 71:2831–2847, 2008.
- [48] E. Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273, 1982. 10.1007/BF00275687.
- [49] E. Oja. Neural networks, principal components, and subspaces. *International Journal of Neural Systems*, 1:61–68, 1989.
- [50] D. T. Pham. Blind separation of instantaneous mixture of sources via an independent component analysis. *Signal Processing, IEEE Transactions on*, 44(11):2768 –2779, nov 1996.
- [51] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.
- [52] C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
- [53] E. Rolls and G. Deco. *Computational Neuroscience of Vision*. Oxford University Press, 2002.
- [54] T. D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2(6):459–473, 1989.
- [55] F. Schwenker, H.A. Kestler, and G. Palm. Three learning phases for radial-basis-function networks. *Neural Networks*, 14(4–5):439–458, 2001.
- [56] T. J. Sejnowski and C. R. Rosenberg. Nettalk: a parallel network that learns to read aloud. In James A. Anderson and Edward Rosenfeld, editors, *Neurocomputing: foundations of research*, pages 661–672. MIT Press, Cambridge, MA, USA, 1988.
- [57] G.L. Shaw and Hrsg. G. Palm. *Brain Theory Reprint Volume*. World Scientific, Singapore, New Jersey, Hong Kong, 1988.
- [58] H.T. Siegelmann. *Neural Networks and Analog Computation: Beyond the Turing Limit*. Birkhauser, Boston, 1998
- [59] B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. London: Chapman & Hall/CRC, 1998.
- [60] B.G. Tabachnick and L.S. Fidell. *Using Multivariate Statistics*. Pearson Education, 2005.
- [61] J. von Uexküll. *Streifzüge durch die Umwelten von Tieren und Menschen*. Rowohlt, Hamburg, 1956.
- [62] B. Widrow and M.E. Hoff. Adaptive switching circuits. *Convention record, IRE WESCON*, 1960. New York: IRE.
- [63] H. R. Wilson and J. D. Cowan. A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue. *Biological Cybernetics*, 13:55–80, 1973. 10.1007/BF00288786.
- [64] L. Wiskott and T. J. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, April 2002.
- [65] L. Xu, A. Krzyzak, and A. Yuille. On radial basis function nets and kernel regression: Statistical consistency, convergence rates, and receptive field size. *Neural Networks*, 7(4):609–628, 1994.
- [66] Y. Yeshurun and E.L. Schwartz. Cepstral filtering on a columnar image architecture: A fast algorithm for binocular stereo segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:759–767, 1989.

12 Maschinelles Lernen und Data Mining

Stefan Wrobel, Thorsten Joachims und Katharina Morik

Das Forschungsgebiet des maschinellen Lernens (vom englischen *machine learning*) beschäftigt sich, ebenso wie die eng verwandten Nachbargebiete des Data Mining bzw. der Wissenentdeckung in Datenbanken (*Knowledge Discovery in Databases, KDD*) mit der *computergestützten Modellierung und Realisierung von Lernphänomenen*, der Extraktion von Wissen aus großen Datenbeständen und der Realisierung adaptiver Systeme. Wie kaum ein anderes Gebiet der künstlichen Intelligenz haben maschinelles Lernen und Data Mining im Verlauf der letzten Jahrzehnte ihren Weg in praktische Anwendungen in Industrie und Wirtschaft bis hin zum privaten Bereich gefunden. So sorgen Lernalgorithmen zum Beispiel dafür, dass Suchmaschinen ihre Ergebnisse gut sortieren (wie z.B. bei Google [60]), sie lernen unsere Vorlieben für Kinofilme und andere Produkte (siehe z.B. [12], entlarven den Missbrauch von gestohlenen Kreditkarten und andere betrügerische Aktivitäten [71]), und sie filtern Spam aus unserer Email [35].

Das folgende Kapitel gibt eine Einführung in einige der wichtigsten Themen, theoretischen Grundlagen, Verfahren und Anwendungen des „maschinellen Lernens“ und des Data Mining. Es gliedert sich in drei Abschnitte, die jeweils eine Lernaufgabe und die dazugehörigen Lernmethoden einführen.

Die am meisten untersuchte und angewandte Lernaufgabe des maschinellen Lernens ist das **überwachte Lernen** (engl. *supervised learning*). Es wird im Abschnitt 12.2 eingeführt und beschreibt das Lernen von Funktionen aus Beispielen. Der Lernalgorithmus erhält Beispiele des funktionalen Zusammenhangs (z.B. Emails die von Hand als Spam oder Ham klassifiziert wurden) und soll daraus eine allgemeine Funktion lernen. Wir beschreiben verschiedene Varianten der überwachten Lernaufgabe, zusammen mit prototypischen Verfahren zu ihrer Lösung: Entscheidungsbäume (Abschnitt 12.3), instanzbasiertes Lernen (Abschnitt 12.4), und die Stützvektormethode (Abschnitt 12.5). Da den neuronalen Netzen in diesem Buch ein eigenes Kapitel gewidmet ist, werden sie hier nicht noch einmal behandelt. Es folgt ein Abschnitt, der sich mit der grundlegenden theoretischen Lernbarkeit von Funktionen aus der Sicht des sog. PAC-Lernens (Abschnitt 12.6) beschäftigt. Ähnlich grundlegend ist Abschnitt 12.7, der sich aus logischer Perspektive mit dem Lernen aus strukturierten Daten beschäftigt.

Die zweite Lernaufgabe ist das **unüberwachte Lernen** (engl. *unsupervised learning*). Während beim überwachten Lernen der Lernalgorithmus Beispiele eines bestimmten funktionalen Zusammenhangs erhält, bekommen unüberwachte Lernalgorithmen keine klassifizierten Beispiele. Das unüberwachte Lernen ist eher deskriptiver Natur und versucht interessante Strukturen in unklassifizierten Daten zu finden. Die entdeckten Strukturen sind dabei oft *lokal*, d.h. sie charakterisieren nur einen Teil der untersuchten Objekte, bei denen besondere Auffälligkeiten entdeckt wurden. Unüberwachte Lernenverfahren unterscheiden sich insbesondere in der Art

der Struktur, nach der in den Daten gesucht wird. In Abschnitt 12.8 erörtern wir Verfahren zum assoziativen Regellernen, gefolgt von der Aufgabe der Subgruppenentdeckung in Abschnitt 12.9 und der Clusteranalyse in Abschnitt 12.10. Diese Verfahren finden insbesondere bei der Wissensentdeckung in Datenbanken im Data Mining ihre Anwendung, mit dem Ziel interessante Abhängigkeiten in grossen Datenmenge (z.B. Abverkaufsdaten, Erfolg von Marketing-Aktionen) zu finden und kompakt zusammenzufassen.

Als dritte Lernaufgabe diskutieren wir das **Verstärkungslernen** (engl. *reinforcement learning*) (Abschnitt 12.11). Es behandelt das Problem des Erlernens optimaler Handlungsvorschriften aus Erfahrungen, wie sie zum Beispiel für Roboter und andere autonome Agenten, die erfolgreich in der Welt agieren sollen, gebraucht werden. Das Verstärkungslernen hat Ähnlichkeiten zum überwachten Lernen, da der Agent positive und negative Belohnungen für sein Verhalten erhält. Ein wesentlicher Unterschied ist jedoch, dass die Belohnungen nicht einzelnen Beispielen zugeordnet sind, sondern das Ergebnis einer langen Folge von Aktionen sein können.

Für alle drei Lernaufgaben haben wir für dieses Buch besonders grundlegende Algorithmen ausgewählt, an denen sich die wichtigsten Prinzipien gut erläutern lassen. Für jeden der vorgestellten Grundalgorithmen sind in den vergangenen Jahren natürlich unzählige Varianten entstanden, die sich durch universellere Verwendbarkeit, bessere Lernleistung oder höhere Effizienz auszeichnen, hierzu geben wir in Abschnitt 12.12 entsprechende Literaturhinweise auf vertiefende Lehrbücher. Ebenfalls in diesem Abschnitt verweisen wir auf Bücher, die zur Ergänzung des hier behandelten Stoffes dienen können, insbesondere bei Themen wie Lernverfahren für probabilistische, insbesondere graphische Modelle, weitere Verfahren für unüberwachtes und halbüberwachtes (semi-supervised) Lernen, und dem grossen Gebiet des Lernens mit strukturierten Daten, das wir hier nur aus logischer Perspektive einführen.

Das Thema des Lernens mit neuronalen Netzen, das üblicherweise ebenfalls dem maschinellen Lernen zugerechnet wird, wird in diesem Buch aus etwas anderer Perspektive im Kapitel 11 behandelt.

12.1 Was ist maschinelles Lernen

Wie oben ausgeführt bezeichnet man als „maschinelles Lernen“ ein Forschungsgebiet, das sich mit der *computergestützten Modellierung und Realisierung von Lernphänomenen* beschäftigt. Dabei ist es durchaus nützlich, sich bei der Interpretation des für das Gebiet zentralen Schlüsselbegriffs „Lernen“ zunächst von der intuitiven Bedeutung dieses Wortes leiten zu lassen. Wie wir nun sehen werden, ist es nämlich nicht einfach, eine zugleich präzise und umfassende intensionale Definition von „Lernen“ zu geben.

12.1.1 Intensionale Definitionsversuche

Es ist nicht einfach, unser intuitives Verständnis von Lernen in eine präzise und umfassende Definition zu fassen, wie zwei klassische Definitionen von „Lernen“ belegen. So definiert Herbert Simon [85]:

Lernen ist jeder Vorgang, der ein System in die Lage versetzt, bei der zukünftigen Bearbeitung derselben oder einer ähnlichen Aufgabe diese besser zu erledigen.

Diese Definition wurde unter anderem von Ryszard Michalski, einem weiteren Pionier des maschinellen Lernens, kritisiert, weil das Ziel beim Lernen nicht immer offensichtlich ist, und es auch Lernen ohne konkretes vorgefasstes Ziel geben kann, z. B. wenn man durch eine Stadt läuft und sich merkt, wo sich die Bibliothek befindet, ohne konkret vorzuhaben, die Bibliothek aufzusuchen. Auch ist nicht jede verbesserte Aufgabenlösung Lernen (ein geschärftes Messer schneidet besser, hat aber nicht gelernt). Als alternative Definition hat Michalski daher vorschlagen [61]:

Lernen ist das Konstruieren oder Verändern von Repräsentationen von Erfahrungen.

Diese Definition ist einerseits bereits so allgemein, dass sie kaum noch nützlich ist, und kann andererseits immer noch als zu eng kritisiert werden, denn nicht immer müssen beim Lernen Repräsentationen im engeren Sinne im Spiel sein.

Wesentlich präziser, allerdings deutlich weniger umfassend, ist demgegenüber die Zurückführung auf eine der drei klassischen logischen Schlussformen, nämlich den induktiven Schluss. Die drei Schlussformen werden gerne anhand der Sterblichkeit von Sokrates illustriert:

Deduktion: Aus „Alle Menschen sind sterblich.“ und „Sokrates ist ein Mensch.“ schließe „Sokrates ist sterblich.“ (wahrheitserhaltend).

Induktion: Aus „Sokrates ist ein Mensch.“ und „Sokrates ist sterblich.“ schließe „Alle Menschen sind sterblich.“ (falschheitserhaltend bzgl. der zweiten Aussage).

Abduktion: Aus „Sokrates ist sterblich.“ und „Alle Menschen sind sterblich.“ schließe „Sokrates ist ein Mensch.“ (falschheitserh. bzgl. der ersten Aussage).

Während wohl die meisten Forscher zustimmen würden, dass der induktive Schluss vom Besonderen auf das Allgemeine ein wesentliches Element des maschinellen Lernens ist, so schließt auch diese Definition viele Spielarten des Lernens aus und ist nicht so präzise, dass mit ihrer Hilfe genau entschieden werden könnte, ob ein bestimmtes Phänomen ein Lernphänomen ist.

12.1.2 Extensionale Definition über Lernaufgaben

Alternativ kann man deshalb versuchen, das maschinelle Lernen extensional über die einzelnen Typen von *Lernaufgaben* zu definieren, mit denen man sich (bisher) in diesem Gebiet beschäftigt. Dies hat den Vorteil, dass eine einzelne Lernaufgabe in informatiktypischer Manier sehr präzise definiert werden kann.

Eine *Lernaufgabe* wird definiert durch eine Beschreibung der dem lernenden System zur Verfügung stehenden Eingaben (ihrer Art, Verteilung, Eingabezeitpunkte, Darstellung und sonstigen Eigenschaften), der vom lernenden System erwarteten Ausgaben (ihrer Art, Funktion, Ausgabezeitpunkte, Darstellung und sonstigen Eigenschaften) und den Randbedingungen des Lernsystems selbst (z.B. maximale Laufzeiten oder Speicherverbrauch).

Innerhalb einer jeden so definierten Lernaufgabe ist also genau spezifiziert, was es bedeutet, wenn ein System lernt: es lernt erfolgreich genau dann, wenn es in der Lage ist, bei Eingaben, die den Spezifikationen entsprechen, unter den geforderten Randbedingungen Ausgaben mit den gewünschten Eigenschaften zu erzeugen.

Verwenden wir die bekannteste und am häufigsten betrachtete Lernaufgabe, das *Lernen aus klassifizierten Beispielen* zur ersten (noch unpräzisen und idealisierten) Illustration. Bei dieser Lernaufgabe wird angenommen, dass es eine unbekannte Funktion (nennen wir sie f) gibt, die Objekten einer bestimmten Grundmenge einen Zielwert zuordnet. Die Objektmenge könnte beispielsweise alle Kreditkartentransaktionen eines bestimmten Anbieters sein, und die gesuchte Funktion ordnet jeder Transaktion entweder den Wert „in_ordnung“ oder den Wert „betrügerisch“ zu. Dem lernenden System wird nun eine Menge von *Beispielen* der gesuchten Funktion zur Verfügung gestellt. Jedes Beispiel besteht erstens aus einer Objektbeschreibung, und zweitens dem Zielwert, den f diesem Objekt zuordnen würde. Lernen bedeutet in dieser Klasse von Lernaufgaben, das das lernende System mit den gegebenen Beispielen als Eingabe eine Funktionsbeschreibung h (die „Hypothese“) ausgibt, mit der bei zukünftig vorgelegten Instanzen, also Objektbeschreibungen mit noch unbekanntem Zielwert, der Zielwert von f möglichst gut vorhergesagt werden kann.

Wir werden auf diese und andere Lernaufgaben im Verlauf des Kapitels noch genauer eingehen, ihre verschiedenen Präzisierungen und entsprechende Lernverfahren kennenlernen, und dabei also das Gebiet des maschinellen Lernens sozusagen *en passant* extensional definieren. Dabei bleibt die Schwäche jeder extensionalen Definition natürlich erhalten: dieses Kapitel kann nicht alle bekannten Lernaufgaben behandeln, und die Weiterentwicklung der Forschung im maschinellen Lernen ist gerade auch durch die Neu- und Redefinition von Lernaufgaben gekennzeichnet.

12.1.3 Motivationen und Anwendungen

Je nach Motivation der Forschenden für die computergestützte Beschäftigung mit Lernphänomenen ergeben sich unterschiedliche Blickrichtungen auf das Gebiet. Aus der kognitionsorientierten Perspektive ist man daran interessiert, mit Hilfe von Computermodellen das menschliche Lernen besser zu verstehen und abzubilden. Aus der theoretischen Perspektive ist man daran interessiert, grundsätzlich zu verstehen, welche Lernaufgaben für welche lernenden Systeme mit welchem Aufwand prinzipiell zu bewältigen sind. Aus der algorithmisch-anwendungsorientierten Perspektive schließlich ist man daran interessiert, Verfahren zu entwickeln und Systeme zu konstruieren, die in praktisch relevanten Anwendungen Lernaufgaben lösen und einen Nutzen erbringen. Dabei sind diese Perspektiven keineswegs völlig disjunkt. Es gibt im Gegenteil viele Arbeiten im maschinellen Lernen, bei denen praktisch erfolgreiche Verfahren zunächst aus kognitiver Perspektive entwickelt worden sind und schließlich auch zu entsprechenden formalen Modellen führten.

Bei der praktischen Anwendung des maschinellen Lernens lassen sich (mindestens) zwei große Bereiche unterscheiden. Da ist zum einen die interaktive Analyse von vorher gesammelten Datenbeständen mit Hilfe von Lernverfahren. Dieser Anwendungstyp hat in den letzten Jahren eine sehr hohe Aufmerksamkeit erfahren, nicht zuletzt im Data Mining bzw. bei der Wissensentdeckung in Datenbanken (engl. knowledge discovery in databases), auf die wir im folgenden Abschnitt etwas genauer eingehen. Genau so wichtig ist aber die Verwendung von Lernverfahren zur Erzielung adaptiven Verhaltens, d.h., die Einbettung von Lernverfahren in ein System, das kontinuierlich neue Daten erhält und auf ihrer Basis auch kontinuierlich Voraussagen machen und Entscheidungen treffen muss. Die verschiedenen Abschnitte dieses Kapitels enthalten jeweils Anwendungsbeispiele aus diesen Bereichen für die verschiedenen Lernaufgaben.

12.1.4 Wissensentdeckung

über die Definitionen der Begriffe Data Mining bzw. Knowledge Discovery in Databases (KDD) herrscht derzeit weitgehend Einigkeit. Im wissenschaftlichen Bereich wird die folgende Definition des Begriffs KDD oft zitiert [29]:

Wissensentdeckung in Datenbanken ist der nichttriviale Prozess der Identifikation gültiger, neuer, potentiell nützlicher und schlussendlich verständlicher Muster in (großen) Datenbeständen.

Data Mining wird dabei als Bezeichnung für den eigentlichen Analyseschritt, in dem Hypothesen gesucht und bewertet werden, verwendet, d.h. Data Mining ist ein Teilschritt des KDD-Prozesses. Im kommerziellen Bereich wird diese Unterscheidung oft nicht getroffen, hier wird der Begriff Data Mining in der Regel als Synonym von bzw. anstatt des (vielleicht zu unhandlich langen) Begriffs „Knowledge Discovery in Databases“ verwendet. Im Deutschen wird in der Regel die Bezeichnung „Wissensentdeckung“ (WED) oder einfach ebenfalls „Data Mining“ verwendet [97].

Die Betonung des Prozessaspektes in der Definition verweist auf die umfassende Sichtweise, die KDD auf den Prozess der Datenanalyse hat: es werden alle Schritte von der ersten Beschäftigung mit einer Domäne bis hin zur Verwendung der Ergebnisse in Reports oder installierten Softwaresystemen betrachtet (Abbildung 12.1).

Was unterscheidet nun KDD vom Maschinellen Lernen? Zunächst einmal beinhaltet das Maschinelle Lernen große Bereiche, die bei KDD nicht betrachtet werden, insbesondere die Nutzung von Lernverfahren zur Erzielung von Adaptivität. Diese „Hälfte“ des Maschinellen Lernens ist bei KDD normalerweise nicht von Interesse (ebenso wie Modelle menschlicher Lernprozesse). Gemeinsam ist beiden Gebieten das Interesse an der Analyse bereits gesammelter Daten, und tatsächlich kommen im Analyseschritt zu grossen Teilen Techniken des Maschinellen Lernens zum Einsatz. Der Bereich der Wissensentdeckung hat andererseits das Maschinelle Lernen um einige interessante deskriptive Lernaufgaben bereichert, wie

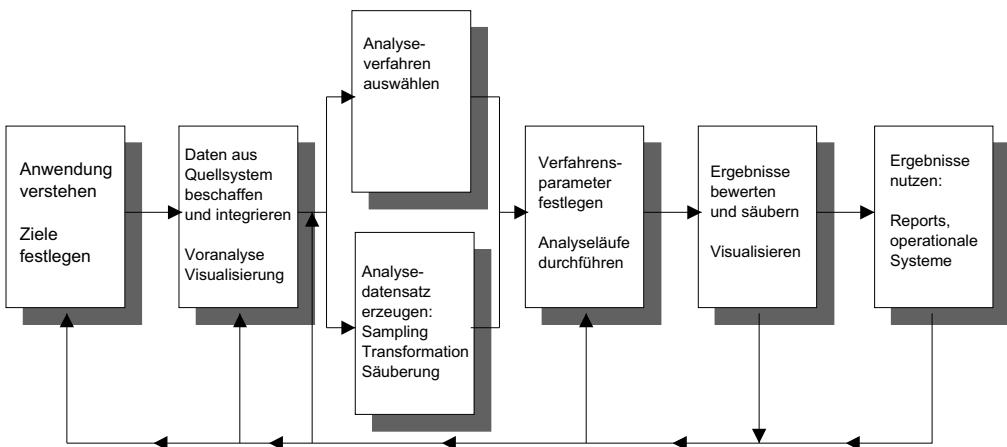


Abbildung 12.1: Der KDD-Prozess.

z.B. die Subgruppenentdeckung (Abschnitt 12.9) oder das Lernen von Assoziationsregeln (Abschnitt 12.8).

Ein weiterer Unterschied zwischen KDD und ML liegt in den Anforderungen an die Skalierbarkeit der Verfahren. Hier betont KDD sehr viel stärker als das „klassische“ ML die Notwendigkeit, mit „großen“ Datenmengen umzugehen. Dabei bezieht sich Größe sowohl auf die Anzahl der Variablen (Spalten der einzelnen Tabellen), wo einige Hundert bis einige Tausend angestrebt werden, als auch auf die Anzahl der Datenpunkte (Zeilen), wo die Ziele im Millionen- bis Milliardenbereich liegen. So ist denn auch der Datenbankbereich neben der Statistik eine der wichtigsten zu KDD beitragenden Nachbardisziplinen.

Auch wenn in der KDD der interaktive und iterative Prozess, bei dem Mensch und Lernverfahren gemeinsam verständliches und interessantes Wissen entdecken, sehr stark betont wird, so heißt das nicht, dass diese Sicht im Maschinellen Lernen nicht ebenso verbreitet war; KDD hat hier jedoch zu einer deutlichen Explizierung beigetragen.

12.2 Funktionslernen aus Beispielen

Das Funktionslernen aus Beispielen ist unbestritten die populärste, am häufigsten betrachtete und auch am häufigsten genutzte Lernaufgabe des Maschinellen Lernens. Betrachten wir eine Beispielanwendung aus dem Bereich der Pflanzenökologie [51]. Für das wissenschaftliche Verständnis der Pflanzenwelt, aber auch für praktische Zwecke, z.B. für die Auswahl von Pflanzen für die Wiederaufforstung von Flächen, ist es von großer Bedeutung, genau zu verstehen, unter welchen boden- und lokalklimatischen Bedingungen eine bestimmte Pflanzenart wachsen kann. Gesucht wird also ein Klassifikator, der für einen bestimmten Standort entscheiden kann, ob z.B. die Rotbuche an diesem Standort wachsen kann. Um solche Klassifikatoren manuell aufzustellen, bereisen Ökologen unterschiedliche Standorte, halten die Eigenschaften des Bodens und des Lokalklimas fest und stellen fest, ob eine bestimmte Pflanze an diesem Standort wachsen konnte oder nicht. So entstehen also für jede Pflanzenart *Beispiele* von Standorten, an denen diese Pflanze wachsen kann, und Beispiele von Standorten, an denen diese Pflanze nicht wachsen kann¹.

Tabelle 12.1 zeigt einige Beispiele solcher Standorte für die Rotbuche². Jede Zeile dieser Tabelle stellt dabei einen Standort dar. In der ersten Spalte finden wir einen numerischen Bezeichner für den Standort. Das Merkmal Bodenfeuchte kann die Werte trocken oder feucht annehmen, das Merkmal Bodensäure die Werte basisch, neutral oder sauer, und das Merkmal Durchschnittstemperatur die gemessene Durchschnittstemperatur auf eine Stelle gerundet. Die letzte Spalte der Tabelle gibt die Klassifikation des Standortes an. Mit W sind diejenigen Standorte gekennzeichnet, an denen die Rotbuche wachsen kann, mit N sind diejenigen Standorte gekennzeichnet, an denen sie nicht wachsen kann.

¹ Es wird geschätzt, dass allein in Deutschland schon einige hunderttausend dieser sogenannten „Standortansprüchen“ gesammelt worden sind. Tatsächlich kann durch eine Standortbegehung nur festgestellt werden, dass eine bestimmte Pflanze an einem bestimmten Standort wächst, da natürlich nicht alle Pflanzen, die potenziell dort wachsen könnten, tatsächlich auch anzutreffen sind. Zur Vereinfachung unseres Beispiels betrachten wir diese Problematik hier nicht.

² Das Beispiel ist fiktiv, stark vereinfacht und dient nur zur Illustration. Es sollte nicht als akkurate Beschreibung ökologischer Zusammenhänge gelesen werden.

Tabelle 12.1: Beispiele für Standorte einer Pflanzenart.

ID	Feuchte	Säure	Temp.	Klasse	ID	Feuchte	Säure	Temp.	Klasse
1	trocken	basisch	7	W	9	trocken	alkalisch	9	W
2	feucht	neutral	8	N	10	trocken	alkalisch	8	W
3	trocken	neutral	7	W	11	feucht	basisch	7	N
4	feucht	alkalisch	5	N	12	feucht	neutral	10	W
5	trocken	neutral	8	N	13	trocken	basisch	6	W
6	trocken	neutral	6	W	14	feucht	alkalisch	7	N
7	trocken	neutral	11	N	15	trocken	basisch	3	N
8	trocken	neutral	9	N	16	trocken	basisch	4	W

Sollen nun die Ökologen bei der aufwendigen Formulierung der Klassifikatoren unterstützt werden, so handelt es sich dabei um die Lernaufgabe „Funktionslernen aus Beispielen“. Wir gehen nämlich implizit davon aus, dass es eine uns und dem Lernverfahren nicht bekannte Klassifikationsfunktion f gibt, die für jeden Standort die „wahre“ Klassifikation ausgibt. Die Beispiele in unserer Tabelle stellen also eine *Stichprobe* des Verhaltens dieser Funktion f dar, die in unserem Fall durch die Gesetze der Biologie bestimmt wird. Den gesuchten Klassifikator können wir als eine Funktion h ansehen, welche idealerweise für jeden möglichen Standort genau das gleiche Ergebnis liefern sollte wie die uns unbekannte Funktion f . Ein möglicher Klassifikator h (in Regelform) könnte beispielsweise sein:

„Wenn Bodenfeuchte=trocken und Bodensäure=neutral,
dann Klasse=W, sonst Klasse=N.“

Dieser Klassifikator liefert allerdings bereits für die Beispiele nicht genau die gleichen Ergebnisse wie die unbekannte Funktion f , so dass wir ihn wohl nicht als wünschenswertes Lernergebnis ansehen würden.

Wir präzisieren daher die Lernaufgabe „Funktionslernen aus Beispielen“ wie folgt.

Definition 12.2.1 (Funktionslernen aus Beispielen). *Es sei X eine Menge möglicher Instanzenbeschreibungen, D eine Wahrscheinlichkeitsverteilung auf X , und Y eine Menge möglicher Zielwerte. Es sei weiterhin H eine Menge zulässiger Funktionen (auch als Hypothesensprache L_H bezeichnet). Eine Lernaufgabe vom Typ Funktionslernen aus Beispielen sieht dann wie folgt aus.*

Gegeben:

- Eine Stichprobe E von Beispielen der Form $(x, y) \in X \times Y$, für die gilt: $y = f(x)$ für eine unbekannte Funktion f .

Finde: Eine Funktion $h \in H$

- so dass der Fehler $\text{error}_D(h, f)$ von h im Vergleich zu f bei gemäß der Verteilung D gezogenen Instanzen aus X möglichst gering ist.

Beispiele der gleichen Form (x, y) können auch mehrfach in E vorkommen. Trotzdem spricht man von E meist als Menge, da jedes Beispiel einem unterschiedlichen Ereignis oder Objekt in

der Welt entspricht. Auch die Rolle der Verteilung D bedarf weiterer Erläuterungen. So wäre es doch viel einfacher gewesen zu fordern, dass der gelernte Klassifikator h möglichst viele der gegebenen Beispiele genauso klassifiziert wie die unbekannte Funktion f . Dieses Gütemaß für eine gelernte Funktion h bezeichnet man als *Trainingsfehler*, da die gegebenen Beispiele E oft auch als *Trainingsmenge* bezeichnet werden.

Definition 12.2.2 (Trainingsfehler). *Seien E , f und H wie in der vorhergehenden Definition. Als Trainingsfehler von h bezüglich f bezeichnet man*

$$\text{error}_E(h) := \frac{1}{|E|} \sum_{(x,y) \in E} \text{error}(h(x), y)$$

wobei man üblicherweise für numerische Zielwerte Y definiert:

$$\text{error}(h(x), y) := (h(x) - y)^2$$

und für diskrete Zielmengen Y mit nur wenigen Werten:

$$\text{error}(h(x), y) := 0, \text{ wenn } h(x) = y, \text{ sonst } 1$$

Ersteren bezeichnet man als den quadrierten Fehler, letzteren als 0-1-Fehler.

Der Trainingsfehler misst also, wie gut unsere gelernte Funktion h die bereits bekannten Objekte klassifiziert. Er ist daher einfach auszuwerten, doch tatsächlich ist man naheliegenderweise mehr daran interessiert, wie gut die gelernte Funktion bei zukünftigen, noch nicht bekannten Objekten mit der wahren, aber unbekannten Funktion f übereinstimmt. Um dies sinnvoll definieren zu können, müssen wir wenigstens annehmen, dass die uns zukünftig präsentierten, zu klassifizierenden Objekte aus den einzelnen Bereichen aus dem Instanzenraum X mit ähnlicher Häufigkeit gewählt werden wie dies bei den Beispielen E der Fall war. Gilt dies nicht, so kann man sich leicht vorstellen, dass unsere Lernaufgabe sehr schwer bzw. unlösbar wird. Wir nehmen genau deshalb an, dass es die oben eingeführte Wahrscheinlichkeitsverteilung D gibt, die die Auswahlwahrscheinlichkeit bestimmter Instanzen aus X sowohl für die Beispiele als auch für zukünftig zu klassifizierende Objekte angibt. Mit Hilfe von D können wir nun auch den eigentlich zu minimierenden wahren Fehler definieren.

Definition 12.2.3 (Wahrer Fehler). *Seien D , f und H wie in den vorangegangenen Definitionen. Als wahren Fehler der Hypothese h bezeichnet man:*

$$\text{error}_D(h, f) := \mathbf{E}_D [\text{error}(h(x), f(x))]$$

also den erwarteten (durchschnittlichen) Fehler, falls Instanzen aus X gemäß D gewählt werden.

Da ein Lernverfahren D und f selbst nicht kennt, sondern nur die mit D gewählten Beispiele E , ist der wahre Fehler nicht direkt zu bestimmen. Dennoch müssen Lernverfahren sicherstellen, dass sie nicht einfach den Trainingsfehler minimieren, denn dabei tritt das Phänomen der sogenannten *Überanpassung* (engl. *overfitting*) auf. Überanpassung bedeutet, dass ein Lernverfahren eine Hypothese auswählt, die zwar auf den Trainingsdaten einen minimalen Fehler hat,

jedoch auf späteren zu klassifizierenden Daten einen sehr viel höheren wahren Fehler aufweist als andere ebenfalls mögliche Hypothesen. Das Lernverfahren hat sich also den verfügbaren Trainingsbeispielen sehr genau angepaßt, dabei jedoch die zugrunde liegende wahre Funktion f nicht optimal erfaßt.

Dieses Phänomen tritt wohlgerne nicht nur dann auf, wenn die Beispieldaten *verrauscht* sind, also durch fehlerhafte Eingaben Beispiele enthalten, bei denen $y \neq f(x)$. Auch wenn die Beispieldaten fehlerfrei der gesuchten Funktion f entspricht, kann es bei genügend großem oder ungünstig gestaltetem Hypothesenraum L_H Hypothesen geben, die zufällig die gesehenen Beispiele richtig klassifizieren, von der gesuchten Funktion f aber ansonsten weit entfernt sind. Alle im folgenden vorgestellten Lernverfahren enthalten daher Mechanismen, um das Überanpassungsphänomen vermeiden zu helfen (siehe z. B. die Nutzung einer Validierungsmenge, Abschnitt 12.3.1, oder die Kreuzvalidierung, Abschnitt 12.4.2). Allgemein werden solche Ansätze als *Modellselektionsverfahren* bezeichnet (siehe z.B. [49]).

Definition 12.2.1 beschreibt nur einen Spezialfall des Funktionslernens aus Beispielen. Es wird die Annahme gemacht, dass es eine Funktion f gibt, welche für die gleiche Instanzenbeschreibung x immer den gleichen Zielwert $y = f(x)$ liefert. Der Zusammenhang ist also deterministisch. Realistischer ist es meist, den Zusammenhang durch eine bedingte Wahrscheinlichkeitsverteilung $P(Y|X)$ zu modellieren. Dann geht nicht nur $D = P(X)$, sondern auch $P(Y|X)$ in den wahren Fehler ein. Analog zu f und $P(X)$ in Definition 12.2.1, beschreibt im allgemeinen Fall $P(X, Y) = P(Y|X)P(X)$ das Lernziel vollständig, wobei die Trainingsmenge aus der Wahrscheinlichkeitsverteilung $P(X, Y)$ gezogen wird.

Abschließend sei noch erwähnt, dass sich für bestimmte Ausprägungen der Lernaufgabe Funktionslernen aus Beispielen besondere Namen eingebürgert haben. Besteht die Zielmenge Y aus einer Menge mit zwei diskreten Werten, spricht man auch von *Begriffslernen aus Beispielen*, bei dem alle Objekte mit einer bestimmten Ausprägung von Y als „Instanzen“ (Mitglieder) und alle Objekte mit der anderen Ausprägung als „Nicht-Instanzen“ (Nicht-Mitglieder) bezeichnet und die gesuchte Funktion als „Begriff“ bezeichnet werden. Bei unserer Beispieldatenwendung zum Pflanzenwuchs handelt es sich also um Begriffslernen aus Beispielen. Hat Y mehr als zwei, aber nur sehr wenige Werte, so spricht man auch vom Lernen von Klassifikationen. Ist Y dagegen numerisch, so bezeichnet man die Funktionslernaufgabe oft auch als *Regression*. Wie wir in Abschnitt 12.7 zur Induktiven Logikprogrammierung sehen werden, lassen sich auch Klassifikations- und Regressionsaufgaben auf binäre Begriffslernprobleme abbilden, indem man nicht eine Funktion lernt, die eine Instanz x auf einen Zielwert y abbildet, sondern eine Funktion, die das Paar (x, y) genau dann auf den Wert 1 (bzw. true) abbildet, wenn $y = f(x)$, und sonst auf 0 (bzw. false).

12.3 Entscheidungsbäume

Entscheidungsbäume [19, 67, 74] zählen seit langer Zeit zu den populärsten und am häufigsten benutzten Verfahren des maschinellen Lernens. Dies liegt vermutlich daran, dass sie einfach zu bedienen sind, nur relativ kurze Laufzeiten benötigen und dass die produzierten Entscheidungsbäume für die Benutzer relativ einfach zu verstehen sind. Insbesondere werden sie aber häufig als Komponenten von anderen Lernverfahren wie Boosting and Bagging (siehe Abschnitt 12.3.2) benutzt.

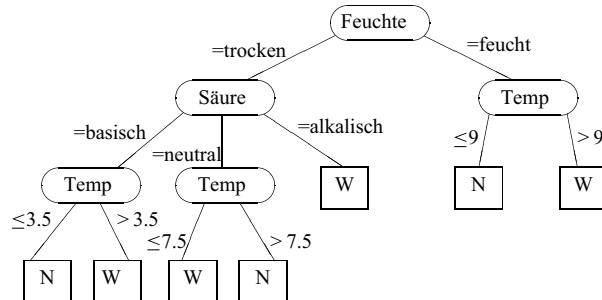


Abbildung 12.2: Entscheidungsbaum zur Standortbeurteilung.

Betrachten wir als Beispiel einen Entscheidungsbaum, mit dem man die im vergangenen Abschnitt vorgestellte Aufgabe der Klassifikation von Standorten hinsichtlich ihrer Eignung für die Rotbuche vornehmen könnte. Der in Abbildung 12.2 gezeigte Entscheidungsbaum ist von oben nach unten zu lesen. Zur Klassifikation eines neuen Standortes beginnt man an der Wurzel des Baumes und prüft zunächst den Wert des in diesem Knoten angegebenen Attributs. Man verfolgt dann den Pfad zum nächsten Knoten, der dem in der zu klassifizierenden Instanz vorgefundenen Wert des Attributs entspricht. Dadurch gelangt man zu einem neuen Knoten, wo das nächste zu prüfende Attribut wartet und so weiter. Schließlich gelangt man zu einem Blattknoten, in dem keine weitere Frage, sondern die auszugebende Klasse steht. Bei dem in der Abbildung gezeigten Entscheidungsbaum würde also der Standort $\langle \text{trocken}, \text{alkalisch}, 7 \rangle$ vom Wurzelknoten aus zunächst in den linken Ast wandern, um dann nach rechts zu verzweigen und somit als „W“ klassifiziert zu werden.

Dieses Beispiel zeigt bereits die Lernaufgabe, die von den meisten aktuellen Entscheidungsbaumverfahren gelöst wird.

Definition 12.3.1 (Entscheidungsbaumlernen). *Entscheidungsbaumverfahren lösen die Lernaufgabe Funktionslernen aus Beispielen, wobei X eine Menge von durch n numerische oder diskrete Attribute beschriebenen Instanzen ist, Y eine kleine Menge von diskreten Klassenwerten und L_H die Menge der aus diesen Attributen und ihren Werten konstruierbaren Entscheidungsbäumen ist³.*

Wie in unserem Beispiel werden als Bedingungen an den Knoten und Pfaden des Baumes in der Regel die Operatoren $=$, \leq und oft auch \in (aktueller Wert muß Element einer angegebenen Wertemenge sein) zugelassen.

Da in jedem Pfad eines Entscheidungsbaumes die Attribute in unterschiedlicher Reihenfolge vorkommen können (numerische Attribute sogar mehrmals in einem Pfad), ist die Anzahl der möglichen Entscheidungsbäume exponentiell in der Anzahl der Attribute. Es ist daher in der Regel nicht praktikabel, alle möglichen Entscheidungsbäume zu betrachten und zu vergleichen. Stattdessen wird der Baum von der Wurzel her (top-down, daher auch *top-down induction of decision trees*, TDIDT, genannt) aufgebaut, indem man sich – zunächst für den Wurzelknoten – überlegt, welches Attribut lokal in dem jeweiligen Knoten die größte Verbesserung der

³ Es gibt auch Entscheidungsbaumverfahren, die komplexere Instanzenräume X beziehungsweise auch numerische Wertemengen Y verarbeiten können. Wir konzentrieren uns hier auf die am häufigsten anzutreffende Variante der Lernaufgabe.

Tabelle 12.2: Beispiele nach verschiedenen Kriterien sortiert. Die ID entsprechen denen aus Tabelle 12.1. Die Spalten „K“ zeigen die Klasse des Beispiels und die Spalten „V“ die Vorhersage.

naiv			nach Feuchte			nach Säure			nach Temperatur			
ID	K	V	ID	K	V	ID	K	V	ID	K	V	
1	W	N		trocken				basisch			≤ 6.5	
2	N	N	1	W	W	1	W	W	15	N	W	
3	W	N	3	W	W	11	N	W	16	W	W	
4	N	N	5	N	W	13	W	W	4	N	W	
5	N	N	6	W	W	15	N	W	6	W	W	
6	W	N	7	N	W	16	W	W	13	W	W	
7	N	N	8	N	W		neutral			> 6.5		
8	N	N	9	W	W	2	N	N	11	N	N	
9	W	N	10	W	W	3	W	N	3	W	N	
10	W	N	13	W	W	5	N	N	14	N	N	
11	N	N	15	N	W	6	W	N	1	W	N	
12	W	N	16	W	W	7	N	N	5	N	N	
13	W	N		feucht			8	N	N	2	N	N
14	N	N	2	N	N	12	W	N	10	W	N	
15	N	N	4	N	N		alkalisch			8	N	N
16	W	N	11	N	N	4	N	N	9	W	N	
			12	W	N	9	W	N	12	W	N	
			14	N	N	10	W	N	7	N	N	
						14	N	N				

Klassifikationsleistung erbringen würde. Wir können das Prinzip am einfachsten anhand unserer Beispielanwendung illustrieren.

Tabelle 12.1 aus dem vorangegangenen Abschnitt zeigt die Beispiele, aus denen wir nun den Entscheidungsbaum bauen wollen. Da wir noch gar keine Verzweigung vorgenommen haben, ist es am günstigsten, für alle Beispiele die Mehrheitsklasse vorherzusagen, in unserem Fall (beide Klassen sind gleich häufig) also z.B. „N“. Damit erreichen wir einen Trainingsfehler von $\frac{8}{16}$. Diese naive Vorhersage zeigt der linke Kasten in Tabelle 12.2. Teilen wir nun probeweise unsere Standorte anhand des Attributes Bodenfeuchte gemäß der auftretenden Werte in zwei Gruppen auf, und sagen dann für diese beiden Gruppen jeweils deren Mehrheitsklasse voraus, so ergibt sich das in dem zweiten Kasten von Tabelle 12.2 dargestellte Bild.

In der Gruppe der Standorte, bei denen Bodenfeuchte gleich trocken war (11 Standorte) erreichen wir so eine Fehlerrate von $\frac{4}{11}$. In der anderen Gruppe (5 Standorte) erreichen wir eine Fehlerrate von $\frac{1}{5}$. Insgesamt ergibt dies also jetzt eine Trainingsfehlerrate von $\frac{11}{16} \cdot \frac{4}{11} + \frac{5}{16} \cdot \frac{1}{5} = \frac{5}{16}$. Wir können also feststellen, dass wir mit der Aufteilung der Beispieldatenmenge anhand des Attributs Bodenfeuchte unseren Trainingsfehler verbessern können. Möglicherweise ist jedoch die Verbesserung durch eines der anderen Attribute noch stärker. Wir prüfen dies zunächst für das Attribut Bodensäure und erhalten die Situation in dem dritten Kasten von Tabelle 12.2.

Wie man sieht, können wir nach Aufteilung mit diesem Attribut in den jeweiligen Gruppen 2 von 5, 3 von 7 und 2 von 4 Standorten richtig klassifizieren, erreichen also insgesamt einen Trainingsfehler von $\frac{5}{16} \cdot \frac{2}{5} + \frac{7}{16} \cdot \frac{3}{7} + \frac{4}{16} \cdot \frac{2}{4} = \frac{7}{16}$. Mit diesem Attribut ist also eine weniger große

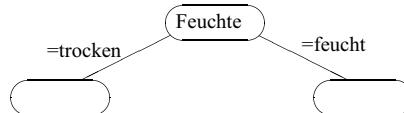


Abbildung 12.3: Entscheidungsbaum nach Bestimmung des Wurzelattributs.

Verbesserung des Trainingsfehlers möglich. Das dritte Attribut Durchschnittstemperatur ist numerisch, so dass es nicht sinnvoll ist, anhand der einzelnen Werte aufzuteilen. Wir teilen die Standorte stattdessen anhand eines Schwellwertes in zwei Gruppen auf. Da wir einen Schwellwert jeweils in die Mitte zwischen zwei tatsächlich aufgetretene Werten legen können, ergeben sich daraus in unserem Beispiel (9 verschiedene Werte) 8 mögliche Aufteilungen in 2 Gruppen.

Der rechte Kasten in Tabelle 12.2 zeigt die Aufteilung für den Schwellwert 6,5, welche von allen 8 möglichen Schwellwerten den günstigsten Trainingsfehler erreicht, nämlich $\frac{5}{16} \frac{2}{5} + \frac{11}{16} \frac{5}{11} = \frac{7}{16}$. Da auch dieser Wert schlechter ist als der für das Attribut Bodenfeuchte, entscheiden wir uns, unseren Entscheidungsbaum mit dem Attribut Bodenfeuchte zu beginnen. Der Entscheidungsbaum sieht also jetzt aus wie in Abbildung 12.3.

Durch das Einführen des ersten Knotens teilen sich, wie in Tabelle 12.2 schon gezeigt, unsere Beispiele nun auf die beiden Zweige des Baumes auf. Zum weiteren Aufbau des Baumes können wir daher nun die gleiche Prozedur rekursiv jeweils mit den nun kleineren Teilgruppen in den beiden neu entstandenen, noch leeren Knoten durchführen. Dies tun wir solange, bis wir in einem Knoten und seiner Teilgruppe einen Trainingsfehler von 0 erreicht haben (alle Instanzen sind von einer Klasse), bzw. bis uns die möglichen Vergleichsbedingungen für die Knoten ausgehen (denn schließlich ist es nicht sinnvoll, denselben Test mehrmals in einem Pfad zu wiederholen). In unserem Beispiel ergibt sich dabei die in Tabelle 12.3 dargestellte Gruppenaufteilung und der in Abbildung 12.2 dargestellte Entscheidungsbaum. Abbildung 12.4 enthält den allgemeinen Algorithmus zur Entscheidungbaumkonstruktion, in dem das Qualitätsmaß „Qua-

Tabelle 12.3: Endaufteilung mit Trainingsfehler 0.

ID	Feuchte	Säure	Temp.	Klasse	Vorhersage
15	trocken	basisch	3	N	N
16	trocken	basisch	4	W	W
13	trocken	basisch	6	W	W
1	trocken	basisch	7	W	W
6	trocken	neutral	6	W	W
3	trocken	neutral	7	W	W
5	trocken	neutral	8	N	N
8	trocken	neutral	9	N	N
7	trocken	neutral	11	N	N
9	trocken	alkalisch	9	W	W
10	trocken	alkalisch	8	W	W
4	feucht	alkalisch	5	N	N
11	feucht	basisch	7	N	N
14	feucht	alkalisch	7	N	N
2	feucht	neutral	8	N	N
12	feucht	neutral	10	W	W

Sei E die Beispieldmenge und $tests$ die Menge aller aus den Attributen und Attributwerten der Beispieldmenge konstruierbaren Knotentests.

$TDIDT(E, Tests)$

- Falls E nur Beispiele einer Klasse enthält, liefere einen Blattknoten mit dieser Klasse zurück. Andernfalls:
- Für jeden Test in $Tests$, berechne $Qualität(Test, E)$
- Wähle den Test mit der höchsten Qualität für den aktuellen Knoten aus.
- Teile E anhand dieses Tests in 2 oder mehr Teilmengen E_1, \dots, E_k auf.
- Für $i = 1, \dots, k$ rufe rekursiv: $T_i := TDIDT(E_i, Tests \setminus \{test\})$
- Liefere als Resultat den aktuellen Knoten mit den darunter hängenden Teilbäumen T_1, \dots, T_k zurück.

Abbildung 12.4: Grundalgorithmus für Entscheidungsbaumverfahren.

lität“ eine wichtige Rolle spielt. Wir haben in unserem Beispiel einfach die durchschnittliche Verringerung des Trainingsfehlers genommen. Tatsächlich hat sich herausgestellt, dass einige andere Bewertungsfunktionen für diesen Zweck geeigneter sind. Eine davon, der sogenannte *Informationsgewinn* ([74], engl. *information gain*) soll hier nun kurz vorgestellt werden. In der Kodierungstheorie [84] bezeichnet man als Informationsgehalt einer bestimmten Menge von Symbolen die minimale Anzahl von Fragen, die man stellen muß, um ohne vorheriges Wissen ein Symbol dieser Menge eindeutig identifizieren zu können. Da die Symbole unterschiedlich wahrscheinlich sind, ist genauer gesagt die im Mittel zu erwartende Anzahl von Fragen gemeint. Falls die m Symbole einer Menge mit Wahrscheinlichkeiten p_1, \dots, p_m auftreten, so lässt sich der auch *Entropie* genannte Informationsgehalt, also die zu erwartende und in Bit gemessene Anzahl der notwendigen Fragen wie folgt berechnen:

$$I(p_1, \dots, p_m) := \sum_{i=1}^m -p_i \log p_i$$

Beim Bau eines Entscheidungsbaumes kennen wir die Wahrscheinlichkeiten der einzelnen Klassen nicht, können aber näherungsweise die relativen Häufigkeiten der Klassen im aktuellen Knoten verwenden. Statt desjenigen Tests, der den durchschnittlichen zu erwartenden Trainingsfehler am stärksten reduziert, verwenden wir beim Informationsgewinn-Kriterium denjenigen Test, der den durchschnittlichen Informationsgehalt der neu entstehenden Teilmengen am stärksten reduziert. Da der Informationsgehalt vor der Aufteilung für alle möglichen Tests gleich ist, brauchen wir diese Differenz nicht tatsächlich zu berechnen, sondern können einfach das Qualitätsmaß wie folgt definieren:

Definition 12.3.2 (Qualitätsmaß). Sei $Test$ ein Test, der die Beispieldmenge E in Teilmengen E_1, \dots, E_k zerlegt. Es seien $p_{i,1}, \dots, p_{i,m}$ die relativen Häufigkeiten der m Klassen in der Teilmenge E_i . Dann definiere:

$$Qualität(Test, E) := IG(Test, E) := - \sum_{i=1}^k \frac{|E_i|}{|E|} I(p_{i,1}, \dots, p_{i,m})$$

12.3.1 Stutzen des Baumes

Auch durch die Verwendung des Informationsgewinns als Qualitätsmaß ändert sich nichts daran, dass die Baumkonstruktion nur am Trainingsfehler orientiert ist, und erst endet, wenn der Trainingsfehler Null geworden ist. Hier besteht also ein großes Risiko der Überanpassung der Trainingsdaten. Bei Entscheidungsbaumverfahren wird deshalb üblicherweise der aufgebaute Baum in einem separaten Schritt *gestutzt*.

Abbildung 12.5 zeigt den Baum, der durch das Stutzen der Blätter ganz unten links entsteht. Da nun ein vorheriger innerer Knoten zum Blattknoten wird, finden sich in diesem Knoten Instanzen unterschiedlicher Klassen wieder, das heißt, der gestutzte Baum macht jetzt, wie sich Tabelle 12.3 entnehmen lässt, auf der Trainingsmenge einen Fehler (Beispiel 15 wird jetzt fälschlicherweise als „W“ klassifiziert).

Warum ist diese Prozedur sinnvoll, und wieweit soll der Baum zurückgestutzt werden? Man kann zum einen argumentieren, dass Bäume geringerer Tiefe den Instanzenraum weniger fein unterteilen können, und dass bei diesen Bäumen daher das Risiko der Überanpassung geringer ist. Kleinere Bäume sind auch aus Gründen der Verständlichkeit vorzuziehen. Andererseits ist klar, dass spätestens dann, wenn nur noch der Wurzelknoten übrig bleibt, der Baum vermutlich nicht nur einen hohen Trainings-, sondern auch einen hohen wahren Fehler haben wird.

Man braucht also eine Methode, um zu schätzen, wie sich der wahre Fehler des Baumes beim Zurückstutzen entwickelt. Dann kann man den Baum soweit zurückstutzen, wie der wahre Fehler sinkt. Eine allgemeine Methode, den wahren Fehler zu schätzen, besteht darin, einen bestimmten Anteil der verfügbaren Trainingsbeispiele nicht zum Training zu verwenden, sondern ihn als *Validierungsmenge* zunächst beiseite zu legen. Da der Entscheidungsbaum unabhängig von dieser Validierungsmenge erzeugt wird, kann der gemessene Fehler auf der Validierungsmenge als Schätzer für den wahren Fehler verwendet werden.

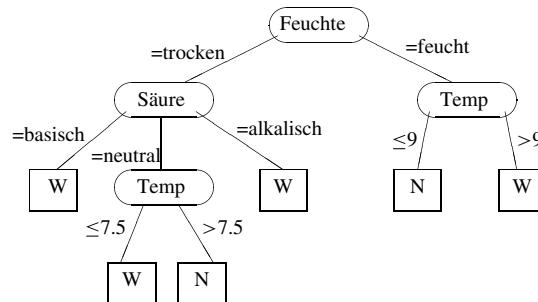


Abbildung 12.5: Gestuzzter Entscheidungsbaum.

Sind nur wenige Trainingsbeispiele vorhanden, würde das Beiseitelegen einer Validierungsmenge möglicherweise nicht mehr genügend Daten übrig lassen, um sinnvoll den ungestutzten Baum zu erzeugen. In diesen Fällen, und dann, wenn eine separate Validierungsmenge zu aufwendig erscheint, kann man sich behelfen, indem man eine heuristische Schätzfunktion verwendet [74]. Diese Funktion kann beispielsweise die erzielte Vereinfachung des Baumes (z.B. Anzahl eingesparter Knoten) verrechnen mit dem durch diese Vereinfachung gestiegenen Trainingsfehler. Diese heuristische Steuerung des Stutzens kann recht gut funktionieren, generell ist jedoch eine separate Validierungsmenge vorzuziehen [63].

12.3.2 Boosting and Bagging: Ensemble-Methoden

Entscheidungsbäume wie oben beschrieben zeichnen sich durch ein sehr leicht verständliches Lernergebnis und eine auch im Vergleich mit komplexeren und neueren Verfahren akzeptable Klassifikationsgüte aus und werden daher auch heute noch gerne verwendet. Gerade angesichts der für Nutzer attraktiven, weil anschaulichen Bäume fällt jedoch eine andere Eigenschaft manchmal negativ ins Gewicht: Entscheidungsbäume sind zwar bezüglich ihrer Gesamt-Genauigkeit, aber nicht bezüglich der Struktur der produzierten Bäume stabil. Relativ kleine Veränderungen der Trainingsmenge können dazu führen, dass sehr unterschiedliche Entscheidungsbäume gelernt werden. Die Ursache hierfür liegt im „top-down“ Vorgehen des Algorithmus, bei dem einmal eingefügte Knoten nie wieder revidiert werden können.

Geht es nicht um das Erzeugen eines einzelnen Baumes, sondern lediglich um das Erreichen optimaler Vorhersagen, so kann die Instabilität sogar zum Vorteil genutzt werden, in dem absichtlich aus unterschiedlich modifizierten Beispilmengen mehrere leicht unterschiedliche Bäume erzeugt und dann zur Vorhersage kombiniert werden. Tatsächlich konnte nicht nur für Entscheidungsbäume, sondern allgemein auch für andere Lernverfahren gezeigt werden, dass solche *Ensemble-Methoden* [27] nicht nur praktisch, sondern in vielen Fällen auch theoretisch begründet die Genauigkeit gegenüber dem Basis-Lernverfahren steigern können. Wir führen im folgenden zwei der populärsten Methoden, *bagging* und *boosting*, am Beispiel der Entscheidungsbäume ein.

Für Entscheidungsbäume existieren mehrere Methoden, um deren Mangel an Stabilität zu beheben bzw. sogar auszunutzen [16, 17, 18], aber die mit Abstand populärste Methode ist *Bagging* [16]. Anstatt den Entscheidungsbäumlernalgorithmus selbst zu verändern, ist Bagging eine Methode, die als äußere Schleife agiert und somit auch für andere Lernverfahren angewandt werden kann. Der Bagging-Algorithmus ist in Abbildung 12.6 zusammengefasst. Anstatt nur einen Entscheidungsbäum zu lernen, generiert der Bagging-Algorithmus eine große Anzahl von Entscheidungsbäumen, deren Vorhersagen dann per Mehrheitsentscheidung kombiniert werden. Jeder einzelne Entscheidungsbäum wird auf einer sogenannten *Bootstrap-Stichprobe* der Trainingsmenge gelernt. Diese Stichproben werden erzeugt, indem man n Beispiele aus der ursprünglichen Trainingsmenge E mit Zurücklegen zieht. Dies bedeutet, dass jede Bootstrap-Stichprobe die gleiche Anzahl von Beispielen enthält wie E , ein bestimmtes Trainingsbeispiel aber mehrfach oder gar nicht enthalten sein kann. Durch die Kombination vieler Entscheidungs-

Sei $E = ((x_1, y_1), \dots, (x_n, y_n))$ die Trainingsmenge von n Beispielen, und L ein beliebiges Lernverfahren, das aus Beispielen eine Hypothese erzeugt.

Bagging(E,L,I)

- Für $i = 1, \dots, I$
 - Setze E' leer.
 - Für $i = 1, \dots, n$ wähle zufällig und gleichverteilt ein Beispiel aus E aus und füge es zu E' hinzu.
 - Trainiere Lernalgorithmus L auf E' und speichere gelernten Baum als h_i .
- Liefere als Resultat die Regel $h(x) = \text{maj vote}\{h_1(x), \dots, h_I(x)\}$, welche die Mehrheitsentscheidung der h_i berechnet.

Abbildung 12.6: Bagging Algorithmus für ein Basis-Lernverfahren L .

Sei $E = ((x_1, y_1), \dots, (x_n, y_n))$ die Trainingsmenge von n Beispielen. Gegeben ist ein Lernalgorithmus L welcher Regeln der Form $h : X \rightarrow \{-1, +1\}$ lernt.

Adaboost(E,L,I)

- Setze $D_1(j) = 1/n$ für alle $j \in \{1..n\}$
- Für $i = 1, \dots, I$
 - Trainiere Lernalgorithmus L auf Beispielen aus E , gewichtet mit D_i , und speichere gelernten Baum als h_i .
 - Berechne den gewichteten Trainingsfehler $\epsilon_i = \sum_{j=1}^n D_i(j) \text{error}(h_i(x_j), y_j)$
 - Setze $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$
 - Setze $D_{i+1}(j) = D_i(j) e^{-\alpha_i y_j h_i(x_j)} / Z_i$ für alle $j \in \{1..n\}$, wobei $Z_i = \sum_{j=1}^n D_i(j) e^{-\alpha_i y_j h_i(x_j)}$
- Liefere als Resultat die Regel $h(x) = \text{sign} \left\{ \sum_{i=1}^I \alpha_i h_i(x) \right\}$, welche die gewichtete Mehrheitsentscheidung der h_i berechnet.

Abbildung 12.7: Adaboost Algorithmus für ein Basis-Lernverfahren L .

bäume wird die von Bagging gelernte Klassifikationsregel stabiler und hat typischerweise einen geringeren Vorhersagefehler.

Ebenfalls eine Kombination von Entscheidungsbäumen lernen auch *Boosting*-Algorithmen wie Adaboost [30]. Adaboost führt allerdings an zwei Stellen Gewichtungen ein, nämlich eine adaptive Gewichtung der Trainingsbeispiele und eine Gewichtung der gelernten Entscheidungsbäume bei der Mehrheitsentscheidung. Die Idee hinter Boosting geht auf Arbeiten in der Lerntheorie [47, 77] zurück. Dort wurde die Frage untersucht, ob man aus einem sogenannten „schwachen“ Lernalgorithmus, der nur etwas besser als zufällig vorhersagt, durch geeignete wiederholte Nutzung einen insgesamt „starken“ Lernalgorithmus, der beliebig genau ist, konstruieren kann.

Der Adaboost Algorithmus ist in Abbildung 12.7 zusammengefasst. In jeder Iteration i wird eine neue Regel gelernt, die den durch $D_i(j)$ gewichteten Fehler minimiert. Entweder benutzt man hierfür einen Lernalgorithmus, der direkt mit gewichteten Beispielen arbeiten kann, oder man zieht eine neue Trainingsmenge E' aus E anhand der Wahrscheinlichkeitsverteilung D_i . D_i wird in jeder Iteration angepasst. Dabei bekommen die Trainingsbeispiele ein hohes Gewicht, die in den vergangenen Iterationen häufig falsch klassifiziert wurden (d.h. bei denen das Produkt $y_j h_i(x_j)$ negativ war). Adaboost versucht somit mehr und mehr, auch diese besonders „schwierigen“ Trainingsbeispiele richtig zu klassifizieren. Obwohl es hierbei im Prinzip zur Überanpassung kommen kann, ist dies in der Praxis meist wenig problematisch. Eine weiterführende Einleitung zum Boosting findet sich in [31].

12.3.3 Erweiterungen des Basis-Entscheidungsbaumverfahrens

In diesem Lehrbuchkapitel kann auf die Vielfalt mittlerweile entwickelter Entscheidungsbaumverfahren natürlich nicht eingegangen werden. Dennoch sollen kurz die wichtigsten Varianten skizziert werden.

Splitting-Kriterien. Neben dem in diesem Kapitel vorgestellten Informationsgewinn-Kriterium sind viele andere Kriterien entwickelt worden. Einen Überblick gibt z.B. [67].

Regelerzeugung. Jeder Pfad von der Wurzel eines Entscheidungsbaumes bis zu einem Blattknoten entspricht einer logischen Regel. Vereinfacht man die einem Entscheidungsbaum entsprechende Regelmenge in geeigneter Weise, so entsteht ein leistungsfähiges Regel-lernverfahren [74].

Regression. Schon relativ früh wurde erkannt, dass man in den Blattknoten eines Entscheidungsbaumes statt einer einfachen Klassenzuweisung auch eine lineare Regressionsfunktion angeben kann, um so auch numerische Werte vorhersagen zu können [19]. Mit dieser Funktion wird dann lokal im jeweiligen Blattknoten die Vorhersage für eine dorthin klassifizierte Instanz berechnet. Auch die Verwendung von linearen Trennebenen als Knotentests im Inneren des Baumes ist bereits untersucht worden [68].

Fehlende Werte. Fehlende Werte in den Beispielen oder zu klassifizierenden Instanzen sollten nicht durch die Einführung eines speziellen Sonderwertes wie „unknown“ behandelt werden. Stattdessen ist es üblich, bei der Konstruktion des Baumes fehlende Werte gemäß der sonstigen Verteilung des betroffenen Attributes zu ergänzen. Bei der Klassifikation von Instanzen mit fehlenden Werten kann man entsprechend in probabilistischer Weise alle möglichen Pfade durch den Baum verfolgen und die Vorhersagen aller so erreichten Blattknoten gemäß ihrer Wahrscheinlichkeiten gewichtet miteinander kombinieren.

Skalierbarkeit. Bei der Konstruktion eines Entscheidungsbaumes müssen die n Beispiele grob gesprochen auf ungefähr $\log n$ Ebenen betrachtet werden, so dass der Aufwand mindestens in der Größenordnung von $n \log n$ liegt. Damit gehören Entscheidungsbaumverfahren zwar zu den schnellsten Lernverfahren, dennoch muß man sich für größere Datenbestände auch hier Gedanken über die Skalierbarkeit machen. Standard ist hier die Verwendung eines *Lernfensters* ([74], *windowing*), in das anfänglich nur ein Bruchteil der Daten aufgenommen wird, und das dann wiederholt durch einen Bruchteil der vom jeweils erzeugten Entscheidungsbaum nicht korrekt klassifizierten anderen Trainingsdaten ergänzt wird, bis der Gesamtfehler sich nicht weiter verbessert. Auch existieren bereits Verfahrensvarianten, die durch geschickte Datenstrukturen in der Lage sind, den Aufwand für die wiederholte Sortierung der Daten in jedem Knoten zu vermeiden [82]. Diese Verfahren sind nicht darauf angewiesen, alle Daten im Hauptspeicher zu halten und können darüber hinaus leicht parallelisiert werden.

12.4 Instanzbasiertes Lernen

Unter der Bezeichnung „Instanzbasierte Lernverfahren“ [5] sind im Maschinellen Lernen Verfahren wiederaufgenommen und weiterentwickelt worden, die in der Statistik schon lange als „Nächste Nachbarn“-Verfahren [25] populär waren und sind. Instanzbasierte Verfahren beruhen auf einer bestechend einfachen Idee. Um die Klasse eines neuen, unbekannten Objektes vorherzusagen, vergleicht man das zu klassifizierende Objekt mit den gegebenen Beispielen, findet das Beispiel, das dem zu klassifizierenden Objekt am *ähnlichsten* ist, und sagt dann den für dieses Beispiel gespeicherten Funktionswert voraus.

Auch instanzbasierte Lernverfahren lösen also die Lernaufgabe „Funktionslernen aus Beispielen“, aber auf ganz andere Art als die im vergangenen Abschnitt vorgestellten Entscheidungsbaumverfahren. Während Entscheidungsbaumverfahren die übergebene Beispieldatenmenge E analysieren und auf ihrer Basis eine kompakte Hypothese in Form eines Entscheidungsbaumes produzieren, besteht die Lernphase bei instanzbasierten Lernverfahren einfach nur aus dem Abspeichern der Beispiele. Arbeit investieren diese Verfahren erst dann, wenn tatsächlich der Funktionswert für ein neues unbekanntes Objekt vorhergesagt werden soll. Man bezeichnet die instanzbasierten Lernverfahren deswegen oft auch als *lazy learners*.

Im Gegensatz zu Entscheidungsbäumen, die zur Klassifikation sehr schnell durchlaufen werden können, ist die Klassifikation bei instanzbasierten Verfahren aufwendiger, da das neue Objekt mit allen bisherigen Objekten verglichen werden muss. Die große Popularität von instanzbasierten Lernverfahren erklärt sich zum einen dadurch, dass es für viele Nutzer natürlicher und überzeugender ist, als Begründung für eine getroffene Vorhersage nicht eine abstrakte Hypothese wie einen Entscheidungsbaum zu erhalten, sondern einfach Verweise auf ähnliche Instanzen aus den bekannten Beispielen E . Zum zweiten sind instanzbasierte Lernverfahren, trotz (bzw. gerade wegen) ihres einfachen Prinzips hervorragende Funktionsapproximatoren, deren Genauigkeit oft über der Genauigkeit von Entscheidungsbäumen oder anderen Lernverfahren liegt. Wie wir unten sehen werden, lassen sich darüber hinaus mit instanzbasierten Lernverfahren ebenso einfach diskrete Klassifikationen wie numerische Vorhersagen lernen.

Zur Präzisierung der von instanzbasierten Lernverfahren gelösten Lernaufgabe müssen wir zunächst überlegen, was eigentlich der Hypothesenraum eines solchen Verfahrens ist. Zur Erinnerung: Die Hypothese h ist eine Funktion, die eine Instanz $x \in X$ auf einen vorhergesagten Funktionswert $h(x)$ abbildet. Bei instanzbasierten Lernverfahren geschieht dies durch einen geeigneten Vergleichsalgorithmus auf der Basis der gegebenen Beispiele E . Da die Klassifikationsfunktion, wie unten beschrieben, meistens noch durch weitere Parameter gesteuert wird, wollen wir sie als Funktion

$$\kappa_{S,P} : X \rightarrow Y$$

wobei S eine Teilmenge aus X ist, im Regelfall also einfach die Menge der gegebenen Beispiele, und $P \in \mathcal{P}$ eine zulässige Parametrisierung der instanzbasierten Klassifikationsfunktion ist. Eine typische Parametrisierung wäre beispielsweise die Zahl der aus S zum Vergleich heranzuziehenden Nachbarn (siehe unten). Wir können dann die Lernaufgabe für instanzbasierte Lernverfahren wie folgt definieren.

Definition 12.4.1 (Instanzbasiertes Lernen). *Instanzbasierte Lernverfahren lösen die Lernaufgabe Funktionslernen aus Beispielen, wobei X eine Menge von durch n numerische oder diskrete Attribute beschriebenen Instanzen ist, Y eine numerische oder diskrete Menge von Zielwerten ist, und E mit Beispielen $(x, y) \in X \times Y$ gegeben ist. Wenn κ die gegebene instanzbasierte Klassifikationsfunktion mit zulässigen Parametrisierungen \mathcal{P} ist, so ist der Hypothesenraum $L_H := \{\kappa_{S,P} \mid S \subseteq E, P \in \mathcal{P}\}$, ist also abhängig von E .*

Diese Definition der Lernaufgabe erlaubt es also einem instanzbasierten Lernverfahren, etwas weniger „faul“ zu sein. Das Verfahren kann sich entscheiden, nur eine Teilmenge S der gegebenen Beispiele E für die Klassifikation aufzuheben und es kann während der Lernphase bestimmte Parameterwerte P bestimmen, die später der Klassifikationsprozedur übergeben werden.

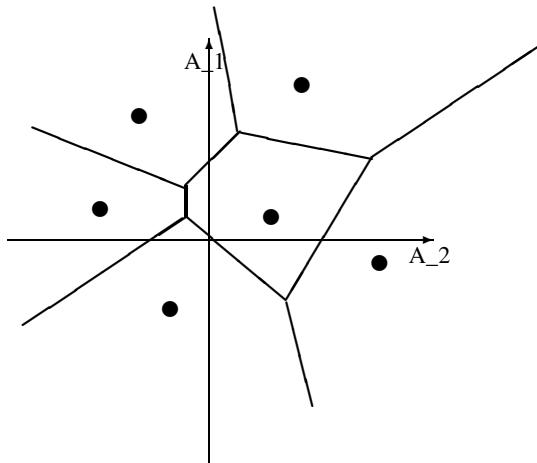


Abbildung 12.8: Voronoi-Diagramm für Nachbarschaftsbeziehungen.

Wie sieht nun die einfachste mögliche Klassifikationsfunktion aus? Nehmen wir zunächst an, die Ähnlichkeit zweier Instanzen $x_1 \in X, x_2 \in X$ sei durch eine Funktion

$$\text{sim} : X \times X \rightarrow [0, 1]$$

definiert, so dass höhere Werte von sim ähnlicheren Paaren von Instanzen entsprechen. Nehmen wir weiter an, dass die Funktion keine weiteren Parameter hat. Dann lässt sich die einfachste instanzbasierte Klassifikationsfunktion sehr kurz definieren (für $S \subseteq X$):

$$\kappa_{S,\emptyset}(x_{\text{new}}) := y_m, \text{ wobei } (x_m, y_m) = \underset{(x,y) \in S}{\text{argmax}} \text{sim}(x, x_{\text{new}})$$

Es wird also der gespeicherte Funktionswert y desjenigen Beispiels (x, y) verwendet, das der zu klassifizierenden Instanz x_{new} am ähnlichsten ist⁴.

Verwenden wir nun diese einfachste, nur auf einem Nachbarn basierende Klassifikation und nehmen immer alle vorhandenen Beispiele als Grundlage, so besteht für jede Beispieldmenge E der Hypothesenraum effektiv aus einem einzigen Element:

$$L_H := \{\kappa_{E,\emptyset}\}$$

Dieses einfachste aller instanzbasierten Lernverfahren wird üblicherweise als *Nächste-Nachbarn*-Verfahren (NN, *nearest neighbor*) bezeichnet. Abbildung 12.8 visualisiert für Instanzen mit zwei reellwertigen Attributen grafisch die beim NN-Verfahren entstehende kleinteilige Parzellierung des Instanzenraums („Voronoi-Diagramm“). Alle in eine Zelle dieses Diagramms fallenden Instanzen haben das eingezeichnete Beispiel als nächsten Nachbarn.

Diese kleinteilige Parzellierung ist für eine hohe Klassifikationsgüte zunächst einmal von Vorteil, da effektiv für jede Zelle des Voronoi-Diagramms die zum Beispiel passende Klassifikation gewählt wird (bei Verwendung nur eines Nachbarn). Genau diese Möglichkeit zur lokalen Anpassung birgt aber auch ein großes Risiko der Überanpassung. Einzelne verrauschte oder

⁴ Bei Gleichstand wollen wir hier und im folgenden immer annehmen, dass zufällig ausgewählt wird.

sonstwie außergewöhnliche Datenpunkte beeinflussen direkt die Klassifikation, auch wenn sie keinen generelleren Zusammenhang wiedergeben.

Aus diesem Grund wird üblicherweise die k -NN (k -Nächste-Nachbarn) genannte Variante des Verfahrens verwendet, bei der die Vorhersage nicht aus einem nächsten Nachbarn, sondern aus den k nächsten Nachbarn gewonnen wird. Hier ist also k der Parameter für die Klassifikationsfunktion κ , der vom Verfahren bestimmt (oder vom Nutzer vorgegeben) werden muss.

Verwendet man k Nachbarn, so erfolgt bei diskreten Vorhersageproblemen die Vorhersage naheilgenderweise durch eine Mehrheitsentscheidung innerhalb der Gruppe der k nächsten Nachbarn. Der Beitrag eines bestimmten Nachbarn zur Entscheidung wird dabei üblicherweise so gewichtet, dass er mit zunehmender Entfernung immer geringer wird.

Definition 12.4.2 (k -NN, diskret). *Gegeben eine Beispieldatenmenge E , eine zu klassifizierende Instanz x_{new} , eine natürliche Zahl k und eine Ähnlichkeitsfunktion sim , bezeichne mit $N_k \subseteq E$ die k nächsten Nachbarn von x_{new} , und mit $N_{k/y_j} := \{e = (x, y_j) \in N_k\}$ die Nachbarn, die für den Wert y_j stimmen. Die k -NN Vorhersage ist dann definiert als:*

$$\kappa_{E,k} := \operatorname{argmax}_{y_j} \sum_{e=(x,y_j) \in N_{k/y_j}} sim(x, x_{new})^a,$$

wobei $a \geq 0$ ein üblicherweise vom Benutzer bestimmter Parameter des Verfahrens ist ($a = 0$ entspricht einer ungewichteten Kombination, höhere Werte werten weiter entfernte Nachbarn immer stärker ab).

Bei kontinuierlichen Vorhersageproblemen ist es weder notwendig noch sinnvoll, eine Mehrheitsentscheidung herbeizuführen. Stattdessen bildet man hier den – wiederum entsprechend der Ähnlichkeiten gewichteten – Mittelwert der gespeicherten Funktionswerte der nächsten Nachbarn.

Definition 12.4.3 (k -NN, kontinuierlich). *Seien E , x_{new} , k , N_k und sim wie bei der vorangegangen Definition. Dann definiere:*

$$\kappa_{E,k} := \frac{\sum_{e=(x,y) \in N_k} sim(x, x_{new})^a \cdot y}{\sum_{e=(x,y) \in N_k} sim(x, x_{new})^a}$$

Üblich ist, dass k vom Verfahren in der Lernphase bestimmt wird, (siehe unten), so dass k Teil der oben eingeführten Verfahrensparameter \mathcal{P} wird.

12.4.1 Die Ähnlichkeitsfunktion

Wie kann man nun die Ähnlichkeit bzw. den Abstand zweier Instanzen berechnen? Falls die Instanzen ausschließlich durch numerische Attribute beschrieben sind, bietet sich die Verwendung des euklidischen Abstandes aus der Geometrie an. Da wir aber auch diskrete Attribute zulassen wollen, müssen wir die Abstandsfunktion etwas allgemeiner formulieren. Dies kann zum Beispiel dadurch geschehen, dass wir als Abstand den Durchschnitt der Abstände entlang der einzelnen Attribute verwenden.

Allgemein verlangt man für eine Abstandsfunktion $dist$ die folgenden Eigenschaften.

1. $dist : X \times X \rightarrow \mathbb{R}^+, dist(x, x) = 0 \forall x \in X,$
2. $dist(x_1, x_2) = dist(x_2, x_1) \forall x_1, x_2 \in X,$
3. $dist(x, y) \leq dist(x, z) + dist(z, y) \forall x, y, z \in X.$

Sind die Bedingungen 1 bis 3 erfüllt, so ist $dist$ eine *Metrik*. Bedingung 3 wird von den meisten Verfahren nicht ausgenutzt, so dass man meistens nur 1 und 2 fordert. Da es üblich ist, den Wertebereich der Funktion $dist$ auf das Intervall $[0, 1]$ zu skalieren, kann man statt einer Distanzfunktion auch eine Ähnlichkeitsfunktion verwenden:

$$dist(x, y) := 1 - sim(x, y) \quad \forall x, y \in X$$

Wird sim wie folgt definiert, so ist das sich daraus ergebende Distanzmaß tatsächlich eine Metrik.

Definition 12.4.4 (Ähnlichkeitsfunktionen). *Seien x_1, x_2 Instanzen aus X , jeweils mit Attributaten A_1, \dots, A_n . Seien w_1, \dots, w_n nicht negative reelle Zahlen, die Attributgewichte. Dann definiere:*

$$sim(x_1, x_2) := \frac{\sum_{i=1, \dots, n} w_i \cdot sim_{A_i}(x_1[i], x_2[i])}{\sum_{i=1, \dots, n} w_i},$$

wobei $x[i]$ den Wert von Attribut A_i im Beispielvektor x bezeichnet.

Für ein numerisches Attribut A mit minimal (maximal) auftretenden Werten min_A (max_A), definiere weiter:

$$sim_A(v_1, v_2) := 1 - \frac{|v_1 - v_2|}{max_A - min_A}$$

Für ein diskretes Attribut A , definiere:

$$sim_A(v_1, v_2) := 1, falls v_1 = v_2, und 0 anderenfalls^5.$$

Die Verwendung geeigneter Attributgewichte w_i ist in den meisten Fällen für den Erfolg des k -NN-Verfahrens ausschlaggebend. Dies liegt daran, dass dieses Verfahren, im Gegensatz z.B. zu Entscheidungsbaumverfahren, standardmäßig alle Attribute in die Abstandsberechnung einbezieht. Sind in den Instanzenbeschreibungen nun Attribute enthalten, die für die gesuchte Vorhersage überhaupt nicht notwendig sind, und daher auch in keiner systematischen Weise mit dem vorherzusagenden Wert korrelieren, so verfälschen die Attributwerte ohne die Verwendung von Gewichten das Abstandsmaß mehr oder weniger stark. Gewichte ermöglichen es nun, statt Attribute völlig aus den Instanzen zu verbannen, ihnen auch graduell eine geringere Bedeutung zu verleihen. Wo möglich sollten diese Attributgewichte vom Benutzer festgelegt werden, viele Verfahren können jedoch diese Attributgewichte auch als Teil ihrer Trainingsphase bestimmen. In diesem Fall werden also dann auch die w_i Teil der oben eingeführten Parameter \mathcal{P} für das k -NN-Verfahren.

Allgemein sollte betont werden, dass die Wahl einer problemadäquaten Abstandsfunktion entscheidend für den Erfolg von instanzbasierten Lernverfahren ist. Wenn auch einfacher Euklidischer Abstand bzw. Funktionen wie die oben definierte üblich sind, so gibt es viele populäre Variationen, wie z.B. aus der Statistik die Mahalanobis-Abstände (siehe z.B. [75]), welche auch die Kovarianzen der Variablen berücksichtigen.

12.4.2 Parameterbestimmung durch Kreuzvalidierung

Bezieht man Parameter wie k oder auch die Gewichte w_i in den Hypothesenraum L_H ein, so kann das k -NN-Verfahren seine Lernphase nicht mehr ganz so „faul“ gestalten, denn nun muss es in der Lernphase diese Parameter ja bestimmen. Ähnlich wie beim Stutzen von Entscheidungsbäumen darf man beim Bestimmen dieser Parameter nun nicht einfach die resultierende Trainingsgenauigkeit zum Maßstab nehmen, denn dadurch droht wieder eine Überanpassung. Es kann z. B. eine unabhängige Evaluierungsmenge zur Schätzung des wahren Fehlers verwendet werden. Bei k -NN-Verfahren kommt dazu üblicherweise die Technik der *Kreuzvalidierung* zum Einsatz, die einen noch besseren Schätzer für den wahren Fehler darstellt als die Verwendung einer einzigen separaten Evaluierungsmenge⁶.

Definition 12.4.5 (m -fache-Kreuzvalidierung). *Sei E eine Beispielmengen, und L ein Lernverfahren. Dann partitioniere E in m möglichst gleichgroße Teilmengen E_1, \dots, E_m . Erzeuge nun die Hypothesen h_1, \dots, h_m wie folgt:*

$$h_i := \text{Ergebnis von } L \text{ auf Basis der Beispielmengen } E \setminus E_i.$$

Dann schätze den wahren Fehler von L auf E wie folgt:

$$\text{error}_{CV(E_1, \dots, E_m)}(L, E) := \frac{\sum_{i=1, \dots, m} \text{error}_{E_i}(h_i)}{m}$$

Es wird also bei der Kreuzvalidierung jede der m Teilmengen der Beispielmengen einmal als Evaluierungsmenge verwendet und dann der Durchschnitt aus den entstehenden Genauigkeiten gebildet. Das bei nur einer Evaluierungsmenge vorhandene Risiko, zufällig nur besonders leicht oder besonders schwer vorherzusagende Instanzen in der Evaluierungsmenge zu haben, wird dadurch vermieden, und es kommt zu einer besseren Schätzung, die um so genauer wird, je größer die Anzahl m der Teilmengen gewählt wird. Da für jede Teilmenge ein Lernlauf durchgeführt werden muss, wählt man bei Lernverfahren mit aufwendiger Lernphase entsprechend kleine Werte von m . Da bei einem einzelnen k -NN-Lauf ohne Parameterschätzung in der Lernphase nichts passiert, kann hier zur Schätzung des wahren Fehlers eine Kreuzvalidierung durchgeführt werden, bei der die Beispielmengen E in einelementige Teilmengen unterteilt wird. Es wird also jeweils genau eine Instanz aus der Trainingsmenge entfernt und als Testinstanz benutzt (*leave-one-out crossvalidation*).

Mit diesem Schätzer für den wahren Fehler kann man nun eine Suche nach den günstigsten Werten für die Parameter durchführen. Am einfachsten ist dies für den Parameter k : hier schätzt man einfach den jeweiligen Fehler, den die k -NN-Methode für einen bestimmten Wertebereich von k , beispielsweise von 1 bis 15 produziert, und wählt dann das günstigste k . Auch für die Attributgewichte kann eine Suche durchgeführt werden, z. B. mit der *hill-climbing* Suchtechnik (s. Kapitel 3). Üblicher ist es allerdings, die Attributgewichte zu bestimmen, indem man einen einfachen statistischen Test der Korrelation des jeweiligen Attributes mit den Zielwerten durchführt. Man gibt dann denjenigen Attributen, die die höchsten Korrelationen aufweisen, die höchsten Gewichte (einen Überblick gibt [93]).

⁶ Weitere Alternativen zur Schätzung des wahren Fehlers diskutiert [79].

12.4.3 Weitere Verfahrensvarianten

Auch die grundlegende k-NN-Methode hat, ähnlich wie die Entscheidungsbäume, im Laufe der Zeit vielfältige Variationen erfahren, um bestimmte Eigenschaften der Methode weiter zu verbessern. Hier sind insbesondere die folgenden Gruppen von Ansätzen zu nennen.

Beispielauswahl. Wie ganz zu Anfang angemerkt, wird die Klassifikationsphase eines k-NN-Verfahrens dadurch verteuert, dass alle Beispiele betrachtet werden müssen. Tatsächlich kann man sich jedoch gut vorstellen, dass nicht alle Beispiele auch wirklich eine wichtige Rolle bei der Auswahl der nächsten Nachbarn für Testinstanzen spielen. Deshalb sind in verschiedenen Arbeiten (siehe z.B. [5]) Vorschläge gemacht worden, wie man in der Lernphase eines k-NN-Verfahrens Instanzen identifizieren kann, die weitgehend ohne Schaden aus der Beispieldmenge entfernt werden könnten. Diese Verfahren liefern dann also als Hypothese nur eine Teilmenge der Beispieldmenge zurück.

Prototyp- und Rechteckansätze. Eine etwas weitergehende Idee stellen Prototyp- und Rechteckansätze dar (siehe z.B. [76, 94]). Während bei der klassischen k-NN-Lernaufgabe, die wir oben definiert haben, nur die Auswahl von Beispielen erlaubt ist, gestattet man bei Prototypverfahren auch die Konstruktion neuer Instanzen, die als Durchschnitt oder Prototypen einer ganzen Gruppe von tatsächlich beobachteten Beispielen dienen, so dass diese Gruppe von Beispielen dann entfernt werden kann. Bei den Rechteckverfahren werden Gruppen von Beispielen nicht durch neukonstruierte Prototypbeispiele ersetzt, sondern durch eine allgemeine Beschreibung des entsprechenden Bereiches des Instanzenraums in Form eines Hyperrechtecks. Diese Rechtecke stellen natürlich eine echte Verallgemeinerung der gesehenen Beispiele dar, so dass diese Verfahren sich schon sehr deutlich von den instanzbasierten Lernverfahren weg bewegen in Richtung auf verallgemeinernde Lernverfahren wie die Entscheidungsbäume.

Mächtigere Instanzenräume. Bei einem instanzbasierten Lernverfahren – wenn es nicht Prototypen oder Rechtecke verwendet – wird auf den Instanzenraum X nur über die Ähnlichkeitsfunktion sim zugegriffen. Instanzbasierte Lernverfahren lassen sich also auch für mächtigere Instanzenräume verwenden, wenn es möglich ist, auf diesen Instanzenräumen entsprechende Ähnlichkeitsfunktionen zu definieren. Dies ist in verschiedenen Arbeiten beispielsweise für prädikatenlogische Instanzenräume durchgeführt worden (siehe z.B. [28, 40]).

12.5 Stützvektormethode

Die Stützvektormethode (engl. Support Vector Machine, SVM) [91] ist ein Lernverfahren, das direkt aus der statistischen Lerntheorie hervorgegangen ist. Im Sinne von SVMs beschreibt diese Theorie, wie gut aus einer gegebenen Menge von Beispielen auf weitere, ungesehene Beispiele geschlossen werden kann. Im einfachsten Fall behandeln auch SVMs die Lernaufgabe des Funktionslernens aus Beispielen⁷. In etwas verfeinertener Notation schreiben wir die Lernmenge E mit l Beispielen nun als $(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_l, y_l)$. Jedes Beispiel besteht also aus einem Merkmalsvektor $\vec{x} \in X$ und einer Klassifikation $y \in Y$. Y enthält die Elemente $+1$ und -1 . Für Instanzen des Begriffs ist $y = +1$ (positives Beispiel), für Nicht-Instanzen $y = -1$ (negatives Beispiel). Der Merkmalsvektor $\vec{x}^T = (x_1, \dots, x_n)^T$ fasst in dieser Notation alle n

⁷ SVMs können auch zur Regression, Dichteschätzung, etc. benutzt werden (siehe [91]).

Merkmalswerte des Beispiels zusammen. Anders als bei der „Standort“-Aufgabe in Abschnitt 12.3, müssen alle Merkmalswerte reelle Zahlen sein. \vec{x}^T ist der transponierte Vektor zu \vec{x} .

SVMs werden besonders dann eingesetzt, wenn die Anzahl der Merkmale groß ist. Die Merkmale könnten z. B. die Bild-Pixel eines Fotos sein, wenn gelernt werden soll das Gesicht einer bestimmten Person zu erkennen. Bei einem Graustufenbild gibt dann der Merkmalswert x_i z. B. die Helligkeit des jeweiligen Pixels durch eine Zahl im Intervall [0..100] wieder. Positive Beispiele sind Fotos der entsprechenden Person, negative Beispiele sind Fotos anderer Personen (oder auch ganz anderer Objekte).

Oft werden SVMs auch zur Textklassifikation eingesetzt (z. B. [42]). Hierunter fällt z. B. die Aufgabe, zwischen WWW-Seiten über Informatik und anderen Seiten zu unterscheiden. Die Merkmale sind hier nicht Bildpixel, sondern das Vorkommen von Wörtern auf der Seite. Für jedes Wort w der Sprache existiert ein zugehöriges Merkmal x_w . Der Wert des Merkmals x_w ist die Häufigkeit mit der das Wort w auf der WWW-Seite vorkommt. Positive Beispiele sind Seiten über Informatik, negative Beispiele sind beliebige andere Seiten.

Das Ziel von SVMs ist es nun aus den gegebenen Beispielen eine Klassifikationsregel abzuleiten, die neue Beispiele gleichen Typs mit größtmöglicher Genauigkeit klassifiziert. Anders gesagt möchte man die Wahrscheinlichkeit, auf einem neuen Beispiel die falsche Klasse vorherzusagen, minimieren. Aber wie erreichen SVMs dieses Ziel?

12.5.1 SVMs und die optimale Hyperebene

In ihrer einfachsten Form lernen SVMs lineare Klassifikationsregeln, d. h. L_H besteht aus Regeln h der Form

$$h(\vec{x}) = \text{sign}\left(b + \sum_{i=1}^n w_i x_i\right) = \text{sign}(\vec{w} \cdot \vec{x} + b)$$

Dabei sind \vec{w} und b die Komponenten der Klassifikationsregel, die von der SVM angepasst werden. \vec{w} ordnet jedem Merkmal ein gewisses Gewicht zu. Man nennt ihn deshalb Gewichtsvektor. Der Merkmalsvektor \vec{x} des zu klassifizierenden Beispiels wird mit \vec{w} durch das Skalarprodukt $\vec{w} \cdot \vec{x} = \sum_{i=1}^n w_i x_i$ verknüpft. b ist ein Schwellwert. $\text{sign}(a) = 1$, wenn a größer als null ist. Ansonsten ist $\text{sign}(a) = -1$. Wenn also der Wert von $\vec{w} \cdot \vec{x}$ plus den Schwellwert b größer als null ist, dann klassifiziert die Klassifikationsregel den Merkmalsvektor $\vec{x}^T = (x_1, \dots, x_i, \dots, x_n)^T$ als positives Beispiel, sonst als negatives Beispiel. In der einfachsten Form findet die SVM eine Klassifikationsregel, die alle Beispiele in der Lernmenge richtig klassifiziert. Diese Forderung kann man wie folgt formal aufschreiben:

$$y_1[\vec{w} \cdot \vec{x}_1 + b] > 0, \dots, y_l[\vec{w} \cdot \vec{x}_l + b] > 0$$

Die SVM muss also den Gewichtsvektor \vec{w} und den Schwellwert b so wählen, dass alle Ungleichungen erfüllt sind. Die SVM tut aber noch etwas mehr und findet eine sehr spezielle Belegung (\vec{w}, b) , die Vapnik die „optimale Hyperebene“ nennt.

Eine Gleichung der Form $\vec{w} \cdot \vec{x} + b = 0$ beschreibt eine Hyperebene. Im zweidimensionalen Raum, wie in dem Beispiel in Abbildung 12.9, entspricht die Klassifikationsregel somit einer

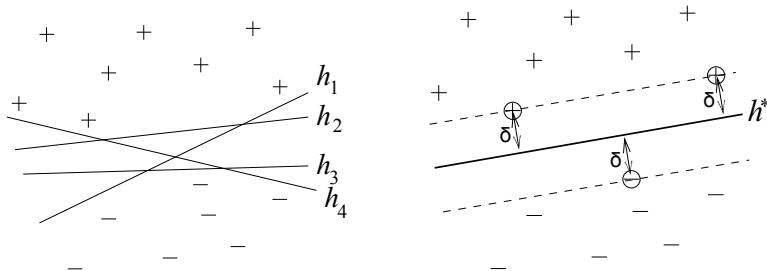


Abbildung 12.9: Ein binäres Klassifikationsproblem mit zwei Merkmalen. Positive Beispiele sind durch +, negative Beispiele durch - gekennzeichnet. Links sieht man, dass alle Hyperebenen h_1 bis h_4 die positiven von den negativen Beispielen trennen. Auf der rechten Seite ist die „optimale Hyperebene“ h^* eingezeichnet, wie sie von der SVM gefunden wird. Sie trennt die Beispiele mit maximalem Abstand.

Geraden. Alle Merkmalsvektoren auf der einen Seite der Hyperebene werden als positive, alle Merkmalsvektoren auf der anderen Seite als negative Beispiele klassifiziert. Wie im linken Teil der Abbildung 12.9 ersichtlich, gibt es viele Geraden, die positive und negative Beispiele fehlerfrei trennen. Aus diesen wählt die SVM die Hyperebene aus, welche positive und negative Beispiele mit maximalem Abstand trennt. Dies ist die Hyperebene h^* im rechten Teil von Abbildung 12.9. Zu jedem Beispiel hat sie mindestens einen Abstand von δ . Formal kann man das Problem des Lernens bei SVMs durch das folgende Optimierungsproblem beschreiben:

$$\text{finde } \vec{w}, b \quad (12.1)$$

$$\text{so dass } \delta \text{ maximal} \quad (12.2)$$

$$\text{und es gilt } y_1 \frac{1}{\|\vec{w}\|} [\vec{w} \cdot \vec{x}_1 + b] \geq \delta, \dots, y_l \frac{1}{\|\vec{w}\|} [\vec{w} \cdot \vec{x}_l + b] \geq \delta \quad (12.3)$$

$\|\vec{w}\| \equiv \sqrt{\vec{w} \cdot \vec{w}}$ bezeichnet die Euklid'sche Länge des Vektors \vec{w} . In Worten lässt sich das Optimierungsproblem wie folgt zusammenfassen. Finde einen Gewichtsvektor \vec{w} und einen Schwellwert b , so dass alle Beispiele der Lernmenge auf der „richtigen“ Seite der Ebene liegen (d. h. die Ungleichungen (12.3) sind erfüllt) und der Abstand δ aller Beispiele von der Trennebene maximal ist. Es sei daran erinnert, dass der Absolutbetrag von $\frac{1}{\|\vec{w}\|} [\vec{w} \cdot \vec{x} + b]$ den Abstand des Punktes \vec{x} von der Hyperebene berechnet. Die am nächsten an der Hyperebene liegenden Beispiele werden Stützvektoren (engl. Support Vectors) genannt. Ihr Abstand zur Hyperebene ist genau δ , d. h. die zugehörige Ungleichung ist per Gleichheit erfüllt. Die drei Stützvektoren sind in Abbildung 12.9 durch Kreise hervorgehoben. Eine interessante Eigenschaft von SVMs ist, dass allein die Stützvektoren die optimale Hyperebene bestimmen. Selbst wenn man alle anderen Beispiele wegließe, käme die SVM zu dem gleichen Ergebnis.

12.5.2 Wie berechnet man die optimale Hyperebene

Im vorangegangenen Abschnitt wurde beschrieben, wie die optimale Hyperebene definiert ist. Obwohl man recht einfach ihre formale Definition hinschreiben kann, wissen wir noch nicht, wie man sie mit einem Computer berechnet. Hierzu bringt man das Optimierungsproblem zunächst

in eine einfachere Form. Da die Länge des Gewichtsvektors $\|\vec{w}\|$ die Lage der Hyperebene nicht beeinflusst, kann sie z. B. durch $\delta = \frac{1}{\|\vec{w}\|}$ fixiert werden. Man substituiert nun alle Vorkommen von δ . Anstelle nun $\frac{1}{\|\vec{w}\|}$ zu maximieren, kann man äquivalent hierzu $\vec{w} \cdot \vec{w}$ minimieren und kommt zu dem folgenden Optimierungsproblem.

$$\text{finde } \vec{w}, b \quad (12.4)$$

$$\text{so dass } \vec{w} \cdot \vec{w} \text{ minimal} \quad (12.5)$$

$$\text{und es gilt } y_1[\vec{w} \cdot \vec{x}_1 + b] \geq 1, \dots, y_l[\vec{w} \cdot \vec{x}_l + b] \geq 1 \quad (12.6)$$

Aber auch dieses Minimierungsproblem ist noch immer schlecht mit numerischen Methoden zu lösen. Deshalb formt man es weiter in das folgende, äquivalente Optimierungsproblem um (eine genaue Herleitung findet sich in [91], Seite 404):

$$\text{finde } \alpha_1, \dots, \alpha_l \quad (12.7)$$

$$\text{so dass } -\sum_{i=1}^l \alpha_i + \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l y_i y_j \alpha_i \alpha_j (\vec{x}_i \cdot \vec{x}_j) \text{ minimal} \quad (12.8)$$

$$\text{und es gilt } \sum_{i=1}^l y_i \alpha_i = 0 \text{ und } \forall i : 0 \leq \alpha_i \quad (12.9)$$

Dieses Optimierungsproblem ist ein quadratisches Programm in Normalform. Es gilt, das Minimum bezüglich der Variablen $\alpha_1, \dots, \alpha_l$ zu finden. Alle α_i müssen hierbei größer oder gleich null sein und die Gleichung in (12.9) muss erfüllt sein. Es existiert eine Reihe von Algorithmen, um Probleme dieser Art zu lösen. Allerdings sind sie oft sehr ineffizient für grosse l . Speziell für SVMs entwickelt wurden u. a. die Algorithmen in [43], [72] (siehe <http://svmlight.joachims.org>). Für allgemeine SVMs (mit Kernfunktionen wie in Abschnitt 12.5.4) haben diese Algorithmen meist ein Laufzeitverhalten von ca. n^2 . Für lineare SVMs existieren allerdings schnellere Algorithmen [44, 83].

Hat man die Lösung $\alpha_1^*, \dots, \alpha_l^*$ des Optimierungsproblems berechnet, kann man die Klassifikationsregel wie folgt ablesen.

$$h(x) = \text{sign} \left(b + \sum_{i=0}^l \alpha_i^* y_i \vec{x}_i \cdot \vec{x} \right) \quad (12.10)$$

Der Gewichtsvektor ergibt sich also als Linearkombination der Beispiele aus der Lernmenge $\vec{w} = \sum_{i=0}^l \alpha_i^* y_i \vec{x}_i$. Hierbei haben nur die Stützvektoren ein $\alpha_i^* > 0$. Den Schwellwert b kann man mit Hilfe eines beliebigen Stützvektors \vec{x}_i bestimmen $b = y_i - \vec{w} \cdot \vec{x}_i$.

12.5.3 Statistische Eigenschaften der optimalen Hyperebene

Wir wissen nun, wie die SVM die Hyperebene berechnet, die positive und negative Beispiele mit dem maximalen Abstand trennt. Aber warum ist dies sinnvoll? Die folgenden zwei Sätze [91] machen Aussagen über den wahren Fehler von SVMs. Sie beschreiben den Erwartungswert des wahren Fehlers $error_D$ einer SVM, die auf $l - 1$ Beispielen trainiert wurde – kurz

$\mathbf{E}[\text{error}_D(h_{l-1}, f)]$. Dies ist der wahre Fehler den die SVM durchschnittlich erreicht, wenn sie mit Trainingsmengen E der Größe $l - 1$ von der Verteilung D und der Funktion f trainiert wird.

Satz 12.5.1. *Der erwartete wahre Fehler $\mathbf{E}[\text{error}_D(h_{l-1}, f)]$ von h_{l-1} , der Hyperebene $\text{sign}(\vec{w} \cdot \vec{x} + b)$ die $l - 1$ Beispiele mit maximalem Abstand trennt, ist beschränkt durch*

$$\mathbf{E}[\text{error}_D(h_{l-1}, f)] \leq \frac{\mathbf{E}[\#SV_l]}{l},$$

wobei $\mathbf{E}[\#SV_l]$ die erwartete Anzahl von Stützvektoren ist, wenn man l Beispiele mit der optimalen Hyperebene trennt. Der Erwartungswert auf der linken Seite der Gleichung ist also über Lernmengen der Grösse $l - 1$, der Erwartungswert auf der rechten Seite über Lernmengen der Grösse l .

Beweis: Dieser Beweis benutzt die Leave-One-Out Methode aus Abschnitt 12.4.2. Mit der Leave-One-Out Methode kann man schätzen, wie gut ungewohnte Beispiele klassifiziert werden. Der Leave-One-Out Fehler $\text{error}_{CV}(\text{SVM}, E)$ ist die Anzahl der fehlerhaft klassifizierten Beispiele geteilt durch l . Er ist eine Schätzung für den wahren Fehler, welche der Lerner nach $l - 1$ Beispielen erreicht. Es ist aus der Statistik bekannt [58], dass $\text{error}_{CV}(\text{SVM}, E)$ und der wahre Fehler nach dem Lernen auf $l - 1$ Beispielen den gleichen Erwartungswert haben, also $\mathbf{E}[\text{error}_D(h_{l-1}, f)] = \mathbf{E}[\text{error}_{CV}(\text{SVM}, E)]$. Der entscheidende Schritt zum Beweis des Satzes ist die Einsicht, dass die SVM nur dann beim Leave-One-Out Test einen Fehler machen kann, wenn das zurückgehaltene Beispiel ein Stützvektor ist. Denn nur Stützvektoren beeinflussen die optimale Hyperebene. Daraus folgt $\text{error}_{CV}(\text{SVM}, E) \leq \frac{\#SV_l}{l}$ und somit $\mathbf{E}[\text{error}_{CV}(\text{SVM}, E)] \leq \frac{\mathbf{E}[\#SV_l]}{l}$, was den Satz beweist. \square

Wenn wir zusätzlich den Schwellwert $b = 0$ fordern, so gilt auch der folgende Satz. Er verbindet den erwarteten wahren Fehler mit der erwarteten Separationsweite.

Satz 12.5.2. *Der erwartete wahre Fehler $\mathbf{E}[\text{error}_D(h_{l-1}, f)]$ von h_{l-1} , der Hyperebene $\text{sign}(\vec{w} \cdot \vec{x} + 0)$ die $l - 1$ Beispiele mit maximalem Abstand trennt, ist beschränkt durch*

$$\mathbf{E}[\text{error}_D(h_{l-1}, f)] \leq \frac{\mathbf{E}[R^2]}{l},$$

wobei δ die Separationsweite und R die Euklid'sche Länge des längsten Merkmalsvektors in der Trainingsmenge ist. Der Erwartungswert auf der linken Seite der Gleichung ist also über Lernmengen der Grösse $l - 1$, der Erwartungswert auf der rechten Seite über Lernmengen der Grösse l .

Besonders bemerkenswert ist, dass in keine der obigen Formeln die Anzahl der Merkmale ein geht. Im Gegensatz zu den anderen hier vorgestellten Lernverfahren können SVMs somit auch bei vielen Merkmalen gut lernen, solange die Aufgabe eine große Separationsweite oder wenige Stützvektoren aufweist. Für ein weiteres Argument zur Generalisierungsgenauigkeit von SVMs sei auf den Abschnitt zum PAC-Lernen verwiesen.

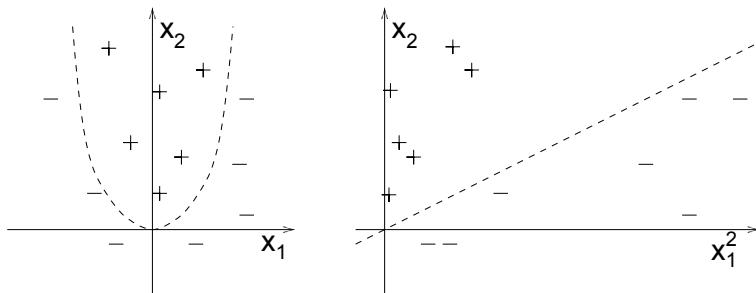


Abbildung 12.10: Die linke Graphik zeigt eine linear nicht trennbare Lernmenge im Eingaberaum (x_1, x_2) . Die rechte Graphik zeigt das gleiche Problem nach der nichtlinearen Transformation, projiziert auf (x_1^2, x_2) . In diesem Raum sind die Beispiele linear trennbar.

12.5.4 Nicht-lineare SVMs durch Kernfunktionen

Bis jetzt konnten wir mit SVMs lediglich lineare Klassifikationsregeln lernen. Für viele Probleme ist das allerdings nicht ausreichend. Sie haben eine nicht-lineare Struktur. Eine herausragende Eigenschaft von SVMs ist, dass man sie sehr einfach in nicht-lineare Klassifikatoren umwandeln kann. Dies geschieht nach dem folgenden Prinzip. Die Merkmalsvektoren \vec{x}_i werden durch eine nichtlineare Funktion $\Phi(\vec{x}_i)$ in einen hochdimensionalen Raum X' abgebildet. In X' lernt die SVM dann eine Hyperebene. Obwohl die trennende Klassifikationsregel in X' linear ist, ergibt sich im Eingaberaum eine beliebig komplexe, nichtlineare Klassifikationsregel. Hierzu folgt ein Beispiel mit zwei Merkmalen x_1, x_2 . Als nichtlineare Abbildung wählen wir $\Phi((x_1, x_2)^T) = (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1)^T$. Obwohl man im Eingaberaum die Beispiele im linken Bild von Abbildung 12.10 nicht linear trennen kann, sind sie nach der Transformation durch $\Phi(\vec{x})$ (rechts) linear trennbar. Dies leistet z. B. der Gewichtsvektor $(-1, 0, 0, 0, \sqrt{2}, 0)$ wie eingezeichnet.

Im Allgemeinen ist eine solche Abbildung $\Phi(\vec{x})$ ineffizient zu berechnen. Boser et al. [15] haben jedoch eine besondere Eigenschaft von SVMs erkannt. Sowohl in der Lernphase, als auch bei der Anwendung der gelernten Klassifikationsregel braucht man lediglich Skalarprodukte in x' zu berechnen, also $\Phi(\vec{x}_i) \cdot \Phi(\vec{x}_j)$. Dies kann für bestimmte Abbildungen $\Phi(\vec{x})$ sehr effizient mit einer sog. Kernfunktion $K(\vec{x}_i, \vec{x}_j) = \Phi(\vec{x}_i) \cdot \Phi(\vec{x}_j)$ geschehen. Mit der Kernfunktion

$$K_{poly}(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + 1)^d$$

kann die SVM so Polynome vom Grad d lernen, mit

$$K_{rbf}(\vec{x}_i, \vec{x}_j) = \exp(\gamma(\vec{x}_i - \vec{x}_j)^2)$$

sog. Radial Basis Klassifikatoren, und mit

$$K_{sigmoid}(\vec{x}_i, \vec{x}_j) = \tanh(s(\vec{x}_i \cdot \vec{x}_j) + c)$$

zweilagige neuronale Netze. Der Einsatz von Kernfunktionen macht SVMs zu einer sehr flexiblen Lernmethode. Für das obige Beispiel ist $K_{poly}(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + 1)^2$ die zugehörige Kernfunktion. Man braucht also die Abbildung $\Phi(\vec{x})$ nicht explizit zu berechnen.

In Formel (12.8) kann nun das Skalarprodukt $\vec{x}_i \cdot \vec{x}_j$ einfach durch den Wert der gewünschten Kernfunktion $K(\vec{x}_i, \vec{x}_j)$ ersetzt werden. Hat man das Optimierungsproblem gelöst, kann man ebenso in der Klassifikationsregel (12.10) das Skalarprodukt $\vec{x}_i \cdot \vec{x}$ durch $K(\vec{x}_i, \vec{x})$ ersetzen und erhält:

$$h(x) = \text{sign} \left(b + \sum_{i=0}^l \alpha_i^* y_i K(\vec{x}_i, \vec{x}) \right)$$

12.5.5 SVMs mit „weicher“ Trennung

Viele reale Lernmengen lassen sich auch mit „komplexen“ Klassifikationsregeln nicht fehlerfrei trennen. Um Trainingsfehler beim Lernen zulassen zu können, benutzt man SVMs mit weicher Trennung [24].

$$\text{finde } \vec{w}, b \quad (12.11)$$

$$\text{so dass } \vec{w} \cdot \vec{w} + C \sum_{i=1}^l \xi_i \text{ minimal} \quad (12.12)$$

$$\text{und es gilt } y_1[\vec{w} \cdot \vec{x}_1 + b] \geq 1 - \xi_1, \dots, y_l[\vec{w} \cdot \vec{x}_l + b] \geq 1 - \xi_l \quad (12.13)$$

$$\xi_1 \geq 0, \dots, \xi_l \geq 0 \quad (12.14)$$

Man minimiert nicht wie zuvor allein die Länge des Gewichtsvektors, sondern zusätzlich die Summe der Fehler ξ_i . Jedes ξ_i misst, wie weit das entsprechende Beispiel im oder jenseits des Separationsbereichs liegt. C ist ein Parameter, der ein Abwägen von Separationsweite und Fehler erlaubt. Das duale quadratische Optimierungsproblem in Standardform für SVMs mit „weicher“ Trennung gleicht dem für SVMs mit „harter“ Trennung, wenn man in Gleichung (12.9) $\forall i : 0 \leq \alpha_i$ durch $\forall i : 0 \leq \alpha_i \leq C$ ersetzt.

Als weiterführende Literatur ist das Tutorium von Burges [20] und die Bücher [23] und [80] zu empfehlen. Das Standardwerk zu SVMs ist [91].

12.6 Lernbarkeit in wahrscheinlich annähernd korrektem Lernen (PAC)

Dieses Kapitel beschäftigt sich mit der Frage, welche Funktionsklassen überhaupt lernbar sind. Es definiert zum einen was es heisst, einen Begriff zu lernen. Zum anderen liefert es Kriterien dafür, welche Aufgaben nicht lernbar sind. Auch wenn die Verbindung von Theorie und Praxis oft nur eingeschränkt möglich ist, sind diese theoretischen Aussagen auch für reale Anwendungen relevant. Ein beweisbar nicht lernbares Problem sollte man nicht angehen, es sei denn man kann zusätzliche Einschränkungen treffen.

Seit Leslie Valiants Artikel hat sich die Komplexitätstheoretische Herangehensweise an das Problem der Lernbarkeit von Begriffen aus Beispielen durchgesetzt [90]. Das von ihm eingeführte Paradigma des wahrscheinlich annähernd korrekten Lernens (*probably approximately correct*

learning – PAC-learning) geht von der Überlegung aus, dass es völlig aussichtslos ist, ein korrektes und vollständiges Lernverfahren zu fordern, das nach einer bestimmten Menge von Eingaben sicher und prompt das richtige Ergebnis ab liefert und dann anhält. Beim *PAC-learning* schwächt man die Anforderung an die Korrektheit des Lernergebnisses ab. Das Lernergebnis ist nur noch mit einer bestimmten Wahrscheinlichkeit von $1 - \delta$ mit einem Fehler von höchstens ϵ richtig. Es wird also nur approximiert, nicht mehr identifiziert. Der Abschwächung bei der Korrektheit stehen aber zwei schwierige Anforderung an das Lernen gegenüber: Das Lernen soll in polynomiell beschränkter Rechenzeit zum Ergebnis kommen, nachdem es eine beschränkte Zahl von Beispielen gesehen hat. Die Beispiele werden hierbei nach einer unbekannten aber festen Wahrscheinlichkeitsverteilung gezogen. Die Lernmenge ist also eine Stichprobe. Das Lernergebnis soll die Begriffsdefinition oder Erkennungsfunktion für den Begriff sein.

Hier wird das Szenario des *PAC-learning* kurz vorgestellt. Eine kurze, übersichtliche Einführung bietet [39], eine ausführliche Behandlung des Bereiches bieten [8, 48, 50, 69].

Für eine gegebene Menge X von Instanzenbeschreibungen (alle möglichen Beispiele) ist ein Begriff c eine Teilmenge von X (nämlich alle Objekte, die der Begriff abdeckt) und eine Begriffs klasse C eine Teilmenge der Potenzmenge von X . Meist wird C durch die Form der Begriffe spezifiziert (z.B. Boolesche Funktionen oder Formeln in einer Normalform mit k Termen). Oft setzt man den Begriff mit der Klassifikation durch den Begriff gleich. Wie beim Funktionslernen ist c dann eine Funktion, die X auf 1 (positives Beispiel) oder 0 (negatives Beispiel) abbildet. Analog zur Repräsentationsklasse C des Zielbegriffs c wird die Repräsentationsklasse H des Lernergebnisses h definiert. H ist der Hypothesenraum, h eine Hypothese. Wenn $h(x) = c(x)$ für alle x einer Lernmenge E ist, so sagt man die Hypothese h ist konsistent mit den gegebenen Beispielen.

Definition 12.6.1 (PAC-Lernbarkeit). *Eine Begriffs klasse C mit Instanzen x der Größe $|x|$ ist (PAC-)lernbar durch einen Hypothesenraum H , wenn es einen Algorithmus L gibt, der*

- *für alle $0 \leq \epsilon \leq 1/2$ und $0 \leq \delta \leq 1/2$*
- *bei allen beliebigen aber festen Wahrscheinlichkeitsverteilungen D über X*
- *für alle $c \in C$*
- *in einer Zeit, die durch ein Polynom über $1/\epsilon, 1/\delta, |c|, |x|$ begrenzt ist*
- *mit einer Wahrscheinlichkeit von mindestens $1 - \delta$*

eine Hypothese $h \in H$ ausgibt, deren wahrer Fehler $\text{error}_D(h, c)$ nicht größer ist als ϵ .

Die Repräsentationsgröße $|c|$ wird unterschiedlich angegeben. Es kann z.B. die Länge einer Repräsentation bei effizienter Codierung sein. Die Forderung der polynomiellen Laufzeit impliziert, dass auch die Anzahl der Beispiele polynomiell beschränkt ist.

In diesem Szenario kann man nun für bekannte Sprachklassen bzw. ihre Automaten die prinzipielle Lernbarkeit von Begriffen, die in dieser Sprache ausgedrückt sind, untersuchen. In den letzten Jahren ist eine Fülle von Beweisen zur Lernbarkeit bestimmter Repräsentationsklassen erarbeitet worden. Interessanterweise ist die Hierarchie der Lernbarkeit nicht dieselbe wie die übliche Komplexitätshierarchie. Das bekannte Beispiel hierfür sind k-KNF und k-Term-DNF.

Definition 12.6.2 (k-KNF). *Eine Formel $C_1 \wedge \dots \wedge C_l$, bei der jedes C_i eine Disjunktion von höchstens k Literalen über Booleschen Variablen ist, ist in k-konjunktiver Normalform (k-KNF).*

Definition 12.6.3 (k-Term-DNF). Eine Formel $T_1 \vee \dots \vee T_k$, bei der jeder Term T_i eine Konjunktion über höchstens n Booleschen Variablen ist, ist in k-Term disjunktiver Normalform (k-Term-DNF).

Natürlich ist k-KNF ausdrucksstärker als k-Term-DNF, denn man kann jeden Ausdruck in k-Term-DNF in k-KNF schreiben, aber nicht umgekehrt. Die Repräsentationsklasse $C = k - KNF$ ist lernbar durch $H = k - KNF$. Die Repräsentationsklasse $C = k - Term - DNF$ ist nicht lernbar durch $H = k - Term - DNF$, wohl aber lernbar durch $H = k - KNF$. Der mächtigere Hypothesenraum, der immer noch polynomiell lernbar ist, erleichtert das Lernen. Der ‚kleinere‘ Hypothesenraum macht das Lernen schwieriger, weil je Beispiel mehr Rechenzeit benötigt wird. Die Lernbarkeit ergibt sich ja einerseits aus der Anzahl der benötigten Beispiele, andererseits aus der Rechenzeit pro Beispiel. Obwohl das Lernen eines Begriffs in der Klasse der k-DNF-Formeln nur polynomiell viele Beispiele braucht, ist er nicht lernbar, weil er im schlimmsten Falle zu lange für die Verarbeitung eines Beispiels braucht. Man kann die Anzahl der Beispiele abschätzen, die man dem Algorithmus geben muss, damit er lernen kann (Stichprobenkomplexität). Damit hat man dann den ersten Schritt eines PAC-Lernbarkeitsbeweises geschafft.

12.6.1 Stichprobenkomplexität

Die Stichprobenkomplexität gibt an, nach wie vielen Beispielen der Lerner einen Begriff spätestens gelernt hat.

Definition 12.6.4 (Stichprobenkomplexität). Die Stichprobenkomplexität eines Algorithmus für gegebenes H , C , ϵ und δ ist die Anzahl von Beispielen l , nach denen der Algorithmus spätestens ein beliebiges $c \in C$ mit Wahrscheinlichkeit von mindestens $1 - \delta$ bis auf einen Fehler von höchstens ϵ approximiert hat.

Nehmen wir an, ein Lernalgorithmus gibt nur die Hypothesen aus, die alle positiven Beispiele und kein negatives Beispiel abdecken. Wenn wir weiterhin annehmen, dass der Zielbegriff in der Hypothesensprache enthalten ist, dann können wir recht einfach die Stichprobenkomplexität berechnen. Zunächst betrachten wir Hypothesenräume mit endlicher Größe $|H|$.

Satz 12.6.1 (Stichprobenkomplexität endlicher Hypothesenräume). Für endliche Hypothesenräume H mit $H = C$ kann spätestens nach

$$l \geq \frac{1}{\epsilon} (\ln |H| + \ln(1/\delta)) \quad (12.15)$$

Beispielen jedes $c \in C$ mit Wahrscheinlichkeit von mindestens $1 - \delta$ bis auf einen wahren Fehler von höchstens ϵ gelernt werden.

Beweis: David Haussler hat die Formel 12.15 anhand des Versionenraums bewiesen [38] (ähnliche Schranken wurden zuvor auch von Vapnik bewiesen [91]). Der Versionenraum enthält ja zu jedem Zeitpunkt nur mit den bisher gesehenen Beispielen konsistente Hypothesen und der Hypothesenraum ist endlich. Damit erfüllt er genau die Annahmen. Der Beweis betrachtet den schlimmsten Fall: bezüglich der bisher gesehenen Beispiele ist eine Hypothese im Versionenraum zwar korrekt, tatsächlich ist ihr Fehler aber größer als ϵ . Die Wahrscheinlichkeit, dass eine

beliebige Hypothese mit Fehler größer als ϵ ein zufällig gezogenes Beispiel richtig klassifiziert, ist also höchstens $1 - \epsilon$. Bei l Beispielen beträgt die Wahrscheinlichkeit also höchstens $(1 - \epsilon)^l$. Bei k Hypothesen ist die Wahrscheinlichkeit, dass eine von ihnen schlecht ist (d. h. einen Fehler größer als ϵ hat), höchstens $k(1 - \epsilon)^l$. k kann natürlich nicht größer sein als der Hypothesenraum. Im schlimmsten Fall ist die Wahrscheinlichkeit, dass eine der Hypothesen im Versionenraum einen Fehler größer als ϵ hat, also $|H| (1 - \epsilon)^l$. Dies beschränkt die Wahrscheinlichkeit, dass l Beispiele nicht ausreichten, um die schlechten Hypothesen aus dem Versionenraum zu tilgen. Da ϵ zwischen 0 und 1 liegt, können wir sie abschätzen als $|H| e^{-\epsilon l}$ (oft additive Chernoff-Schranke genannt). Damit ein Begriff PAC-gelernt wird, muss diese Wahrscheinlichkeit kleiner sein als δ :

$$|H| e^{-\epsilon l} \leq \delta$$

Eine Umformung ergibt genau 12.15. □

Bei unendlichen Hypothesenräumen, wie z. B. den Hyperebenen der SVM, kann man die Größe nicht mehr durch einfaches Abzählen angeben. Allerdings gibt es auch in unendlichen Hypothesenräumen oft nur endlich viele Hypothesen, die sich wirklich substantiell unterscheiden. Die Ausdrucksstärke auch unendlicher Hypothesenräume gibt die **Vapnik-Chervonenkis-Dimension** (VC-Dimension) an. Sie misst die Kapazität („Größe“) eines Hypothesenraums und kann auch bei unendlichen Hypothesenräumen verwendet werden, um die Stichprobenkomplexität zu berechnen. Die VC-Dimension gibt an, wie viele Beispiele man maximal mit Hypothesen aus H beliebig aufteilen (d. h. zerschmettern) kann.

Definition 12.6.5 (Zerschmettern einer Beispielmenge). *Sei H der Hypothesenraum über X und S eine m -elementige Teilmenge von X . S wird von H zerschmettert (shattered), falls es für alle $S' \subseteq S$ eine Hypothese $h_{S'} \in H$ gibt, die genau S' abdeckt, d.h. $S \cap h_{S'} = S'$.*

Alle Teilmengen von S werden also durch Hypothesen in H erkannt.

Definition 12.6.6 (Vapnik-Chervonenkis-Dimension). *Die Vapnik-Chervonenkis-Dimension von H , $VCdim(H)$, ist die Anzahl der Elemente der größten Menge $S \subseteq X$, die von H zerschmettert wird.*

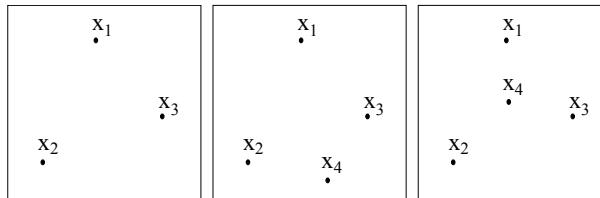
$$VCdim(H) = \max \{m : \exists S \subseteq X, |S| = m, H \text{ zerschmettert } S\}$$

Wenn es kein Maximum der Kardinalität von S gibt, ist $VCdim$ unendlich.

Satz 12.6.2 (VC-Dimension endlicher Hypothesenräume). *Wenn der Hypothesenraum H endlich ist, dann ist $VCdim(H) \leq \log_2(|H|)$.*

Beweis: Um eine Menge der Größe m zu zerschmettern, sind mindestens 2^m verschiedene Hypothesen nötig, weil es ja 2^m verschiedene Teilmengen gibt. □

Wie bestimmt man die VC-Dimension eines unendlichen Hypothesenraumes? Das folgende Beispiel untersucht die VC-Dimension von Hyperebenen, wie sie schon bei der SVM eingeführt wurden.

Abbildung 12.11: 3 und 4-elementige Teilmengen von X .

- X : Punkte in einer Ebene, dargestellt durch $\vec{x} = (x_1, x_2)$;
- H : Hyperebenen $w_1x_1 + w_2x_2 + b$ mit zwei Gewichten w_1 und w_2 und einem Schwellwert b . Jeder Hyperebene entspricht die Boolese Funktion $h(\vec{x}) = 1$ gdw. $w_1x_1 + w_2x_2 + b > 0$.

Für eine 3-elementige Teilmenge S von X gibt es 2^3 Möglichkeiten, die Elemente von S in positive und negative Beispiele zu klassifizieren (d. h. zu zerschmettern). Die 8 verschiedenen Aufteilungen sind:

S	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8
(x_1, y_1)	-	+	-	+	-	+	-	+
(x_2, y_2)	-	-	+	+	-	-	+	+
(x_3, y_3)	-	-	-	-	+	+	+	+

Für die drei Beispielvektoren \vec{x}_1 , \vec{x}_2 und \vec{x}_3 im linken Bild von Abbildung 12.11 können alle 8 Aufteilung durch Hyperebenen realisiert werden. Wie bei der SVM kann man sich eine Hypothese aus H als trennende Linie zwischen positiven und negativen Beispielen vorstellen. $VCdim(H)$ ist also mindestens 3. Aber könnte H nicht auch eine 4-elementige Teilmenge von X zerschmettern? Abbildung 12.11 zeigt, wie man 4 Punkte in der Ebene entweder konvex (Abb. 12.11, Mitte) oder nicht konvex (Abb. 12.11, rechts) anordnen kann. Im ersten Fall können $\{\vec{x}_1, \vec{x}_4\}$ und $\{\vec{x}_2, \vec{x}_3\}$ nicht durch eine Linie getrennt werden. Ebenso lässt sich im zweiten Fall $\{\vec{x}_4\}$ nicht von den anderen drei Beispielen linear trennen. Dies illustriert, dass die $VCdim(H)$ nicht nur mindestens 3, sondern genau 3 ist.

Im Allgemeinen gilt, dass die VC-Dimension von Hyperebenen im n -dimensionalen Raum gleich $n + 1$ ist. Man kann aber in vielen Fällen eine wesentlich kleinere $VCdim$ erhalten, wenn man die Separationsweite δ (siehe Abschnitt über SVMs) betrachtet.

Satz 12.6.3. [91] Für Hyperebenen $h(\vec{x}) = \text{sign}(\vec{w} \cdot \vec{x} + b)$ im n -dimensionalen Raum, die l Beispiele wie folgt trennen,

$$\forall_{i=0}^l : \frac{1}{\|\vec{w}\|} |\vec{w} \cdot \vec{x}_i + b| \geq \delta \text{ und } \|\vec{x}_i\| \leq R$$

gilt, dass ihre $VCdim$ nicht größer ist als

$$\min \left(\frac{R^2}{\delta^2}, n \right) + 1$$

In vielen Fällen ist es schwierig, die $VCdim$ genau zu bestimmen. Oft werden nur Abschätzungen gefunden. Den Zusammenhang zwischen der VC-Dimension einer Begriffsklasse und ihrer Stichprobenkomplexität geben die folgenden Ergebnisse an:

Satz 12.6.4. $C = H$ hat polynomielle Stichprobenkomplexität gdw. $VCdim(C)$ endlich ist.

Analog zur Hypothesenzahl in endlichen Hypothesenräumen (Gleichung 12.15) kann man mit der VC-Dimension die Stichprobenkomplexität auch bei unendlichen Hypothesenräumen abschätzen.

Satz 12.6.5 (Stichprobenkomplexität unendlicher Hypothesenräume). *Für unendliche Hypothesenräume H mit $H = C$ kann spätestens nach*

$$l \geq \frac{1}{\epsilon} \left(4 \log_2 \left(\frac{2}{\delta} \right) + 8 \text{VCdim}(H) \log_2 \left(\frac{13}{\epsilon} \right) \right)$$

Beispielen jedes $c \in C$ mit Wahrscheinlichkeit von mindestens $1 - \delta$ bis auf einen wahren Fehler von höchstens ϵ gelernt werden.

Wir können somit angeben, welche Anzahl von Beispielen ausreicht, um einen Begriff zu lernen. Allerdings ist die so berechnete Stichprobenkomplexität oft unrealistisch hoch. Meist reichen weit weniger Beispiele für ein gutes Lernergebnis aus. Des Weiteren muss man sehr genau prüfen, ob die im PAC-Lernen gemachten Annahmen in der Anwendung wirklich zutreffen.

12.7 Lernen aus strukturierten Daten: Logik

Viele der grundlegenden Verfahren des maschinellen Lernens sind zunächst nur für tabellarische Eingabedaten (Zeilen und Spalten einer einzelnen Tabelle wie in Tabelle 12.1) konstruiert worden (*feature-vector* oder *attribute-value* Daten). Viele natürlich vorkommende Objekte lassen sich jedoch nicht als einzelne Spalte einer Tabelle mit einer festen Zahl von Merkmalen darstellen. Dies gilt zum Beispiel für chemische Moleküle und andere Objekte, die inhärent die Struktur von Bäumen oder Graphen haben, aber auch für einfache Objekte wie Zeichenketten, oder noch komplexere wie *relationale* Daten, bei denen Objekte über verschiedene logische Relationen dargestellt werden. Im maschinellen Lernen hat sich deshalb das Gebiet des Lernens aus solchen, allgemein *strukturiert* genannten Daten, enorm an Bedeutung gewonnen.

Dabei ist die Anpassung eines Lernverfahrens an die Verwendung strukturierter Daten je nach Verfahren unterschiedlich komplex. Bei all diejenigen Verfahren, die auf Abstandsmaßen, Ähnlichkeits- oder Kernfunktionen basieren (siehe Abschnitt 12.4 und Abschnitt 12.5), also letzten Endes auf dem Vergleich zweier Objekte, ist eine Anpassung des grundlegenden Verfahrens nicht notwendig, denn die Behandlung strukturierter Objekte kann in der jeweiligen Vergleichsfunktion erfolgen. Dementsprechend konzentrieren sich viele Arbeiten des maschinellen Lernens nunmehr auf die Formulierung entsprechender Funktionen, vorwiegend Kernfunktionen, die dann in jedem Kern-basierten Lernansatz verwendet werden können (siehe z.B. [32] für eine etwas älteren Überblick).

In diesem Abschnitt wollen wir uns demgegenüber auf Verfahren für logische Repräsentationen konzentrieren, bei denen die zu Grunde liegenden Algorithmen direkt mit der strukturierten Repräsentation arbeiten. Solche Verfahren sind seit den 1990er Jahren im Gebiet der so genannten *Induktiven Logikprogrammierung* entwickelt worden. Wir führen im folgenden die dazu notwendigen Grundlagen ein. Besonders interessante Weiterentwicklungen sind speziell die sogenannten statistisch-relationalen Verfahren, bei denen die logische Grundlage mit probabilistischen Informationen verbunden wird (siehe z.B. [34]).

12.7.1 Repräsentation

Zur Einführung der logischen Repräsentation wollen wir die Domäne der Verwandtschaftsbeziehungen verwenden. Wie kann zum Beispiel der Begriff der *Ehe* oder der *Grossmutter* definiert werden? Wir können natürlich bei einem Menschen in Form von Attributen angeben, wer der Ehepartner ist und wer seine Großmütter sind. Beschränken wir uns auf den Begriff der Großmutter mütterlicherseits, so zeigt Tabelle 12.4 zwei Beispiele für diesen Begriff: Bertha ist die Großmutter von Doris und Bettina die von Dieter. Wir können also die Beispiele in Form von Attributen aufschreiben. Aber wie können wir den allgemeinen Begriff darstellen? Dies gelingt mit Relationen sehr einfach. In Prädikatenlogik können wir Relationen und ihre Verknüpfungen darstellen. Wir schreiben die Beispiele als zweistellige Prädikate:

$$\begin{array}{lll} \text{verheiratet(doris, dieter)} & \text{mutter(doris, christa)} & \text{mutter(christa, berta)} \\ \text{verheiratet(christa, christoph)} & \text{mutter(dieter, carla)} & \text{mutter(carla, bettina)} \\ \text{verheiratet(carla, carlos)} & & \end{array}$$

Den allgemeinen Begriff können wir als prädikatenlogische Formel schreiben:

$$\text{mutter}(X, Y) \wedge \text{mutter}(Y, Z) \rightarrow \text{oma}(X, Z)$$

Verwandschaftsbeziehungen sind also typische relationale Begriffe, aber auch andere Strukturen (z.B. Graphen) können leicht durch solche Relationen dargestellt werden.

Um relationale Begriffe ausdrücken zu kennen, brauchen wir eine eingeschränkte Prädikatenlogik. Dies kann z.B. eine terminologische Logik (Beschreibungslogik) sein oder eben ein *logisches Programm* aus Hornklauseln (siehe Kapitel 5) in der induktiven logischen Programmierung [70].

Als Lernaufgabe wollen wir im folgenden nicht direkt die Funktionsleraufgabe aus Abschnitt 12.2 verwenden, sondern die grundlegender Aufgabe, eine *konsistente Hypothese* zu finden, die also alle gegebenen Beispiel korrekt klassifiziert. Andererseits wird bei logischem Lernen in der Regel eine weitere Komponente in die Lernaufgabe eingebracht, die die Möglichkeiten solcher Lerner entscheidend erweitert: das so genannte *Hintergrundwissen*. Dabei handelt

Tabelle 12.4: Beispiele für den Begriff der Großmutter mütterlicherseits.

Person	Partner	Mutter	Oma mütterlich
Doris	Dieter	Christa	Berta
Dieter	Doris	Carla	Bettina
Christa	Christoph	Berta	Andrea
Carla	Carlos	Bettina	Angela

es sich um Menge logischer Ausdrücke, die dem Lernverfahren als Vorwissen vom Nutzer mitgegeben werden können, und mit denen dann erheblich komplexere Probleme gelernt werden können.

Formal wird das im logischen Lernen über den logischen Folgerungsbegriff abgebildet, so dass die Lernaufgabe wie folgt definiert ist:

Gegeben:

positive Beispiele E^+ und negative Beispiele E^- in einer Sprache L_E , $E := E^+ \cup E^-$,
Hintergrundwissen B in einer Sprache L_B , wobei $B \cup H \not\models \perp$

Ziel: eine Hypothese H in einer Sprache L_H , so dass

$B, H, E \not\models \perp$ (Konsistenz),
 $B, H \models E^+$ (Vollständigkeit),
 $\forall e \in E^- : B, H \not\models e$ (Korrekttheit)

Meist besteht sowohl B als auch E aus variablenfreien Fakten. L_H ist eine eingeschränkte Prädikatenlogik.

12.7.2 Algorithmus FOIL

Wir wollen nun ein einfaches Verfahren für das lernen aus relational-logischen Daten einführen, nämlich das System FOIL [73], welches – ähnlich wie Entscheidungsbaumverfahren – das heuristische Maß des Informationsgewinns verwendet, jedoch logische Regeln erzeugt. Die Hypothesensprache L_H für FOIL umfasst funktionsfreie Klauseln, deren Kopf aus genau einem positiven Literal besteht und deren Körper aus positiven oder negierten Literalen besteht. Terme müssen mit dem Klauselkopf verbunden sein. Beispiele werden in Form von variablen- und funktionsfreien Fakten gegeben. Wenn keine negativen Beispiele vorliegen, berechnet FOIL alle möglichen, nicht als positive Beispiele gegebenen Fakten als negative Beispiele (*closed-world assumption*). Das Hintergrundwissen liegt in Form von extensionalen Definitionen von Prädikaten in Form von variablen- und funktionsfreien Fakten vor. Wie RDT erzeugt auch FOIL die Hypothesen von der allgemeinsten zu den spezielleren. Im Gegensatz zu RDT durchsucht FOIL den Hypothesenraum nur dort, wo ein Informationsgewinn erwartet wird. Dazu wird eine Heuristik verwendet, die – ähnlich wie bei ID3 – auf der Entropie basiert.

FOIL erstellt aus den gegebenen Beispielen für ein k -stelliges Zielprädikat eine Menge von k -stelligen Tupeln, wobei jedes Tupel eine Werteverteilung der k Variablen angibt. Die Tupel sind als positive oder negative Beispiele klassifiziert. Die Tupelmenge ist vollständig, d.h. alle möglichen Werteverteilungen kommen entweder als positives oder als negatives Beispiel vor. Der Algorithmus besteht aus zwei Schleifen, einer äußeren und einer inneren. Die äußere Schleife findet für positive Beispiele bzw. Tupel T^+ eine Klausel, die sie abdeckt. Die abgedeckten positiven Beispiele werden aus der Tupelmenge entfernt und für die verbleibenden positiven Beispiele wird eine weitere Klausel gelernt. Es werden so viele Klauseln gelernt, dass insgesamt von der Menge der Klauseln alle positiven Beispiele abgedeckt werden.

Bis T^+ keine Tupel mehr enthält:

- finde eine Klausel, die positive Beispiele abdeckt,
- entferne alle Tupel aus T^+ , die von dem Klauselkörper abgedeckt werden.

In der inneren Schleife wird der Körper einer Klausel durch Literale erweitert, die Klausel also spezialisiert. Ein zusätzliches Literal wird aufgrund des Informationsgewinns ausgewählt, den es für die Charakterisierung des Zielprädikats erbringt.

- Initialisiere T_i mit den Beispielen in Tupeldarstellung und setze $i = 1$
- Solange T^- Tupel enthält:
 - finde ein nützliches Literal L_i und füge es dem Klauselkörper hinzu,
 - erstelle eine neue Tupelmenge T_{i+1} , die nur diejenigen Beispiele enthält, für die L_i gilt. Falls L_i eine neue Variable einführt, werden die Tupel um diese erweitert. Die Klassifikation der Tupel in T_{i+1} bleibt diejenige von T_i .
 - setze i auf $i + 1$

Die Information, die man bei einer Tupelmenge T_i mit p_i positiven und n_i negativen Beispielen braucht, um zu entscheiden, ob ein Tupel positiv ist, ist

$$I(T_i) = -\log_2 \frac{p_i}{p_i + n_i}$$

Ziehen wir den Informationsgehalt der neuen Tupelmenge von dem der alten Tupelmenge ab, so erhalten wir den Informationsgewinn durch das neu eingeführte Literal. Dieser Informationsgewinn wird durch die Anzahl der positiven Tupel, die durch das neue Literal abgedeckt werden, p_{i+1} , gewichtet. Der gewichtete Informationsgewinn durch ein neues Literal L_i bezogen auf die Tupelmenge T_i wird also durch die folgende Formel ausgedrückt:

$$Gain(L_i) = (p_{i+1} + n_{i+1})(I(T_i) - I(T_{i+1}))$$

Ein einfaches Beispiel ist das Lernen der Relation *tochter* aus den folgenden Beispielen, wobei vor dem Semikolon die positiven Tupel, nach dem Semikolon die negativen Tupel angegeben werden. Das Hintergrundwissen ist durch die Relationen *eltern* und *weibl* gegeben. Das Beispiel ist [57] entnommen.

tochter(name,name)	eltern(name,name)	weibl(name)
mary, ann	ann, mary	mary
eve, tom ;	ann, tom	ann
tom, ann	tom, eve	eve
eve, ann	tom, ian	

Die erste Menge T_1 enthält alle 4 Tupel, $p_1 = 2, n_1 = 2$. Ihr Informationsgehalt ist $-\log_2 \frac{2}{4} = 1$. Die Klausel enthält nur das Zielprädikat *tochter*(X, Y) als Klauselkopf. Wird nun *weibl* als erstes Literal in den Klauselkörper eingetragen, so enthält die neue Tupelmenge T_2 nicht mehr das negative Beispiel *tom, ann*, da *weibl(tom)* nicht gilt. Damit ist die neue Tupelmenge geordneter: $I(T_2) = 0,58$. Alle positiven Beispiele bleiben abgedeckt. Der Informationsgewinn von *weibl* ist also:

$$Gain(\text{weibl}(X)) = 2(1 - 0,58) = 0,84$$

Wird nun noch $\text{eltern}(Y, X)$ als nächstes Literal hinzugefügt, so brauchen wir bei der resultierenden Tupelmenge T_3 keine weitere Information für die Klassifikation: $I(T_3) = 0$. Der Informationsgewinn von eltern ist:

$$\text{Gain}(\text{eltern}(Y, X)) = 2(0, 58 - 0) = 1, 16$$

Die Klausel $\text{eltern}(Y, X) \wedge \text{weibl}(X) \rightarrow \text{tochter}(X, Y)$ deckt alle positiven Beispiele ab und kein negatives. Da in der Menge T_3 kein negatives Beispiel mehr enthalten ist, endet die innere Schleife. Damit ist der erste Schritt der äußeren Schleife abgeschlossen und es werden alle positiven Beispiele aus der Menge T_1 entfernt, die von der Klausel abgedeckt werden. Da dies alle positiven Beispiele sind, endet auch die äußere Schleife.

Da FOIL den Hypothesenraum gemäss der Heuristik des Informationsgewinns durchsucht, kann es eine richtige Lösung des Lernproblems verpassen. Dies ist der Fall, wenn ein Literal mit geringem Informationsgewinn gewählt werden muss, das eine neue Variable einführt, die für die Wahl geeigneter weiterer Literale nötig ist. In sehr vielen Fällen führt die Heuristik jedoch dazu, dass FOIL sehr schnell die richtige Lösung findet.

12.8 Assoziationsregeln

Der folgende Abschnitt behandelt eine der im Data Mining zurzeit populärsten Analyseaufgaben, nämlich das Entdecken von *Assoziationsregeln* in Datenbanken [4]. Im Gegensatz zu den in den vorangegangenen Abschnitten behandelten Lernaufgaben ist diese Lernaufgabe keine Spezialisierung des Funktionslernens aus Beispielen. Wir können dies leicht anhand der vielleicht bekanntesten Anwendung von Assoziationsregelverfahren, der sogenannten *Warenkorbanalyse*, erläutern. In den meisten Supermärkten werden zur Beschleunigung des Kassiervorganges und Erleichterung der Bestandsplanung heute Scannerkassen eingesetzt. Ein solches Kassensystem erfasst die auf jeder Verpackung in Form eines Strichcodes aufgedruckte europäische Artikelnummer (EAN), welche jeden Artikel eindeutig identifiziert. Bereits in einem mittleren Supermarkt sind Zehntausende derart unterschiedener Artikel im Angebot, denn es werden nicht nur Hersteller und Produkt, sondern auch unterschiedliche Packungstypen und -größen unterschieden. Anhand der erkannten Artikelnummer kann so zum einen mit Hilfe der im Kasserechner gespeicherten Preistabelle automatisch der Kassenzettel erstellt werden, zum anderen kann in einer Datenbank abgelegt werden, aus welchen Artikeln der jeweilige Einkauf (auch *Transaktion* genannt) bestanden hat. Es entstehen so sehr schnell recht große Datenbanken, in denen sich das Kaufverhalten der Kunden widerspiegelt und die darum zur Optimierung der Geschäftsprozesse analysiert werden sollten.

Eine wichtige Frage bei solchen Analysen ist, welche Artikel unter welchen Bedingungen bei einer Transaktion gemeinsam gekauft werden. Interessant könnte also beispielsweise folgende Aussage sein:

Falls in einem Einkauf Bier und Pizza zusammen gekauft werden, so ist es wahrscheinlich, dass auch Kartoffelchips gekauft werden.

Solche *Assoziationsregeln* können in einem Supermarkt beispielsweise zur Planung der Warenanordnung genutzt werden, so dass also beispielsweise Kartoffelchips in die Nähe von Bier und

Pizza platziert werden. Weitere Anwendungen ergeben sich im Versand- und Internethandel. Stellt man beispielsweise fest

Kunden, die einen Farbdrucker und Bildbearbeitungssoftware bestellen, bestellen wahrscheinlich auch einen Scanner.

so bietet es sich an, denjenigen Kunden, die bisher nur einen Drucker und Bildbearbeitungssoftware gekauft haben, per direktem Brief oder E-Mail ein günstiges Angebot für einen Scanner zu machen.

Wollte man diese Lernaufgabe als Funktionslernen aus Beispielen auffassen, so könnte man versuchen, für jeden Artikel eine Funktion lernen zu lassen, die, gegeben eine partielle Transaktion, vorhersagt, ob der betreffende Artikel ebenfalls in dieser Transaktion vorkommen wird oder nicht. Da wir uns nicht für die Vorhersage eines bestimmten Artikels interessieren, sondern für jede zusätzliche Marketing-Chance, müssten also Zehntausende von Funktionslernaufgaben gelöst und ihre Ergebnisse vom Benutzer analysiert werden. Dies verbietet sich durch den entstehenden hohen Aufwand. Aber selbst wenn wir annehmen, dass wir nur an einem einzigen Artikel interessiert sind, ergibt sich ein weiteres Problem. Es dürfte sich kaum, wie beim Funktionslernen angestrebt, für den gesamten möglichen Instanzenraum aller partiellen Transaktionen verlässlich vorhersagen lassen, ob beispielsweise Kartoffelchips in dieser Transaktionen enthalten sein werden. Dennoch kann es sein, dass für einzelne Bereiche des Instanzenraums eine sinnvolle Aussage möglich ist, z. B. für die Teilmenge der Transaktionen, die Bier und Pizza enthalten. Auch wenn *global* also keine Vorhersage möglich ist, so sind wir doch stark an Aussagen wie der obigen interessiert, welche *lokal* interessante Zusammenhänge *beschreiben*. Solche Lernaufgaben werden deshalb auch als *deskriptive* Lernaufgaben bezeichnet. Da ein beschreibendes Lernverfahren nicht zu einer Aussage über den gesamten Instanzenraum gezwungen ist, kann es seine Suche besser auf das Finden lokal interessanter Zusammenhänge ausrichten. Die Gesamtheit der entdeckten Zusammenhänge ergibt andererseits darum aber nicht notwendigerweise ein globales Modell.

Als eine Spezialisierung der beschreibenden Lernaufgaben können wir das Entdecken von Assoziationsregeln wie folgt präzisieren.

Definition 12.8.1 (Finden von Assoziationsregeln). *Sei I eine Menge von Objekten („items“) und sei T eine Menge von Transaktionen, wobei $\forall t \in T : t \subseteq I$. Sei weiterhin $s_{min} \in [0, 1]$ eine benutzergegebene Minimalhäufigkeit („minimal support“) und $c_{min} \in [0, 1]$ eine benutzergegebene Minimalkonfidenz. Bei der Lernaufgabe Finden von Assoziationsregeln sind, gegeben I , T , s_{min} und c_{min} , alle Regeln der folgenden Form zu finden:*

$$X \rightarrow Y$$

wobei $X \subseteq I$ und $Y \subseteq I$ und $X \cup Y = \emptyset$, und es gilt:

$$s(r) := \frac{|\{t \in T \mid X \cup Y \in t\}|}{|T|} \geq s_{min}$$

sowie

$$c(r) := \frac{|\{t \in T \mid X \cup Y \in t\}|}{|\{t \in T \mid X \in t\}|} \geq c_{min}$$

In dieser Definition ist also durch zwei Bedingungen spezifiziert worden, wann eine Assoziationsregel als Lösung zulässig ist. Die erste Bedingung verlangt eine gewisse Minimalhäufigkeit s_{min} für die in einer Regel vorkommenden Artikel. Wir sind also nicht an lokalen Zusammenhängen interessiert, die nicht mindestens in einem Anteil von s_{min} aller Transaktionen vorkommt. Die zweite Bedingung spezifiziert, was es bedeutet, dass Artikel „wahrscheinlich“ zusammengekauft werden, und verlangt, dass von den Transaktionen, die die Prämisse X beinhalten, mindestens ein Anteil c_{min} auch die Konklusion Y beinhalten soll. Typische Werte sind beispielsweise $s_{min} = 0,01$ und $c_{min} = 0,5$. Finden wir also in der Artikelmenge $I = \{Bier, Pizza, Chips, Wein, \dots\}$ mit diesen Einstellungen die Assoziationsregel

$$\{Bier, Pizza\} \rightarrow \{Chips\},$$

so heißt das, dass mindestens 1 % der Käufer Bier, Pizza und Chips gekauft haben, und dass mindestens 50% der Bier- und Pizza-Käufer auch Chips mitgenommen haben.

12.8.1 Der Apriori-Algorithmus

Verfahren zur Entdeckung von Assoziationsregeln können die Tatsache ausnutzen, dass der Raum aller Assoziationsregeln geordnet werden kann, und dass sich aus dieser Ordnung effiziente Suchstrategien ableiten lassen [4]. Betrachten wir dazu, was passiert, wenn wir eine Assoziationsregel durch Hinzufügen eines weiteren Objektes in der Prämisse oder der Konklusion verlängern (Abbildung 12.12).

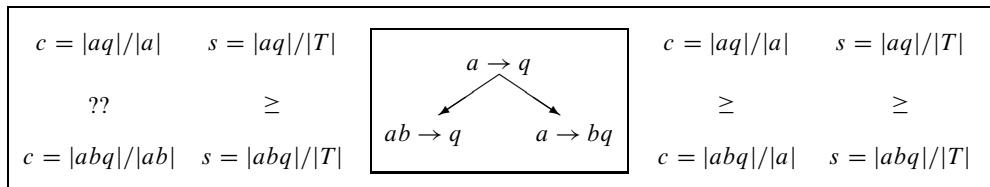


Abbildung 12.12: Absteigende Suche.

Betrachten wir zunächst in der linken Hälfte der Abbildung, wie sich Häufigkeit und Konfidenz bei Erweiterung der Prämisse verändern. Wie dargestellt, kann die Häufigkeit bei dieser Verfeinerung nur abnehmen, während die Konfidenz sowohl steigen als auch sinken kann, je nachdem, wie sich der hinzugefügte Artikel b auf die Transaktionen mit a und q verteilt. Bei Verlängerung der Konklusion können wir im rechten Teil der Abbildung feststellen, dass sowohl Häufigkeit als auch Konfidenz nur abnehmen können. Dies bedeutet, dass wir bei einer Suche von kürzeren zu längeren Regeln die Suche nur anhand der Häufigkeit beschneiden können: wenn eine Assoziationsregel nicht die nötige Häufigkeit erreicht, wissen wir, dass dies auch für alle Nachfolger nicht der Fall sein wird. Diese Nachfolger brauchen daher nicht betrachtet zu werden. Wir können in der obigen Abbildung noch eine weitere Beobachtung machen, die für den Entwurf eines Assoziationsregelverfahrens zentral ist. Aus der Definition der Häufigkeit ergibt sich unmittelbar, dass die Verteilung der Artikel auf Prämisse oder Konklusion unerheblich ist. Es kommt nur auf die Gesamtmenge aller in der Regel enthaltenen Artikel an. Wir brauchen also die beiden in der Abbildung dargestellten Zweige nicht zu unterscheiden, sondern können zunächst einfach im Raum der Artikelmengen suchen, von den einelementigen Mengen zu immer

$$\begin{array}{ll}
 c = |abc|/|ab| & s = |abc|/|T| \\
 \geq & = \\
 c = |abc|/|a| & s = |abc|/|T|
 \end{array}
 \quad
 \boxed{\begin{array}{c} ab \rightarrow c \\ \downarrow \\ a \rightarrow bc \end{array}}$$

Abbildung 12.13: Regelerzeugung.

größeren Mengen. Wir merken uns all diejenigen Mengen, die die notwendige Mindesthäufigkeit erreichen (die sogenannten *häufigen* Mengen⁸) und schneiden die Suche ab, wann immer wir eine Menge erreichen, die unterhalb der minimalen Häufigkeit liegt.

In einer zweiten Phase müssen dann aus den häufigen Artikelmengen Assoziationsregeln abgeleitet werden (Abbildung 12.13).

Auch dabei können wir wieder die entsprechenden Ordnungseigenschaften ausnutzen. Wie in der Abbildung dargestellt, führt das Bewegen eines Artikels von der Prämisse in die Konklusion einer Regel (beide Regeln entstammen derselben Artikelmenge) dazu, dass die Konfidenz gleich bleibt oder sinkt. Bei der Verwandlung einer Artikelmenge in Assoziationsregeln können wir also mit allen einelementigen Konklusionen beginnen, und dann nach und nach weitere Elemente aus der Prämisse der Konklusion hinzufügen. Zur rechnerischen Auswertung der Konfidenz für eine bestimmte Regel kann übrigens die Tatsache ausgenutzt werden, dass gemäß den Definitionen von c und s : $c(X \rightarrow Y) = \frac{s(XY)}{s(X)}$. Wir können also zur Berechnung der Konfidenz einfach auf die gespeicherten Häufigkeiten aller häufigen Artikelmengen zugreifen⁹.

Es ergibt sich der in Abbildung 12.14 dargestellte APRIORI-Algorithmus [4], bei dem zunächst nur anzumerken ist, dass die notwendige Speicherung der Häufigkeiten $s(c)$ für alle häufigen Mengen c nicht explizit angegeben wurde.

Der Algorithmus enthält jedoch zwei Teilprozeduren, die einer genaueren Betrachtung bedürfen (Abbildung 12.15). Die Teilprozedur PRUNE prüft die Häufigkeit der Mengen und entfernt diejenigen, die die geforderte Minimalhäufigkeit nicht erreichen. Soll dies auf nicht-hauptspeicherresidenten Daten durchgeführt werden, so ist es günstig, statt einer Schleife, die für jede Kandidatenmenge alle Transaktionen prüft, eine Schleife zu verwenden, die für jede Transaktion alle Kandidatenmengen prüft. So müssen die externen Daten pro Ebene k des Verfahrens nur einmal durchlaufen werden. Um dabei die in einer Transaktion enthalten Kandidaten schnell finden zu können, können *Hashbäume* verwendet werden [4].

Die zweite Teilprozedur ERZEUGE-KANDIDATEN erzeugt aus den gefundenen k -elementigen häufigen Mengen alle möglichen $k + 1$ -elementigen Kandidaten für die nächste Ebene. Statt dabei einfach alle möglichen Artikel an jede bisherige häufige Menge anzuhängen, reicht es aus, alle solchen Paare bisheriger häufiger Mengen zu betrachten, die genau $k - 1$ Elemente gemeinsam haben, und diese zu einem dann $k + 1$ -elementigen Kandidaten zu vereinigen. Dies ist deshalb ausreichend, weil jede Teilmenge einer häufigen Teilmenge ebenfalls häufig sein muss. Gibt es also für eine $k + 1$ -elementige Menge kein erzeugendes Paar von häufigen Mengen der Größe k , so enthält die Menge offenbar Teilmengen der Größe k , die nicht häufig sind, und kann daher selbst auch nicht häufig sein. Da der Umkehrschluss nicht gilt, wird in der Erzeugungs-

⁸ Engl. *large* oder *frequent itemsets*.

⁹ Da alle Teilmengen einer häufigen Artikelmenge ebenfalls häufig sind, liegt $s(X)$ notwendigerweise ebenfalls vor.

```

proc APRIORI( $I, T, s_{\min}, c_{\min}$ )
   $L := \text{HÄUFIGE-MENGEN}(I, T, s_{\min})$ 
   $R := \text{REGELN}(L, c_{\min})$ 
  return  $R$ 

proc HÄUFIGE-MENGEN( $I, T, s_{\min}$ )
   $C_1 := \{\{i\} | i \in I\}, k := 1,$ 
   $L_1 := \text{PRUNE}(C_1, T)$ 
  while  $L_k \neq \emptyset$ 
     $C_{k+1} := \text{ERZEUGE-KANDIDATEN}(L_k)$ 
     $L_{k+1} := \text{PRUNE}(C_{k+1}, T)$ 
     $k := k + 1$ 
  return  $\bigcup_{j=2}^k L_j$ 

proc REGELN( $L, c_{\min}$ )
   $R := \emptyset$ 
  forall  $l \in L$  such that  $|l| \geq 2$ 
    let  $k := |l|, m := 1$ 
     $H_1 := \{\{i\} | i \in l\}$ 
    loop
      forall  $h \in H_m$ 
        if  $\frac{s(l)}{s(l \setminus h)} \geq c_{\min}$ 
          then add  $l \setminus h \rightarrow h$  to  $R$ 
          else  $H_m := H_m \setminus \{h\}$ 
        endfor
      while  $m \leq k - 2$  (i.e., else exit loop!)
         $H_{m+1} := \text{ERZEUGE-KANDIDATEN}(H_m)$ 
         $m := m + 1$ 
      endloop
  return  $R$ 

```

Abbildung 12.14: APRIORI-Algorithmus.

prozedur jede k -elementige Teilmenge eines Kandidaten noch einmal geprüft. Die Kandidatenmengen werden stets lexikografisch sortiert gehalten, um diese Prüfung effizient durchführen zu können. Die Erzeugungsprozedur kann auch bei der Regelgenerierung verwendet werden, denn wie in Abbildung 12.13 dargestellt, gilt für jede Regel mit ausreichender Konfidenz, dass jede Teilmenge ihrer Konklusion ebenfalls zu einer Regel mit ausreichender Konfidenz geführt haben muss (wobei die übrigen Elemente in die Prämissen bewegt werden).

Die Durchführung des Apriori-Algorithmus wird in Abbildung 12.16 anhand eines Beispiels illustriert. In dieser Abbildung sind links die Transaktionen T sowie s_{\min} und c_{\min} angegeben. In der Mitte finden sich zeilenweise die jeweiligen Kandidatenmengen C_k , ihre Häufigkeiten s und die sich daraus ergebenen häufigen Mengen L_k . Auf der Basis der häufigen Mengen wird im Kasten rechts die Regelerzeugung durchgeführt. Für jede häufige Menge sind die Mengen der ausgewählten Konklusionen A_m angegeben, bei den zweielementigen Mengen ergibt sich

```

proc ERZEUGE-KANDIDATEN( $L_k$ )
   $L_{k+1} := \emptyset$ 
  forall  $l_1, l_2 \in L_k$  so dass
     $l_1 = \{i_1, \dots, i_{k-1}, i_k\}$ 
     $l_2 = \{i_1, \dots, i_{k-1}, i'_k\}$ 
     $i'_k < i_k$  (lexikografische Ordnung)
    let  $l := \{i_1, \dots, i_{k-1}, i_k, i'_k\}$ 
    if alle  $k$ -elementigen Teilmengen von  $l$  sind in  $L_k$ 
    then  $L_{k+1} := L_{k+1} \cup \{l\}$ 
  return  $L_{k+1}$ 

proc PRUNE( $C, T$ )
  forall  $c \in C$  :  $s(c) := 0$ 
  forall  $t \in T$ 
    forall  $c \in C, c \subseteq t$  :  $s(c) := s(c) + 1$ 
  return  $\{c \in C \mid s(c) \geq s_{min} \cdot |T|\}$ 

```

Abbildung 12.15: APRIORI-Subroutinen.

ohnehin nur eine Ebene der Suche, bei der dreielementigen Menge $\{1, 2, 3\}$ ist die zweite Ebene leer, weil keine Regel der ersten Ebene eine Lösung war.

Die Performanz des beschriebenen Algorithmus hängt entscheidend von den Eigenschaften der tatsächlich gegebenen Transaktionsmenge T ab. Im allgemeinen Fall können von exponentiell vielen Teilmengen der Artikelmenge I tatsächlich alle häufig sein, mit entsprechenden Folgen für die Laufzeit des Verfahrens. Tatsächlich kauft aber ein Kunde von den 100.000 Artikeln eines Supermarktes nur recht wenige, so dass auch die Größe der häufigen Mengen sehr klein ist und ihre Anzahl mit wachsendem k sehr rasch abnimmt. Sind diese Annahmen gegeben, weist der Algorithmus eine ungefähr lineare Laufzeit in der Größe der Transaktionsmenge T auf. Ebenfalls sehr wichtig für die Gesamlaufzeit ist dabei aber natürlich die gewählte minimale Häufigkeit. Senkt man die erforderliche minimale Häufigkeit auf die Hälfte, so führt dies in Simulationsexperimenten zu einer Verdoppelung der Laufzeit. Die gewählte minimale Konfidenz ist demgegenüber weniger wichtig, da sie nur in der Regelgenerierungsphase benutzt wird, in der ohnehin nur die wenigen gefundenen häufigen Artikelmengen verwendet werden.

12.8.2 Erweiterungen

Auf der Basis des hier beschriebenen Apriori-Algorithmus sind verschiedene Erweiterungen entwickelt worden, die sich einerseits der Verbesserung der Laufzeit des Algorithmus auf großen Datenbeständen widmen, und andererseits den Algorithmus auf weitergehende Aufgabenstellungen erweitern.

Transaktionslisten. Aufgrund der Tatsache, dass jede Transaktion nur wenige der möglichen Artikel enthält, nimmt die Größe der Mengen C_k und L_k mit wachsendem k sehr rasch ab. Dies bedeutet, dass es auch eine wachsende Anzahl von Transaktionen gibt, die keine noch aktuellen Kandidatenmengen mehr enthalten, aber dennoch bei jedem Durchlauf durch die Daten betrachtet werden. Dies kann man verhindern, indem man statt der Transaktion selbst die Liste der in dieser Transaktion noch enthaltenen Kandidatenmengen verwendet, und alle leer gewor-

T:	C ₁ : 1 2 3 4 5 6 7 8 9 S: 5 5 6 2 2 1 1 2 2 L ₁ : 1 2 3	C ₂ : 12 13 23 S: 4 4 4 L ₂ : 12 13 23	C ₃ : 123 S: 3 L ₃ : 123	Rule Generation: 12: H ₁ ={1}{2} c(1→2)=s(12)/s(1)=4/5=0.8 c(2→1)=s(12)/s(2)=4/5=0.8 13: H ₁ ={1}{3} c(1→3)=s(13)/s(1)=4/5=0.8 c(3→1)=s(13)/s(3)=4/6=0.66 23: H ₁ ={2}{3} c(2→3)=s(23)/s(2)=4/5=0.8 c(3→2)=s(23)/s(3)=4/6=0.66 123: H ₁ ={1}{2}{3} c(12→3)=s(123)/s(12)=3/4=0.75 c(13→2)=s(123)/s(13)=3/4=0.75 c(23→1)=s(123)/s(23)=3/4=0.75 H ₂ =∅
c _{min} =0.8 s _{min} =3/8	Result: 1→2 2→1 1→3 2→3			

Abbildung 12.16: Beispieldurchführung von A Priori (T , s_{min} und c_{min} siehe linker Kasten).

denen Listen aus dem Datenbestand entfernt. Für $k = 1$ wäre eine solche Listendarstellung eine 1-zu-1-Kopie der Transaktionsliste, so dass man üblicherweise mit dem Wechsel zur Listendarstellung wartet, bis man davon ausgehen kann, dass diese Darstellung als ganzes in den Hauptspeicher passen wird [4].

Verallgemeinerte Assoziationsregeln. Bei der Beschreibung des Assoziationsregelverfahrens haben wir bisher die Frage nach der Wahl der richtigen Abstraktionsebene nur implizit beantwortet. Indem wir als Artikelmenge $I = \{Bier, Pizza, Chips, \dots\}$ angegeben haben, haben wir eine recht abstrakte Ebene gewählt, denn hinter jedem der genannten „Artikel“ verbergen sich eigentlich Hunderte von Artikeln unterschiedlicher Hersteller, Produkttypen, Verpackungsarten und -größen. Da es für einen Benutzer nur schwer vorher abzuschätzen ist, ob die interessanten Assoziationen auf der Ebene der Artikelnummer, der Ebene der Hersteller, der Ebene der Produktgruppe oder zwischen diesen Ebenen liegen, sind Verfahren wünschenswert, die Artikelhierarchien (und -mehrachthierarchien) verarbeiten und Assoziationsregeln auf und zwischen verschiedenen Ebenen dieser Hierarchien finden können. Für solche *verallgemeinerten Assoziationsregeln* gibt es entsprechende Verfahren, die die Eigenschaften der Hierarchie ausnutzen, um ihre Suche möglichst effizient zu gestalten [86].

Sequentielle Muster. In vielen potenziellen Anwendungsbereichen von Assoziationsregelverfahren ist eine Transaktion nicht so klar definiert wie an der Kasse eines Supermarktes. So kann es bei einem online-Angebot im Internet beispielsweise sein, dass ein Benutzer bei einem Besuch des Angebotes nur eine bestimmte Sache nachfragt, jedoch in mehr oder weniger kurzen Abständen zurückkehrt, um weitere Angebote abzufragen. Zwar könnte man nun festlegen, dass alle innerhalb einer bestimmten Zeitspanne genutzten Angebote zu einer Transaktion gehören, oder dass eine Pause ab einer gewissen Länge den Beginn einer neuen Transaktion markiert. In Verfahren zur Entdeckung von sogenannten *sequentiellen Mustern* kann der Benutzer üblicherweise einen maximalen Abstand zwischen den einzelnen Elementen eines sequentiellen Musters festlegen, und zusätzlich angeben, dass alle Artikel innerhalb eines gleitenden Zeitfensters von vorbestimmter Länge so behandelt werden können, als ob sie in einer Transaktion aufgetreten wären. Ein sequentielles Muster ist eine geordnete Liste von Artikelmengen. Ob ein sequentiell-

Ist Muster in einer Serie von Transaktionen eines Benutzers vorkommt, wird dann nicht mehr durch einen einfachen Teilmengentest festgestellt, sondern man verlangt, dass zwischen dem frühesten Element einer Artikelmenge in der Liste und dem letzten Element dieser Artikelmenge nicht mehr als die für das gleitende Zeitfenster festgelegte Zeit liegt, dass zwischen zwei aufeinanderfolgenden Artikelmengen in der Liste nicht mehr als der festgelegte Maximalabstand liegt, und dass die Liste eine Subsequenz der tatsächlich aufgetretenen Artikelsequenz ist. Auch für dieses Problem sind effiziente Algorithmen entwickelt worden (siehe z.B. [87]).

12.9 Subgruppenentdeckung

Wir sind im vergangenen Abschnitt über Assoziationsregelverfahren bereits kurz auf die Besonderheiten von *deskriptiv* orientierten Lernaufgaben im Gegensatz zu *prädiktiv* orientierten Lernaufgaben (Funktionslernen aus Beispielen) eingegangen. Bei prädiktiven Lernaufgaben ist es das Ziel, eine unbekannte Funktion möglichst gut zu approximieren, um so für alle möglichen zukünftigen Instanzen aus dem Instanzenraum den Funktionswert möglichst gut vorhersagen zu können. Hierzu ist also ein *globales* Modell erforderlich, das es gestattet, für jede mögliche Instanz auch tatsächlich eine Vorhersage zu machen. Hauptziel erfolgreichen Lernens ist das Finden eines Modells mit minimalem wahren Fehler. Beim deskriptiven Lernen hat man andere Ziele. Hier ist man daran interessiert, durch Hypothesen beschriebene Teilbereiche des Instanzenraums zu identifizieren, über die *lokal* interessante Aussagen gemacht werden können.

Betrachten wir zur besseren Illustration ein Anwendungsbeispiel des deskriptiven Lernens. Unser Beispiel soll ein größeres Unternehmen sein, das seinen Kunden mehrere verschiedene Produkte anbietet und eine Kundendatenbank besitzt, in der für jeden Kunden verzeichnet ist, welche Produkte des Unternehmens bereits genutzt werden. Das Unternehmen könnte z. B. eine Versicherung sein, die ihren Kunden Hausrats-, Haftpflicht-, und Lebensversicherungen anbietet. Um sich weitere Marktchancen zu erschließen, wird ein solches Unternehmen daran interessiert sein, Teilbereiche des Instanzenraums, also Gruppen von Kunden, zu identifizieren, in denen beispielsweise ein bestimmtes Produkt auffällig unterrepräsentiert ist. So könnte ein deskriptives Lernverfahren folgende lokale Beobachtungen entdecken:

„Unter den alleinstehenden jungen Männern in ländlichen Regionen ist der Anteil der Lebensversicherungskunden signifikant niedriger als im gesamten Kundenbestand.“

Oder

„Verheiratete Männer mit Pkws der Luxusklasse machen nur zwei Prozent der Kunden aus, erzeugen aber vierzehn Prozent der Lebensversicherungsabschlusssumme.“

Obwohl mit solchen Aussagen keine globale Vorhersage darüber gemacht werden kann, welcher Kunde nun tatsächlich eine Lebensversicherung kaufen wird, sind sie als lokal beschreibende Aussagen dennoch möglicherweise eine wichtige Basis, z. B. für die Planung von Geschäftsstrategien. Mehr noch, in vielen Anwendungen lassen sich solche interessanten lokalen Aussagen finden, obwohl ein globales prädiktives Modell (z. B. aufgrund nicht genügend reichhaltiger Beschreibung der Instanzen) nicht oder nur mit sehr schlechtem Erfolg erlernt werden kann.

Lernaufgaben wie die gerade beschriebenen werden üblicherweise als *Subgruppenentdeckung* [55] bezeichnet, da man, in Anlehnung an die in der Statistik übliche Terminologie, den Instanzenraum als (Repräsentation einer) *Population* bezeichnen kann, so dass ein beschriebener Teilbereich des Instanzenraum dann als Subgruppe dieser Population aufgefasst wird. Wir können die Lernaufgabe wie folgt präzisieren.

Definition 12.9.1 (Subgruppenentdeckung). *Sei X ein Instanzenraum mit einer Wahrscheinlichkeitsverteilung D und L_H ein Hypothesenraum, in dem jede Hypothese als Extension eine Teilmenge von X hat:*

$$\text{ext}(h) \subseteq X \text{ für alle } h \in L_H.$$

Sei weiterhin

$$S \subseteq X$$

eine gegebene, gemäß D gezogene Stichprobe der Gesamtpopulation.

Es sei schließlich q eine Funktion

$$q := L_H \rightarrow \mathbb{R}.$$

Die Lernaufgabe Subgruppenentdeckung kann dann auf zwei Arten definiert werden:

1. Gegeben X, S, L_H, q und eine Zahl $q_{min} \in \mathbb{R}$, finde alle $h \in L_H$, für die $q(h) \geq q_{min}$. Oder/Und
2. gegeben X, S, L_H, q und eine natürliche Zahl $k \geq 1$, finde eine Menge $H \subseteq L_H, |H| = k$ und es gibt keine $h \in H, h' \in L_H \setminus H : q(h') \geq q(h)$.

Eine zentrale Rolle in dieser Definition spielt die Qualitätsfunktion q , die bewertet, wie „interessant“ eine bestimmte Hypothese bzw. die durch sie repräsentierte Subgruppe $\text{ext}(h)$ ist. Gegeben q verlangen wir bei der Subgruppenentdeckung entweder alle Subgruppen oberhalb einer bestimmten Mindestqualität q_{min} , oder/und die k gemäß q besten Hypothesen aus unserem Hypothesenraum.

12.9.1 Qualitätsfunktionen

Wie kann nun die Funktion q aussehen, welche die Interessanz von Aussagen messen soll? Um die tatsächliche Interessanz einer Beobachtung für einen Benutzer zu messen, wäre eigentlich eine umfassende Benutzermodellierung notwendig, die Ziele, Interesse und Vorwissen des Benutzers erfasst. So sind beispielsweise bereits bekannte Aussagen für die meisten Benutzer nicht besonders interessant. Auch wenn es erste Arbeiten zur Entwicklung von präzisen benutzerbezogenen Interessanzmodellen gibt, so ist es momentan doch noch allgemein üblich, die Interessanz von Aussagen anhand leicht objektivierbarer statistischer Eigenschaften der betrachteten Subgruppe zu messen [54]. Üblicherweise werden dabei zwei unterschiedliche Gesichtspunkte einbezogen. Wie in den obigen Beispieldaten schon angedeutet, interessiert

man sich zunächst einmal für Subgruppen, die statistisch auffällig sind, weil beispielsweise die Verteilung eines bestimmten Merkmals von Interesse („abgeschlossene Lebensversicherungen“) sich „signifikant“ von der Verteilung in der gesamten Stichprobe bzw. der gesamten Population unterscheidet. Als zweiter Gesichtspunkt muss aber die Größe der betrachteten Subgruppe einbezogen werden, denn auch eine bemerkenswerte statistische Auffälligkeit ist nur dann interessant, wenn sie mehr als einige wenige Objekte aus der Population betrifft.

Eine naheliegende Möglichkeit, den Begriff „statistisch auffällig“ zu präzisieren, ist der Rückgriff auf das Instrumentarium statistischer Hypothesentests (siehe z.B. [26]). Betrachten wir dazu die einfachste Variante eines Subgruppenentdeckungsproblems und nehmen an, dass wir an Subgruppen interessiert sind, die hinsichtlich der Verteilung eines binären Attributs von Interesse auffällig sind. Im ersten Beispiel dieses Abschnitts war dies das Attribut „Lebensversicherung abgeschlossen?“. Wir können uns nun ein Experiment vorstellen, bei dem wir (gemäß der Verteilung D) zufällig ein Objekt aus der Gesamtpopulation ziehen und prüfen, ob die gesuchte Eigenschaft bei diesem Objekt vorliegt oder nicht (ob also der Kunde eine Lebensversicherung abgeschlossen hat oder nicht). Die Verteilung der gesuchten Eigenschaft in der Gesamtpopulation entspricht einer Wahrscheinlichkeit p , bei einer Ausführung unseres Experiments ein Objekt mit der gewünschten Eigenschaft zu ziehen.

Um nun die statistische Auffälligkeit einer durch eine Hypothese h beschriebenen Subgruppe der Größe $n := |\text{ext}(h)|$ zu bewerten, betrachten wir die Wahrscheinlichkeit p_h , bei Ziehung eines Objektes aus dieser Subgruppe ein Objekt mit der gesuchten Eigenschaft zu erhalten. Als statistisch hinsichtlich der gesuchten Eigenschaft nicht auffällig wollen wir nun jede Subgruppe h betrachten, die sich hinsichtlich des gesuchten Attributs genau so verhält, wie eine zufällig aus der Gesamtpopulation ausgewählte Stichprobe der Größe n . Dies ist unsere durch einen geeigneten statistischen Test möglicherweise zu verwurfende Nullhypothese über die Subgruppe h . Gilt die Nullhypothese, so ist $p_h = p$, und das Ziehen einer Subgruppe der Größe n ist die n -fache Wiederholung eines Experiments, bei dem mit der Wahrscheinlichkeit p ein Objekt mit der gesuchten Eigenschaft gezogen wird. Die relative Häufigkeit solcher Objekte in dieser Stichprobe, also in $\text{ext}(h)$, ist dann also ein geeigneter Schätzer für p :

$$\hat{p} := \frac{|\{s \in \text{ext}(h) | A(s) = 1\}|}{n}$$

Dieser Schätzer ist *erwartungstreu*, d. h., wenn wir wiederholt Stichproben der Größe n ziehen, wird der Wert dieses Schätzers gemittelt über alle Stichproben schließlich der zu Grunde liegenden Wahrscheinlichkeit entsprechen, also unter der Annahme der Nullhypothese, dass $p_h = p$:

$$E(\hat{p}) = p.$$

In den einzelnen Stichproben der Größe n schwankt der Wert unseres Schätzers je nach den für die jeweilige Stichprobe ausgewählten Objekte. Was können wir aus dem in einer einzelnen Stichprobe der Größe n , nämlich unserer Subgruppe h , beobachteten Wert von \hat{p} über die Gültigkeit der Nullhypothese schließen? Betrachten wir dazu die Wahrscheinlichkeit, mit der ein bestimmter Wert von \hat{p} auftritt, falls p tatsächlich die zu Grunde liegende Wahrscheinlichkeit der gesuchten Eigenschaft ist. Je unwahrscheinlicher ein beobachteter Wert ist, desto weniger plausibel ist es, dass tatsächlich p die zugrundeliegende Wahrscheinlichkeit war, und desto naheliegender ist es, die Nullhypothese zu verwerfen, und anzunehmen, dass die betrachtete Subgruppe sich tatsächlich von der Gesamtpopulation statistisch unterscheidet. Betrachten wir

als Beispiel eine Population, in der ein gesuchtes Merkmal mit der Wahrscheinlichkeit $p = 0,5$ auftritt. Bei Ziehung einer Subgruppe der Größe $n = 2$ erwarten wir im Mittel ein Objekt mit der gesuchten Eigenschaft; die Wahrscheinlichkeit, zwei Objekte mit dieser Eigenschaft anzu treffen, ist jedoch immer noch $P(\hat{p} = 1|p = 0,5, n = 2) = 0,5 \cdot 0,5 = 0,25$.

Wenn wir in einer Subgruppe der Größe 2 diese Verteilung beobachten, ist dies also kein besonders starker Hinweis darauf, dass die Subgruppe sich von der Gesamtpopulation unterscheidet. Die sich für allgemeines n ergebene Wahrscheinlichkeitsverteilung $P(\hat{p} = x|p, n)$ der Werte von \hat{p} (genauer die Verteilung der Anzahl der gezogenen Objekte mit der gesuchten Eigenschaft) ist die *Binomialverteilung*. Für genügend große Werte von n ($n > 30$ wird allgemein akzeptiert) wird diese Verteilung genügend genau durch eine Normalverteilung mit Mittelwert $\mu = p$ und Standardabweichung $\sigma = \sqrt{\frac{p \cdot (1-p)}{n}}$ genähert.

Man könnte nun direkt die Wahrscheinlichkeit $P(\hat{p} = x|p, n)$ eines beobachteten Wertes x für \hat{p} als Maß für die Plausibilität einer Hypothese nehmen, müsste allerdings dann für verschiedene p und n unterschiedliche Verteilungstabellen nutzen. Man führt daher eine so genannte z -Transformation durch, die aus einer beliebigen normalverteilten Variablen eine standardnormalverteilte Variable ($\mu = 0$ und $\sigma = 1$) erzeugt. Dies ergibt hier die V genannte Variable

$$V := \frac{\hat{p} - p}{\sqrt{\frac{p \cdot (1-p)}{n}}}.$$

Der V -Wert hat auch die intuitiv günstige Eigenschaft, dass Abweichungen als Vielfaches der Standardabweichung ausgedrückt werden, so dass die für kleinere n stärker schwankenden Werte von \hat{p} günstiger skaliert sind. Als Maß für die Plausibilität der Nullhypothese wählen wir dann $P(|V| \geq v)$, also die Wahrscheinlichkeit, dass die skalierte Abweichung unseres Schätzers von p mindestens v beträgt¹⁰. Die entsprechenden Werte können den Tabellen in Statistikbüchern entnommen werden, so ist z. B. die Wahrscheinlichkeit, eine skalierte Abweichung von mehr als 4,9 zu beobachten nur $\frac{1}{1000000}$. Beobachtet man einen solchen Wert von V bei einer Subgruppe, so ist das also ein sehr starkes Indiz dafür, dass die Nullhypothese nicht gilt, und dass also die Subgruppe eine andere Verteilung des gesuchten Merkmals aufweist als die Gesamtpopulation.

Für die Definition einer entsprechenden Qualitätsfunktion q können wir noch einige Vereinfachungen vornehmen, denn wir sind in der Regel ja nur an der Reihenfolge interessiert, die q den Hypothesen zuweist und nicht am konkreten absoluten Wert. Da $P(|V| \geq v)$ für wachsende v monoton fallend ist, und wir ohnehin kleineren Wahrscheinlichkeiten eine höhere Qualität (Auffälligkeit) zuordnen wollen, führt die Verwendung des Wertes von V statt der Wahrscheinlichkeit zu einer wie erwünscht genau umgekehrten Reihenfolge der Hypothesen. Da die Bezugswahrscheinlichkeit p der Gesamtpopulation für alle zu bewertenden Subgruppen gleich ist, können wir wiederum ohne Änderung der Reihenfolge wie folgt umformen:

$$|V| := \frac{|\hat{p} - p|}{\sqrt{\frac{p \cdot (1-p)}{n}}} = \frac{\sqrt{n} \cdot |\hat{p} - p|}{\sqrt{p \cdot (1-p)}} \sim \sqrt{n} \cdot |\hat{p} - p| \sim \sqrt{g} \cdot |\hat{p} - p|.$$

Im letzten Schritt dieser Umformung ist die Größe der Subgruppe n ersetzt worden durch ihre relative Größe $g := \frac{n}{|S|}$. Um diese Qualitätsfunktion evaluieren zu können, muss natürlich

¹⁰ Dies ist der zweiseitige Test. Sind nur Abweichung in eine Richtung von Interesse, so entfallen die Betragsstriche.

die Wahrscheinlichkeit p für die Gesamtpopulation geschätzt werden. Dies tut man in nahe liegender Weise durch die relative Häufigkeit der gesuchten Eigenschaft in S :

$$\hat{p}' := \frac{|\{s \in S | A(s) = 1\}|}{|S|}.$$

Wir erhalten also die Qualitätsfunktion¹¹

$$q(h) := \sqrt{g} \cdot |\hat{p} - \hat{p}'|.$$

Die Qualitätsfunktion q , so wie wir sie gerade definiert haben, ordnet also die Subgruppen nach absteigender Signifikanz an. Die besten k Subgruppen dieser Liste müssen also keineswegs Subgruppen sein, bei denen man die Nullhypothese mit genügender Sicherheit ausschließen kann. Um dies sicherzustellen, müssen wir ein Mindestqualitätsniveau q_m in vorgeben, das sich gemäß den obigen Umformungen aus dem gewünschten Signifikanzniveau des statistischen Hypothesentests ergibt. Dabei tritt jedoch noch eine weitere Schwierigkeit auf. Wenn wir bei jeder einzelnen Hypothese mit 95%-iger Sicherheit ausschließen können, dass die betrachtete Subgruppe fehlerhafterweise als auffällig erkannt wurde, so bedeutet das immer noch, dass man bei 5% aller getesteten Subgruppen eine fehlerhaft erkannte Auffälligkeit erwarten muss. Dieses Problem kann dadurch entschärft werden, dass man das Signifikanzniveau für den einzelnen Test entsprechend der Größe des Hypothesenraums verschärft (Bonferroni-Anpassung).

12.9.2 Effiziente Suche

Verfahren zur Entdeckung von Subgruppen müssen gemäß der Definition der Lernaufgabe nicht nur eine Lösung finden, sondern die Menge aller Lösungen oberhalb eines bestimmten Qualitätsniveaus bzw. die Menge der k besten Lösungen im Hypothesenraum. Die Verfahren können deshalb auch keine Hill-climbing-Suche durchführen, sondern müssen versuchen, den Hypothesenraum möglichst vollständig zu durchsuchen. Dies kann im einfachsten Fall z. B. mit einer absteigenden Breitensuche in L_H geschehen (Abbildung 12.17). Man beginnt also mit der allgemeinsten Hypothese in diesem Raum, und verfeinert diese dann zu immer kleineren Subgruppen. Schon bei einfachen Problemen, bei denen die Instanzen nur durch wenige Merkmale beschrieben sind, ergibt sich jedoch ein so großer Hypothesen- und damit Suchraum, dass es notwendig ist, Kriterien zu finden, mit denen möglichst große Teile des Suchraumes abgeschnitten und damit von der Betrachtung ausgeschlossen werden können.

Am leichtesten ist dies möglich, wenn man für die Problemstellung relevante Eigenschaften der Hypothesen finden kann, die sich entlang eines absteigenden Suchpfades monoton entwickeln, also z. B. niemals größer, sondern nur kleiner werden können. Sobald bei einer absteigenden Suche diese Größe einen geforderten Mindestwert unterschreitet, können alle darunterliegenden Hypothesen als Lösung ausgeschlossen werden. Für die bei der Subgruppenentdeckung relevante Eigenschaft, die Qualität q , gilt leider eine solche Monotonieeigenschaft im allgemeinen Fall nicht. Je nachdem, wie sich die Verteilungen bei einer Verkleinerung der Subgruppe ändern, kann die Qualität der Subgruppe fallen oder auch steigen.

Für die zweite Variante des Subgruppenentdeckungsproblems, das Finden der k besten Subgruppen, kann man dennoch erfreulicherweise in vielen Fällen große Teile des Suchraumes

¹¹ In Artikeln über Subgruppenentdeckung ist es üblich, statt \hat{p}' p_0 zu verwenden, und für \hat{p} einfach p .

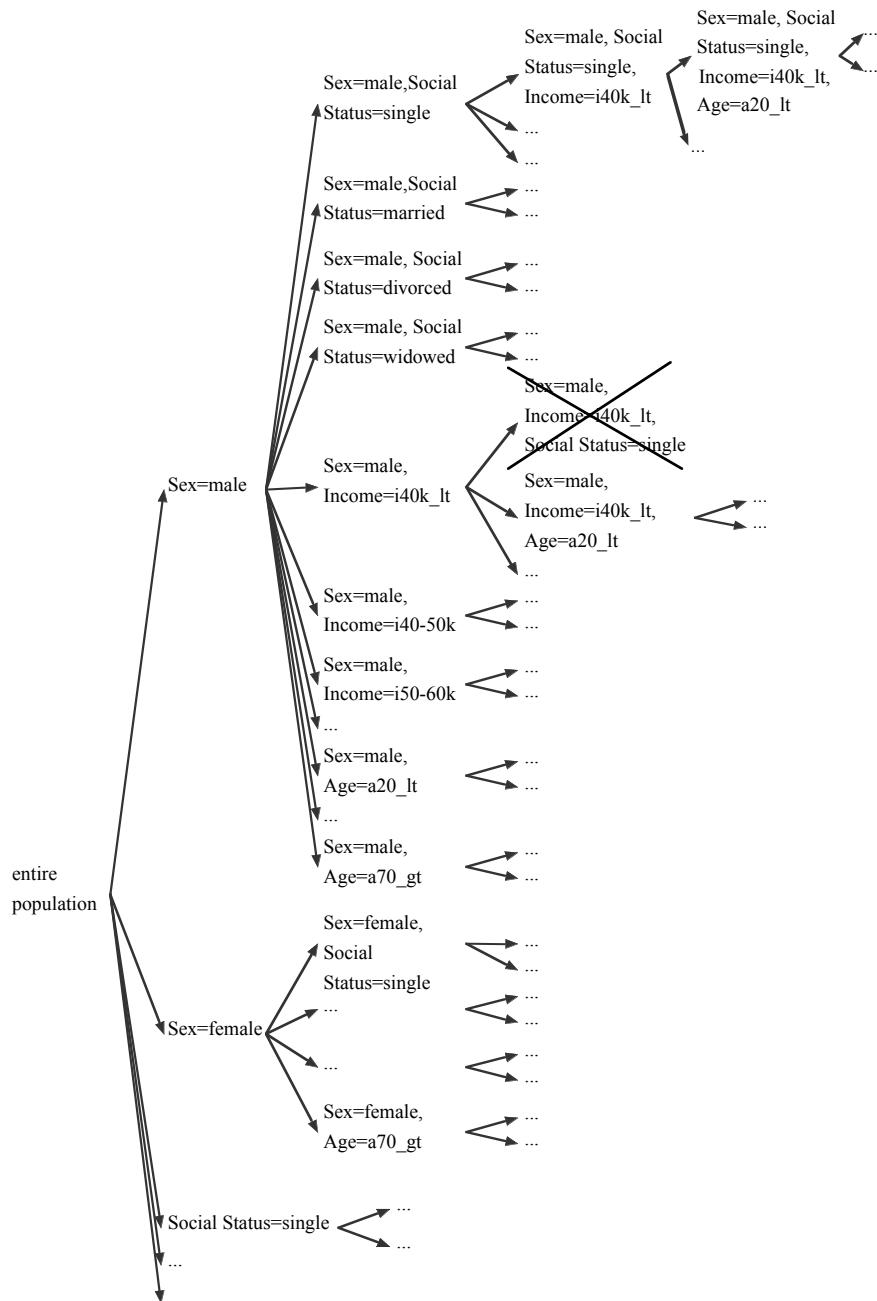


Abbildung 12.17: Suchbaum bei der propositionalen Subgruppensuche.

```

proc SUBGRUPPEN( $S, \rho, q, q_{min}, g_{min}, k$ )
   $Q := \{, \text{Ganze Population}\}, H := \emptyset$ 
  while  $Q \neq \emptyset$ 
    Wählle  $C \subseteq Q$  (gemäß Suchstrategie)
     $\rho(C) := \bigcup_{h \in C} \rho(h)$ 
    forall  $h \in \rho(C)$ 
      Teste  $h$  auf  $S$  (berechne  $q(h)$ )
      if  $q(h) \geq q_{min}$  und  $g(h) \geq g_{min}$  then
        if  $q(h) > \min_{h \in H} q(h)$  then
          if  $|H| < k$ 
            then  $H := H \cup \{h\}$ 
          else ersetze schlechtestes Element von  $H$  durch  $h$ 
        if  $q_{max}(h) < \min_{h \in H} q(h)$  or  $g(h) < g_{min}$ 
          then (Teilbaum wird abgeschnitten)
        else  $Q := Q \cup \{h\}$ 
    return  $H$ 

```

Abbildung 12.18: Subgruppensuche mit optimistischen Schätzfunktionen.

abschneiden, indem man sich sogenannter *optimistischer Schätzfunktionen* für q bedient [96]. Das Ziel einer optimistischen Schätzfunktion ist es, gegeben eine bestimmte Hypothese h und ihre Qualität $q(h)$, abzuschätzen, wie sich die Qualitätsfunktion q bei einer weiteren absteigenden Suche ausgehend von h günstigstenfalls entwickeln könnte. Die optimistische Schätzfunktion sollte garantieren, dass für keine der unterhalb von h liegenden spezielleren Hypothesen der Wert von q oberhalb der optimistischen Schätzung liegt. Können wir dies zusichern, so kann man immer dann den Teilraum unterhalb einer Hypothese h abschneiden, wenn die optimistische Schätzung für diesen Teilraum unterhalb der Qualität der schlechtesten der bisher gefundenen k Hypothesen liegt.

Wir können dies wie folgt präzisieren. Sei $\rho : L_H \rightarrow 2^{L_H}$ der Spezialisierungsoperator, der uns bei der absteigenden Suche zu einer Hypothese h alle unmittelbar spezielleren Nachfolger $\rho(h)$ liefert, und sei ρ^* die transitive Hülle von ρ , also der gesamte Teilraum unterhalb einer Hypothese h . Eine optimistische Schätzfunktion für q ist dann eine Funktion:

$$q_{max} : L_H \rightarrow \mathbb{R}, \text{ so dass } \forall h \in L_H, \forall h' \in \rho(h) \quad q(h') \leq q_{max}(h)$$

Mit Hilfe einer solchen optimistischen Schätzfunktion kann ein Suchgruppenentdeckungsalgorithmus wie in Abbildung 12.18 formuliert werden. In diesem Algorithmus wird optional noch eine vom Benutzer angegebene minimale Größe der zu entdeckenden Subgruppen zum Beschneiden des Suchraumes genutzt.

Je nach Beschaffenheit der Qualitätsfunktion q ist es unterschiedlich schwierig, eine optimistische Schätzfunktion anzugeben, bzw. nachzuweisen, dass sie überhaupt existiert. Wir wollen im folgenden eine optimistische Schätzfunktion für den einfachsten Fall, der Suche nach Subgruppen in Bezug auf ein binäres Zielattribut, angeben, und zwar für die oben erläuterte Qualitätsfunktion

$$q(h) = \sqrt{g} \cdot |\hat{p} - \hat{p}'|.$$

Da g , die Größe der Subgruppe, bei der Spezialisierung nur kleiner werden kann, können wir den ersten Faktor einfach optimistisch mit dem gleichen Wert wie in der aktuellen Hypothese

h schätzen. Im zweiten Faktor kann sich \hat{p} im Intervall $[0, 1]$ sowohl nach oben als auch nach unten bewegen. Eine erste grobe optimistische Schätzfunktion, mit der wir uns an dieser Stelle begnügen wollen, ist deshalb

$$q_{max}(h) := \sqrt{g} \cdot \max(\hat{p}', 1 - \hat{p}').$$

12.9.3 Assoziationsregeln vs. Subgruppen

Es sei zum Schluss noch darauf hingewiesen, dass man auch das Entdecken von Assoziationsregeln als Instanz der Lernaufgabe Subgruppenentdeckung betrachten kann. Jede Assoziationsregel repräsentiert die Subgruppe all derjenigen Transaktionen, in denen alle Artikel der Assoziationsregel vorkommen. Man kann dann eine Qualitätsfunktion q definieren, die für eine Assoziationsregel r den Wert 1 hat genau dann, wenn $s(r) \geq s_{min}$, $c(r) \geq c_{min}$, und ansonsten den Wert 0. Auffälligkeit wird also nicht durch Vergleich mit einer Referenzgruppe definiert, sondern einfach durch einen minimalen Prozentsatz gemeinsamen Auftretens. Die minimale Häufigkeit kommt einfach als zweite Minimalbedingung hinzu und wird nicht mit der Auffälligkeit verrechnet, und es findet keine Differenzierung unterschiedlicher Interessantheitsgrade statt.

Auch wenn man also formal die Assoziationsregeln als Subgruppenentdeckung sehen kann, so ist es aus den gerade genannten Gründen technisch nicht sinnvoll, dies zu tun. Die extrem simple Qualitätsfunktion und die im Abschnitt über Assoziationsregeln diskutierten besonderen Annahmen über die Zusammensetzung der Transaktionen ermöglichen bzw. erfordern ganz andere Suchverfahren. Insbesondere gelten bei den üblicherweise verwendeten mächtigeren Qualitätsfunktionen zur Subgruppenentdeckung die in Abbildung 12.12 gemachten Beobachtungen nicht, denn die Qualität kann bei absteigender Suche sowohl fallen als auch sinken, so dass q , wie oben beschrieben, nicht direkt zur Beschneidung des Suchraums genutzt werden kann.

12.9.4 Erweiterungen

Die Weiterentwicklung von Subgruppenverfahren erfolgt entlang mehrerer Dimensionen. Wie bereits erwähnt sind für unterschiedliche Anwendungszwecke vielfältige Qualitätsfunktionen entwickelt worden [54]. Weitere Entwicklungen betreffen die Nutzung mächtigerer Hypothesenräume, die Einführung zeitbezogener Subgruppenmuster, und die Optimierung von Subgruppenergebnissen.

Mächtigere Hypothesenräume. Die in diesem Abschnitt beschriebene Subgruppensuche kann selbstverständlich nicht nur für Hypothesenräume durchgeführt werden, die sich auf Merkmalsvektoren beziehen, sondern für jeden Hypothesenraum, in dem sich ein Verfeinerungsoperator ρ definieren lässt. So gibt es beispielsweise Subgruppenverfahren für multirelationale prädikatenlogische Hypothesenräume, bei denen durch geeignete Sprachbeschränkungen sichergestellt wird, dass der entstehende Suchraum handhabbar bleibt. Neben den im Abschnitt 12.7 diskutierten syntaktischen Schemata haben sich hier Einschränkungen bewährt, die an die Fremdschlüsselbeziehungen in Datenbanken angelehnt sind [96].

Zeitbezogene Subgruppenmuster. Bei vielen Marktuntersuchungen ist es üblich, die Untersuchungen in bestimmten Abständen zu wiederholen. So könnte beispielsweise ein Unternehmen eine jährliche Kundenbefragung durchführen und daran interessiert sein, die Gesamtheit

der Fragebogenrückläufe auch im zeitlichen Trend mit Subgruppenverfahren zu untersuchen. Es sind zu diesem Zweck spezielle Qualitätsfunktionen entwickelt worden, die nicht nur einen Vergleich von Subgruppen mit einer Gesamtpopulation durchführen, sondern dies auch über mehrere in zeitlichen Abständen erhobene Teilstichproben („Segmente“) tun können [53]. Dabei muss als technische Schwierigkeit beachtet werden, dass die Beschreibungen der Instanzen in den einzelnen Segmenten nicht notwendigerweise die gleichen Attribute aufweisen [52].

Optimierung von Subgruppenergebnissen. Formuliert man deskriptive Lernprobleme mit Qualitätsfunktionen, die auf die einzelnen Hypothesen bezogen sind, so ergibt sich zwangsläufig das Problem, dass die Gesamtmenge aller gefundenen Hypothesen möglicherweise in sich nicht gut aufeinander abgestimmt ist. So kann es bei der Subgruppenentdeckung ohne weitere Vorehrungen beispielsweise vorkommen, dass wir entdecken, dass alleinstehende Männer doppelt so häufig Lebensversicherungen abschließen wie die Gesamtpopulation, und dass wir gleichzeitig als Entdeckung zurückliefern, dass alleinstehende Männer in ländlichen Gegenden doppelt so häufig Lebensversicherungen abschließen wie die Gesamtbevölkerung. Allgemein ergibt sich also die Frage, wie inhaltlich ähnliche Subgruppen erkannt und dann die weniger interessante von zwei sehr ähnlichen Subgruppen unterdrückt werden kann. In unserem Beispiel ist vermutlich die zweite Beobachtung weniger interessant, weil ihr Inhalt vollständig aus der ersten Beobachtung ableitbar ist. Man kann zur Unterdrückung solch redundanter Subgruppen Maße definieren, die die sogenannte *Affinität* von Subgruppen berechnen, und kann dann Schwellenwerte festlegen, so dass oberhalb einer gewissen Affinität und unterhalb eines bestimmten Unterschieds in der Interessantheit die schwächere der beiden Subgruppen unterdrückt wird [33].

12.10 Clusteranalyse

Bei der Clusteranalyse [9, 88] geht es grob gesagt darum, eine gegebene Instanzenmenge S aus einem Instanzenraum X so in verschiedene Teilmengen („Cluster“) aufzuteilen, dass die Objekte innerhalb eines Clusters möglichst ähnlich sind, und dass Objekte verschiedener Cluster einander möglichst unähnlich sind. Diese Lernaufgabe findet immer dann Anwendung, wenn man statt einer großen Zahl einzelner Instanzen lieber mit einer überschaubar kleinen Anzahl idealerweise homogener Gruppen umgehen möchte. So könnte man beispielsweise die Kundendatenbank eines Unternehmens einer Clusteranalyse unterziehen, um homogene Kundengruppen zu finden, mit denen dann in der Angebotsgestaltung und im Marketing gearbeitet werden kann. Im Idealfall sollte ein Clusteringverfahren dabei nicht nur die gefundenen Teilmengen mit den Listen der zu ihnen gehörenden Instanzen liefern, sondern auch eine kompakte und möglichst leicht verständliche Beschreibung für jeden Cluster, die die in dem Cluster enthaltenen Objekte charakterisiert. So könnte eine Clusteranalyse einer Kundendatenbank beispielsweise die Cluster „auf dem Land lebende Menschen mit Kindern“, „Senioren mit hohem Einkommen“, „beamte Akademiker“ usw. ergeben. Man spricht in diesem Fall auch von *begrifflicher Clusteranalyse (conceptual clustering)* [62]. In den meisten Anwendungen verlangt man, dass die gefundenen Cluster sich nicht überlappen sollen, und dass jede mögliche Instanz des Instanzenraumes X einem Cluster zugeordnet werden kann, d.h., die Menge aller Cluster soll eine *Partitionierung* des Instanzenraumes darstellen.

Im Vergleich zur Subgruppenentdeckung ist hervorzuheben, dass die beim Clustern gefundenen Gruppen nicht hinsichtlich ihrer Eigenschaften in Bezug auf ausgewählte Zielmerkmale iden-

tifiziert werden, sondern ausschließlich durch die Forderung nach einer maximalen Ähnlichkeit der Instanzen innerhalb eines Clusters. Die Ähnlichkeits- bzw. Abstandsfunktion nimmt dementsprechend einen wichtigen Platz in der präzisen Definition der Lernaufgabe ein.

Definition 12.10.1 (Cluster-Analyse). *Sei X ein Instanzenraum und $S \subseteq X$ eine Menge von Instanzen. Sei weiterhin*

$$dist : X \times X \rightarrow \mathbb{R}^+$$

eine Abstandsfunktion, und

$$q : 2^{2^X} \rightarrow \mathbb{R}$$

eine Qualitätsfunktion. Gegeben S , $dist$ und q besteht die Aufgabe der Clusteranalyse darin, eine Clusterung

$$\mathcal{C} = \{C_1, \dots, C_k\}, \text{ wobei } C_i \subseteq S \text{ für alle } i = 1, \dots, k,$$

zu finden, so dass $q(\mathcal{C})$ maximiert wird, und (optional)

$$C_i \cap C_j = \emptyset \text{ für alle } i \neq j \in \{1, \dots, k\} \text{ und } \bigcup_{i=1, \dots, k} C_i = S \text{ (Partitionierung).}$$

Bezüglich der Ähnlichkeitsfunktionen können wir an dieser Stelle einfach auf die Ausführungen im Abschnitt 12.4 verweisen. Die Qualitätsfunktion q wird oft auf der Basis der Abstandsfunktion $dist$ definiert, indem man beispielsweise für jeden Cluster den durchschnittlichen Abstand aller Objektpaare in diesem Cluster berechnet, und die negierte Summe dieser Werte für alle Cluster als Qualitätsmaß nimmt. Ist es in einem Instanzenraum X möglich, das Zentrum eines Clusters zu bestimmen, so wird statt des durchschnittlichen paarweisen Abstandes auch der durchschnittliche (quadrierte) Abstand (Fehler) vom Zentrum verwendet (s.u. beim k -Means-Verfahren). Ist die Anzahl der gewünschten Cluster k nicht vom Benutzer vorgegeben, so ist es wichtig, dass die Qualitätsfunktion die Anzahl der Cluster und ihre innere Qualität gegeneinander abwägt. Mittels der Qualitätsfunktion kann auch die gewünschte Zuordnung neuer Instanzen aus X zu einem Cluster gefunden werden: man weist die Instanz einem Cluster so zu, dass danach die Gesamtqualität des Clusterings maximal wird.

12.10.1 Das k -Means-Verfahren

Ein sehr einfaches und populäres Verfahren zur Suche nach Clusterungen der Größe k für vorgegebenes k ist das k -Means-Verfahren [59]. Das k -Means-Verfahren setzt voraus, dass es möglich ist, im Instanzenraum X das *Zentrum* einer Menge von Instanzen zu berechnen, also denjenigen Punkt, der die Summe der Abstände zu allen Punkten dieser Menge minimiert (das Zentrum muss dabei nicht selbst in der Menge enthalten sein). Am einfachsten ist dies, wenn $X = \mathbb{R}^n$, also für rein numerische Instanzenräume. Das Zentrum einer Menge $C \subseteq X$ ist dann einfach das Mittel:

$$z(C) := \frac{1}{|C|} \sum_{\mathbf{c} \in C} \mathbf{c},$$

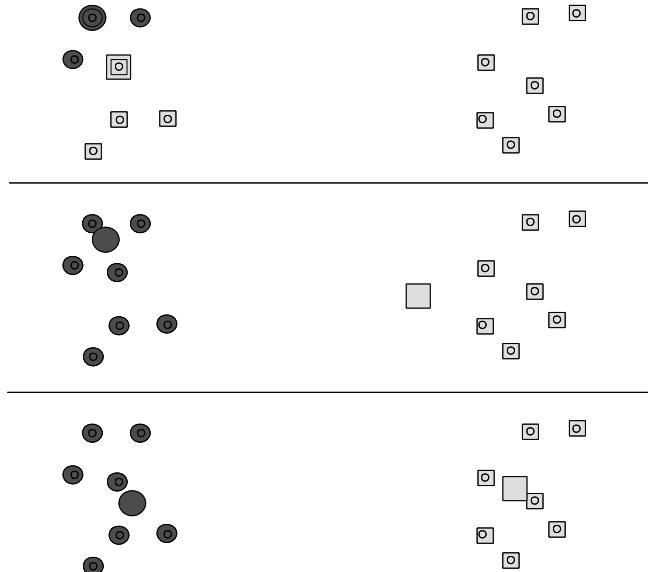


Abbildung 12.19: *k*-Means Beispiel (Oben: Anfangssituation, Mitte: nach einer Iteration, Unten: nach 2 Iterationen).

Der k -Means-Algorithmus arbeitet iterativ wie folgt:

```

proc K-MEANS( $S, k$ )
     $Z = \{z_1, \dots, z_k\} := k$  zufällig gewählte Punkte aus  $S$ 
    while Qualität wird besser
         $C_i := \{s \in S | i = \operatorname{argmin}_{i=1, \dots, k} \operatorname{dist}(s, z_i)\}$  für  $i = 1, \dots, k$ 
         $z_i := z(C_i)$  für  $i = 1, \dots, k$ 
    endwhile
    return  $\{C_1, \dots, C_k\}$ 

```

Es werden also iterativ immer wieder die Zentren der gerade aktuellen Cluster bestimmt, um dann die Cluster auf der Basis dieser Zentren neu zu bilden. Das Verfahren konvergiert sehr schnell, wie auch in Abbildung 12.19 zu sehen ist (die jeweils aktuellen Clusterzentren sind durch den grossen Kreis bzw. das grosse Viereck dargestellt).

Die relevante Qualitätsfunktion beruht hier auf dem durchschnittlichen Abstand zum Clusterzentrum, das Verfahren kann jedoch auch in lokalen Maxima dieser Qualitätsfunktion konvergieren. Da dies von der Menge der anfänglich gewählten Zentren abhängt, sind verschiedene Vorschläge zu deren besserer Initialisierung gemacht worden, bzw. man führt den Algorithmus mehrmals mit unterschiedlich gewählten Anfangszentren bis zur Konvergenz durch. Das Verfahren ist nicht in der Lage, Ausreißer in den Daten zu ignorieren, so dass diese auch als einzelne Punkte dann einen der k Cluster bilden. Eine entsprechende Datenvorverarbeitung zur Entfernung der Ausreißer ist also notwendig. Übliche Implementierungen des k -Means-Algorithmus führen zusätzlich durch wiederholte Läufe noch eine Suche nach dem günstigsten k innerhalb eines vom Benutzer vorgegebenen Bereiches aus.

Es sei schließlich noch angemerkt, dass das k -Means-Verfahren als Instanz eines allgemeinen, aus der Statistik bekannten iterativen Verfahrens zur Schätzung versteckter Parameter betrachtet werden kann, dem sogenannten *EM-Algorithmus*. Im allgemeinen Fall nimmt man dabei an, dass die Datenpunkte in S aus einer Mischung von k verschiedenen Verteilungen erzeugt worden sind, und dass die Parameter dieser Verteilungen und als versteckte Parameter die Clusterzuordnungen zu schätzen sind. Eine solche allgemeinere Modellierung wird beispielsweise vom populären Clusteringverfahren AUTOCLASS [22] benutzt.

12.10.2 Hierarchische Clustering-Verfahren

Eine zweite große Gruppe von Clustering-Verfahren sind die *hierarchischen Clusteringverfahren* [41]. Bei diesen Verfahren wird nicht nur ein einzelnes Clustering \mathcal{C} produziert, sondern eine Serie ineinander geschachtelter Clusterings $\mathcal{C}_1, \dots, \mathcal{C}_m$. Die beiden Endpunkte dieser Liste geschachtelter Clusterings sind dabei zum einen das aus lauter einelementigen Teilmengen bestehende Clustering, und zum anderen das nur aus einem einzigen Cluster mit allen Elementen aus S bestehende Clustering. Hierarchische Clusterungen werden üblicherweise in sogenannten *Dendrogrammen* dargestellt, siehe Abbildung 12.20.

Eine populäre Verfahrensklasse zur Erzeugung solcher Clusterings ist das *agglomerative Clustern*, das ausgehend von den einelementigen Clustern die Hierarchie aufsteigend aufbaut. Dies geschieht einfach dadurch, dass in jedem Schritt des Verfahrens die beiden einander ähnlichsten Cluster zu einem neuen Cluster zusammengefasst werden, wie das im Dendrogramm entsprechend durch eine waagerechte Linie eingezeichnet ist. Die Ähnlichkeit zweier einelementiger Cluster ist dabei einfach die Ähnlichkeit ihrer beiden jeweils einzigen Elemente. Fasst man zwei Cluster zusammen, kann die Ähnlichkeit des neuen Clusters zu allen bisherigen Clustern immer aus den Ähnlichkeiten der beiden bisherigen Cluster zu allen anderen Clustern berechnet werden. Je nachdem, welche Berechnungsvorschrift man dabei wählt, erhält man unterschiedliche Ergebnisse.

$$dist(C_1 \cup C_2, C_3) := f(dist(C_1, C_3), dist(C_2, C_3))$$

Für $f = min$ ergibt sich das sogenannte *single link clustering*, für $f = max$ das *complete-link clustering*, und für $f = average$ das *average-link clustering*.

Hierarchische Clusteringverfahren haben den Vorteil, dass der Nutzer aus dem Ergebnis des Verfahrens jederzeit ein nicht-hierarchisches Clustering mit der gewünschten Anzahl von Clustern ableiten kann. Durch etwas komplexere Wahl von f („Ward’s function“) kann darüber

$$\begin{aligned} & \{(x_1), (x_2), (x_3), (x_4), (x_5)\} \\ & \{(x_1, x_2), (x_3), (x_4), (x_5)\} \\ & \{(x_1, x_2), (x_3, x_4), (x_5)\} \\ & \{(x_1, x_2, x_3, x_4), (x_5)\} \\ & \{(x_1, x_2, x_3, x_4, x_5)\} \end{aligned}$$

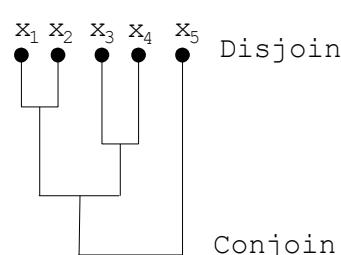


Abbildung 12.20: Hierarchisches Clustering und Dendrogramm.

hinaus sichergestellt werden, dass bei jedem einzelnen Aggregationsschritt die Summe der (quadruierten) Abstände minimiert wird (dies gilt allerdings nicht für das Gesamtergebnis) [41].

12.11 Verstärkungslernen

Mit Verstärkungslernen (engl. Reinforcement Learning) [95] bezeichnet man das Problem welches sich einem Agenten stellt, wenn er durch „Ausprobieren“ in einer dynamischen Umgebung optimales Verhalten lernen soll. Beispiele finden sich im Bereich der Robotik, der Steuerung von industriellen Anlagen oder auch bei Spielen. Die Anwendungen von Verstärkungslernen haben gemein, dass ein Agent komplexe Abfolgen von Aktionen lernen muss. Beim Backgammon sind die Aktionen (möglichst geschickte) Spielzüge, bei mobilen Robotern Kommandos zur Motorsteuerung.

Die Lernaufgabe des Verstärkungslernen unterscheidet sich ganz wesentlich von der des Funktionslernens aus Beispielen. Beim Verstärkungslernen erhält der Agent keine Beispiele von gutem oder schlechtem Verhalten. Stattdessen darf er experimentieren und wird für erfolgreiches Verhalten belohnt (bzw. für Misserfolge bestraft). Hinter diesem Vorgehen steckt die Hypothese, dass es einfacher ist den Agenten auf diese Weise lernen zu lassen, als das Verhalten von Hand zu programmieren und zu warten.

Formal besteht ein Verstärkungslernproblem aus drei Komponenten:

- Eine Menge von Zuständen \mathcal{S} : Zu jedem Zeitpunkt befindet sich der Agent in genau einem der Zustände. Die Zustände sind im einfachsten Fall diskret¹² und wir werden im Folgenden annehmen, dass die Menge der Zustände endlich ist. Beim Backgammon ist \mathcal{S} z. B. die Menge aller möglichen Spielpositionen.
- Eine Menge von Aktionen \mathcal{A} : Aktionen bringen den Agenten gemäß einer Zustandsübergangsfunktion $\delta : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ von einem Zustand zu einem anderen.
- Eine Belohnungsfunktion $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$: Jedemal, wenn der Agent eine Aktion ausführt, erhält er eine (evtl. negative) Belohnung gemäß dieser Belohnungsfunktion. Die Belohnung ist in der Regel eine reelle Zahl.

Diese Komponenten stellen die Umgebung dar, in welcher der Agent agiert. Eine besonders einfache Beispielumgebung ist in Abbildung 12.21 dargestellt. Jedes Kästchen entspricht einem Zustand. Insgesamt gibt es 19 Zustände. Als Aktionen stehen Bewegungen nach *Osten*, *Westen*, *Süden*, *Norden* zur Auswahl. Die Zustandsübergangsfunktion δ gibt den entsprechenden Nachfolgezustand an. Läuft der Agent gegen eine Wand, so bleibt er im aktuellen Zustand. Die Belohnungsfunktion r ist überall null, außer der Agent bewegt sich in den mit *Z* markierten Zustand. Für diese Aktion erhält er eine Belohnung von 100. Im Zustand *Z* gibt es nur eine Aktion – nämlich in ihm zu bleiben. Der Agent erhält keine weiteren Belohnungen. Man nennt einen solchen Zustand *absorbierend*.

Wie interagiert der Agent mit der Umgebung? Von seinem aktuellen Zustand wählt der Agent eine Aktion. Die Auswahl der Aktion geschieht durch die sogenannte Verhaltensregel (engl. policy) $\pi : \mathcal{S} \rightarrow \mathcal{A}$ des Agenten. Eine gute Verhaltensregel ist das Lernziel des Agenten. Als

¹² Weitere Methoden erlauben auch kontinuierliche Zustandsräume, siehe z.B. [11].

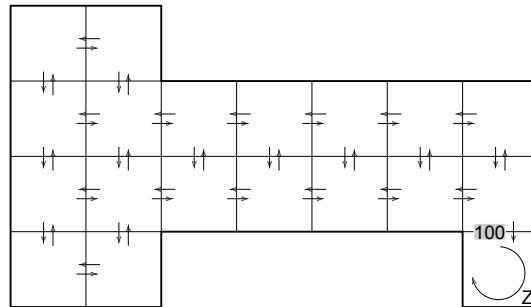


Abbildung 12.21: Eine einfache Beispielumgebung. Jeder Kasten entspricht einem Zustand.

Rückmeldung auf eine Aktion erhält der Agent seinen neuen Zustand und die Belohnung. Beides hängt lediglich vom aktuellen Zustand und der gewählten Aktion ab, nicht aber von Zuständen oder Aktionen, die weiter in der Vergangenheit liegen. Man bezeichnet dieses Szenario auch als deterministischen Markov'schen Entscheidungsprozess. Dieser Vorgang wiederholt sich solange der Agent lebt. Bis jetzt ist allerdings nicht klar, wie eine gute Verhaltensregel aussehen soll. Intuitiv ist klar, dass der Agent möglichst viele Belohnungen erhalten soll. Aber wie definiert man „möglichst viele Belohnungen“ am treffendsten?

12.11.1 Wann handelt ein Agent optimal?

Es gibt mehrere sinnvolle Möglichkeiten, wie man gutes oder optimales Verhalten definieren kann. Hat der Agent eine begrenzte und bekannte Lebenszeit h , so ist es naheliegend, die Summe der Belohnungen zur Bewertung einer Verhaltensregel π heranzuziehen. In diesem Sinne gibt die folgende Funktion $V^\pi(s_t)$ die Summe der Belohnungen an, welche der Agent erhält, wenn er im Zustand s_t startet und dann der Verhaltensregel π folgt.

$$V_{sum}^\pi(s_t) = \sum_{i=0}^h r_{t+i}$$

Die optimale Verhaltensregel ist die, welche für alle Startzustände die Funktion $V^\pi(s_t)$ maximiert. Ist die Lebenszeit des Agenten nicht beschränkt, ist die durchschnittliche Belohnung ein sinnvolles Kriterium.

$$V_{avg}^\pi(s_t) = \lim_{h \rightarrow \infty} \frac{1}{h} \sum_{i=0}^h r_{t+i}$$

Im weiteren Verlauf des Kapitels wird allerdings das folgende Optimalitätskriterium betrachten. Es wird diskontierte kumulative Belohnung (engl. discounted cumulative reward) genannt. Die Idee ist, früh eintretende Belohnungen höher zu bewerten als solche, die sich erst weit in der Zukunft ergeben. Hierzu dient der Parameter $0 \leq \gamma < 1$.

$$V_{dc}^\pi(s_t) = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

Weiter in der Zukunft liegende Belohnungen werden also exponentiell geringer bewertet. Je kleiner γ ist, desto stärker konzentriert sich die Bewertung auf die sofort eintretende Belohnung. Man kann γ als Zinsrate verstehen, oder auch als Wahrscheinlichkeit, dass der Agent einen weiteren Schritt lebt. Insbesondere sind aber die Algorithmen, welche die diskontierte kumulative Belohnung maximieren, besonders einfach und effizient.

Eine bemerkenswerte Eigenschaft ist, dass es eine Äquivalenz zwischen maximaler Bewertungsfunktion $V^*(s_t)$ und optimaler Verhaltensregel π^* gibt. Es reicht also aus, die Bewertungsfunktion zu maximieren. Die optimale Verhaltensregel lässt sich dann leicht ablesen. Der Agent wählt in einem Zustand s_t die Aktion a , welche die gewichtete Summe aus der sofortigen Belohnung $r(s_t, a)$ und der diskontierten Bewertungsfunktion $V_{dc}^*(\delta(s_t, a))$ des Nachfolgezustandes maximiert.

$$\pi(s_t) = \operatorname{argmax}_{a \in \mathcal{A}} [r(s_t, a) + \gamma \cdot V_{dc}^*(\delta(s_t, a))]$$

Das Beispiel in Abbildung 12.22 zeigt wie der Agent handelt, wenn er $V_{dc}^*(s_t)$ (hier für $\gamma = 0.9$) kennt. Der Agent befindet sich im Zustand e . Ihm stehen vier Aktionen zur Auswahl. Geht er nach Süden, so läuft er gegen die Wand und bleibt in Zustand e . Die sofortige Belohnung ist 0 und $V_{dc}^*(e) = 81$. Somit ist der Wert der Aktion „Süden“ $0 + 0.9 \cdot 81.0 = 72.9$. Ebenso berechnet sich der Wert für die Aktion „Westen“ als $0 + 0.9 \cdot 72.9 = 65.6$, für die Aktion „Norden“ als $0 + 0.9 \cdot 72.9 = 65.6$ und für die Aktion „Osten“ als $0 + 0.9 \cdot 90.0 = 81.0$. Der Agent geht also nach „Osten“ und befindet sich nun im Zustand f . Hier stehen wieder vier Aktionen zur Auswahl: „Westen“ hat eine Bewertung von $0 + 0.9 \cdot 81.0 = 72.9$, „Norden“ $0 + 0.9 \cdot 81.0 = 72.9$, „Osten“ $0 + 0.9 \cdot 90.0 = 81.0$ und „Süden“ $100.0 + 0.9 \cdot 0.0 = 100.0$. Der Agent geht also nach Süden und bleibt in dem absorbierenden Zustand Z .

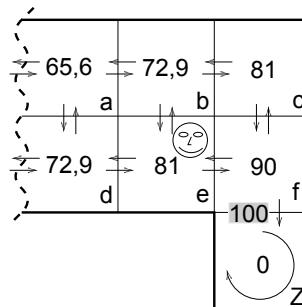


Abbildung 12.22: Die optimalen Werte für $V_{dc}^*(s_t)$ bei $\gamma = 0.9$. Der Agent geht zuerst nach Osten und dann nach Süden.

Es ist nun klar wie der Agent handelt, wenn er $V_{dc}^*(s_t)$ kennt. Die nächsten zwei Abschnitte behandelt die Aufgabe $V_{dc}^*(s_t)$ zu berechnen, bzw. zu lernen.

12.11.2 Dynamische Programmierung

Wie findet man die Verhaltensregel, welche die diskontierte kumulative Belohnung maximiert? Gehen wir zunächst davon aus, dass der Agent die Zustandsübergangsfunktion δ und die Belohnungsfunktion r bereits während des Lernens vollständig kennt. In dieser Situation kann man das Problem mittels dynamischer Programmierung lösen.

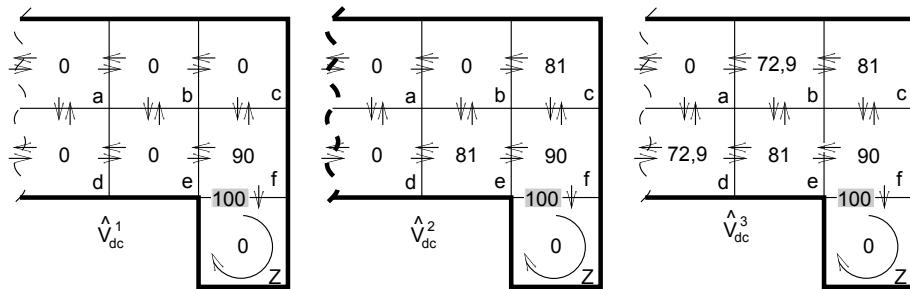


Abbildung 12.23: Die Approximation der Bewertungsfunktion durch den Value-Iteration Algorithmus nach der ersten, zweiten und dritten Iteration für $\gamma = 0.9$.

Mit Bezug auf das Optimalitätskriterium aus dem vorangegangenen Abschnitt suchen wir die Verhaltensregel π , die für jeden Zustand die Bewertungsfunktion maximiert.

$$V_{dc}^*(s) = \max_{\pi} \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

Ein Algorithmus, der diese Verhaltensregel bzw. die maximale Bewertungsfunktion findet, ist der Value-Iteration Algorithmus [13]. Die Idee des Algorithmus ist die Folgende. Der Algorithmus startet mit einer beliebigen Initialisierung seiner Schätzung $\hat{V}_{dc}^0(s)$ für $V_{dc}^*(s)$. Zum Beispiel ist also $\hat{V}_{dc}^0(s) = 0$ für alle Zustände $s \in \mathcal{S}$ zulässig. Dann iteriert der Algorithmus mehrmals über alle Zustände. Für jeden Zustand s wird die optimale Aktion basierend auf der aktuellen Schätzung $\hat{V}_{dc}^t(s)$ berechnet und der Wert $\hat{V}_{dc}^{t+1}(s) = \max_{a \in \mathcal{A}} [r(s, a) + \gamma \cdot \hat{V}_{dc}^t(\delta(s, a))]$ aktualisiert. Man kann zeigen, dass für $t \rightarrow \infty$ der Wert $\hat{V}_{dc}^t(s)$ gegen $V_{dc}^*(s)$ konvergiert.

Algorithmus 1 (Value-Iteration).

- Initialisiere $\hat{V}_{dc}^0(s) = 0$ für alle $s \in \mathcal{S}, t = 0$
- Solange $\hat{V}_{dc}^t(s)$ nicht gut genug
 - Für alle $s \in \mathcal{S}$
 - * $\hat{V}_{dc}^{t+1}(s) = \max_{a \in \mathcal{A}} [r(s, a) + \gamma \cdot \hat{V}_{dc}^t(\delta(s, a))]$
 - $t := t + 1$
- Gib \hat{V}_{dc}^t aus

Abbildung 12.23 zeigt den Fortschritt des Algorithmus in einem Ausschnitt der Beispielumgebung nach dem ersten, zweiten und dritten Durchlauf durch die Schleife.

Es sei an die relativ starken Annahmen erinnert, die der Value-Iteration Algorithmus benötigt – nämlich dass die Zustandsübergangsfunktion δ und die Belohnungsfunktion r bekannt sind. Dies ist bei den meisten Agenten nicht gegeben.

12.11.3 Q-Learning – Lernen in unbekannter Umgebung

In realen Anwendungen sind gerade die Zustandsübergangsfunktion δ und die Belohnungsfunktion r die Teile der Umwelt, die der Agent erst durch Exploration lernen muss. Ein mobiler Roboter, den man in ein unbekanntes Gebäude setzt, weiss nicht in welchem Zustand er sich befindet, wenn er 5 Meter vorwärts durch eine Tür und dann 3 Meter nach rechts geht. Dort könnte eine Wand im Weg sein, oder er könnte eine Treppe hinuntergefallen sein. Desweiteren soll der Roboter erst herausfinden, wofür er Belohnungen erhält. Will ein Agent in unbekannter Umgebung lernen, muss er also Experimente durchführen.

Das erste Problem, welches gelöst werden muss, ist das der Aktionsauswahl. Den Ausdruck $\arg\max_{a \in \mathcal{A}} [r(s_t, a) + \gamma \cdot V_{dc}^*(\delta(s_t, a))]$ kann man ohne Kenntnis von r und δ nicht auswerten. Man kann dies aber leicht umgehen, indem man eine Bewertungsfunktion $Q(s, a)$ direkt für Aktionen lernt. $Q(s, a)$ ist definiert als $r(s, a) + \gamma \cdot V_{dc}^*(\delta(s, a))$. Kennt man $Q(s, a)$ für alle Aktionen im aktuellen Zustand, so kann man wie gehabt die optimale Aktion leicht bestimmen.

Ein Lernalgorithmus, der Q durch Exploration lernt, ist der sogenannte Q-Learning Algorithmus [92]. Er funktioniert nach einem ähnlichen Prinzip wie der Value-Iteration Algorithmus, benutzt aber nur Informationen, an die der Agent durch Ausprobieren von Aktionen selber gelangen kann. Immer wenn der Agent eine Aktion a ausprobiert, nimmt er seinen neuen Zustand s' und die erhaltene Belohnung r wahr. Die geschätzte Bewertung $\hat{Q}(s, a)$ des Ausgangszustandes verändert er dann nach der Regel

$$\hat{Q}(s, a) := r + \gamma \cdot \max_{a' \in \mathcal{A}} \hat{Q}(s', a')$$

Wieder kann man zeigen, dass der Algorithmus die optimale Bewertungsfunktion Q lernt, wenn jede Aktion in jedem Zustand nur häufig genug durchlaufen wird.

Algorithmus 2 (Q-Lernen).

- Initialisiere $\hat{Q}(s, a) = 0$ für alle $s \in \mathcal{S}$ und $a \in \mathcal{A}$, s ist Startzustand
- Wiederhole solange der Agent lebt
 - Wähle eine Aktion a und führe sie aus
 - Erhalte Belohnung r und neuen Zustand s'
 - $\hat{Q}(s, a) := r + \gamma \cdot \max_{a' \in \mathcal{A}} \hat{Q}(s', a')$
 - $s := s'$

Die Aktionsauswahl ist im Algorithmus nicht näher spezifiziert solange sichergestellt ist, dass jede Aktion häufig genug ausprobiert wird. Hier ergibt sich das sogenannte „Exploration vs. Exploitation“ Dilemma. Zu Beginn des Algorithmus wird der Agent Aktionen zufällig auswählen, da alle die gleiche Bewertung haben. Aber nach einiger Zeit hat der Agent ein passables \hat{Q} gelernt. Sollte der Agent weiterhin beliebige Aktionen ausprobieren und so vielleicht ein beseres \hat{Q} finden (Exploration), oder sollte der Agent immer die Aktion mit dem größten $\hat{Q}(s, a)$ wählen, um die Belohnung zu maximieren (Exploitation)? Oft wird eine Mischung zwischen beidem gewählt. Mit über die Zeit steigender Wahrscheinlichkeit wählt der Agent die Aktion mit dem größten $Q(s, a)$, aber mit einer gewissen Wahrscheinlichkeit auch eine der anderen Aktionen. Eine optimale Lösung dieses Problems ist allerdings schwierig.

12.11.4 Erweiterungen

Aus Platzgründen wurde hier nur die einfachste Form des Verstärkungslernens behandelt. Einige Erweiterungen sollen aber trotzdem kurz erwähnt werden:

Nicht-Deterministische Umgebungen. Aktionen wurden als deterministisch angenommen d.h. sie führten immer zum gleichen Nachfolgezustand und zur gleichen Belohnung. Oft ist dies aber nicht der Fall. Z. B. wird ein mobiler Roboter auf den Befehl „gehe einen Meter vorwärts“ oft sich nur 80cm weit bewegen, da der Boden an der Stelle uneben ist. Es sind also mehrere Nachfolgezustände möglich, die jeweils mit einer gewissen Wahrscheinlichkeit eintreten. Gleiches kann auch für die Belohnungen gelten. Der Q-Learning Algorithmus kann aber auch diesen Fall handhaben, wenn man die rekursive Formel durch $\hat{Q}(s, a) := (1 - \alpha)\hat{Q}(s, a) + \alpha[r + \gamma \cdot \max_{a' \in \mathcal{A}} \hat{Q}(s', a')]$ ersetzt, wobei $0 < \alpha \leq 1$ eine Lernrate ist, die langsam gesenkt wird.

Verbogener Zustand. Oft kann der Agent seinen Zustand nicht eindeutig bestimmen. Für einen Roboter mögen zwei Räume gleich „aussehen“. Die Handhabung dieses Problems ist allerdings sehr schwierig (siehe z. B. [46]).

Modellbasiertes Verstärkungslernen. Eine Alternative zum Q-Learning ist es, zuerst δ und r zu lernen und dann Algorithmen wie Value-Iteration anzuwenden. Auch Kombinationen beider Methoden werden untersucht.

Funktionslernen. Für jeden Zustand eine separate Bewertung zu lernen und speichern ist für große Zustandsmengen zu ineffizient. Indem man Zustände durch Merkmale beschreibt, versucht man die Bewertungsfunktion mit Methoden aus Kapitel 12.2 zu lernen. Gelingt es gut zu generalisieren, kann der Agent auch in Situationen gut handeln, die er vorher noch nicht untersucht hat.

Literatur. Als weiterführende Literatur sind [95], sowie, etwas älter und grundlegender, [89] und in kompakterer Form [64] und [46] zu empfehlen.

12.12 Weiterführende Themen

Im vorliegenden Kapitel haben wir eine an der Künstlichen Intelligenz orientierte Einführung in das Maschinellen Lernen gegeben, bei der wir uns auf die grundlegenden klassischen Verfahren für die drei Haupt-Lernaufgaben überwachtes Lernen, unüberwachtes Lernen und Verstärkungslernen konzentriert haben. Für alle drei Lernaufgaben haben wir für dieses Buch besonders grundlegende Algorithmen ausgewählt, an denen sich die wichtigsten Prinzipien gut erläutern lassen. Für jeden der vorgestellten Grundalgorithmen sind in den vergangenen Jahren natürlich unzählige Varianten entstanden, die sich durch universellere Verwendbarkeit, bessere Lernleistung oder höhere Effizienz auszeichnen. Wir ermutigen die interessierten Leser deshalb ausdrücklich, nach Lektüre dieses Kapitels auch weiterführende Quellen zu konsultieren, wie sie beispielsweise die führenden Zeitschriften darstellen (z.B. *Machine Learning* [3], *Journal of Machine Learning Research* [2] oder *Data Mining and Knowledge Discovery* [1]). Aber auch ausführlichere und vollständigere Lehrbücher können wir zur Vertiefung empfehlen

(z.B. [65], [6] (in deutscher Sprache), [7] (englisch), [14], [37] oder [66]). Speziell auf das Data Mining konzentriert sich [36].

Zur weiteren Vertiefung empfehlen sich insbesondere Lehrbücher und Artikel zu den Themen, die in diesem Kapitel aus Platzgründen nicht behandelt werden, die aber in der aktuellen Forschung zum maschinellen Lernen eine zentrale Rolle einnehmen. Dies sind insbesondere Lernverfahren für probabilistische, insbesondere graphische Modelle (siehe z.B. [45], [56]), weitere Verfahren des unüberwachten und halbüberwachten (semi-supervised) Lernens (siehe z.B. [21]) sowie das stark wachsende Gebiet des Lernens mit strukturierten Daten (siehe z.B. [32], [34]) sowie der Prädiktion strukturierter Objekte [10]. Auch Ensemble-Methoden, bei denen mehrere Hypothesen zu einem stärkeren Modell kombiniert werden, sind hier von Interesse (siehe z.B. Bagging [16] und Boosting [78]), ebenso wie Techniken des aktiven Lernens (siehe z.B. [81]).

Literaturverzeichnis

- [1] (2012). Data mining and knowledge discovery. Springer Verlag. ISSN 1384-5810 (print version), 1573-756X (electronic version).
- [2] (2012). Journal of machine learning research. Free and Online. ISSN 1533-7928 (electronic version).
- [3] (2012). Machine learning. Springer Verlag. ISSN 0885-6125 (print version), ISSN 1573-0565 (electronic version).
- [4] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., und Verkamo, A. I. (1996). Fast discovery of association rules. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., und Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*, chapter 12, pages 307–328. AAAI/MIT Press, Cambridge, USA.
- [5] Aha, D., Kibler, D., und Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, 6:37–66.
- [6] Alpaydin, E. (2008). *Maschinelles Lernen*. Oldenbourg Wissenschaftsverlag.
- [7] Alpaydin, E. (2010). *Introduction to Machine learning (2nd ed.)*. MIT Press.
- [8] Anthony, M. und Biggs, N. (1992). *Computational Learning Theory*. Univ. Press, Cambridge, Mass. ua.
- [9] Bacher, J. (1996). *Clusteranalyse*. Oldenbourg, München.
- [10] Bakir, G. H., Hofmann, T., Schölkopf, B., Smola, A. J., Taskar, B., und Vishwanathan, S. V. N., editors (2007). *Predicting Structured Data*. MIT Press, Cambridge, MA.
- [11] Baxter, J. und Bartlett, P. L. (2000). Direct gradient-based reinforcement learning. In *Proceedings of the International Symposium on Circuits and Systems*, pages III–271–274.
- [12] Bell, R. M., Koren, Y., und Volinsky, C. (2010). All together now: A perspective on the netflix prize. *Chance*, 23:24–29.
- [13] Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.
- [14] Bishop, C. M. (2008). *Pattern Recognition and Machine Learning*. Springer.
- [15] Boser, B. E., Guyon, I. M., und Vapnik, V. N. (1992). A traininig algorithm for optimal margin classifiers. In Haussler, D., editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152.
- [16] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.
- [17] Breiman, L. (1998). Arcing classifiers. *Annals of Statistics*, 26(3):801–849.

- [18] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5.
- [19] Breiman, L., Friedman, J., Olshen, R., und Stone, C. (1984). *Classification and regression trees*. Belmont, Wadsworth.
- [20] Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.
- [21] Chapelle, O., Schölkopf, B., und Zien, A., editors (2006). *Semi-Supervised Learning*. MIT Press, Cambridge, MA.
- [22] Cheeseman, P. und Stutz, J. (1996). Bayesian classification (AutoClass): Theory and results. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., und Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*, chapter 6, pages 153–180. AAAI/MIT Press, Cambridge, USA.
- [23] Christianini, N. und Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines*. Cambridge University Press.
- [24] Cortes, C. und Vapnik, V. N. (1995). Support–vector networks. *Machine Learning Journal*, 20:273–297.
- [25] Cover, T. M. und Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27.
- [26] Diehl, J. M. und Arbinger, R. (1990). *Einführung in die Inferenzstatistik*. D. Klotz Verlag, Eschborn.
- [27] Dietterich, T. G. (2000). Ensemble methods in machine learning. In Kittler, J. und Roli, F., editors, *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer.
- [28] Emde, W. und Wettschereck, D. (1996). Rational instance based learning. In Saitta, L., editor, *Proceedings 13th International Conference on Machine Learning*, pages 122–130. Morgan Kaufmann.
- [29] Fayyad, U. M., Piatetsky-Shapiro, G., und Smyth, P. (1996). From data mining to knowledge discovery: An overview. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., und Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*, chapter 1, pages 1–34. AAAI/MIT Press, Cambridge, USA.
- [30] Freund und Schapire (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *JCSS: Journal of Computer and System Sciences*, 55.
- [31] Freund, Y. und Schapire, R. (1999). A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780.
- [32] Gärtnner, T. (2003). A survey of kernels for structured data. *SIGKDD Explorations*, 5(1):49–58.
- [33] Gebhardt, F. (1991). Choosing among competing generalizations. *Knowledge Acquisition*, 3:361–380.
- [34] Getoor, L. und Taskar, B., editors (2007). *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA.
- [35] Guzella, T. S. und Caminhas, W. M. (2009). A review of machine learning approaches to spam filtering. *Expert Syst. Appl.*, 36(7):10206–10222.
- [36] Han, J. und Kamber, M. (2011). *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufmann.
- [37] Hastie, T., Tibshirani, R., und Friedman, J. (2008). *The Elements of Statistical Learning*. Springer.

- [38] Haussler, D. (1988). Quantifying inductive bias: Ai learning algorithms and valiant's learning framework. *Artificial Intelligence*, 36:177–221.
- [39] Hoffmann, A. (1991). Die Theorie des Lernbaren – ein Überblick. *KI*, 1:7–11.
- [40] Horvath, T., Wrobel, S., und Bohnebeck, U. (2000). Rational instance-based learning with lists and terms. *Machine Learning Journal*. to appear.
- [41] Jain, A. und Dubes, R. (1988). *Algorithms for Clustering Data*. Prentice Hall, New York.
- [42] Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, pages 137–142, Berlin. Springer.
- [43] Joachims, T. (1999). Making large-scale SVM learning practical. In Schölkopf, B., Burges, C., und Smola, A., editors, *Advances in Kernel Methods – Support Vector Learning*, chapter 11. MIT Press, Cambridge, MA.
- [44] Joachims, T. (2006). Training linear SVMs in linear time. In *ACM SIGKDD International Conference On Knowledge Discovery and Data Mining (KDD)*, pages 217–226.
- [45] Jordan, M. I., editor (1998). *Learning in Graphical Models*. MIT Press, Cambridge, MA.
- [46] Kaelbling, L. P., Littman, M. L., und Moore, A. W. (1996). Reinforcement learning: A survey. *Artificial Intelligence Research*, 4:237–285.
- [47] Kearns und Valiant (1994). Cryptographic limitations on learning boolean formulae and finite automata. *JACM: Journal of the ACM*, 41.
- [48] Kearns, M. (1990). *The Computational Complexity of Machine Learning*. ACM Distinguished Dissertation. The MIT Press.
- [49] Kearns, M., Mansour, Y., Ng, A., und Ron, D. (1997). An experimental and theoretical comparison of model selection methods. *Machine Learning*, 27:7–50.
- [50] Kearns, M. und Vazirani, U. (1994). *An Introduction to Computational Learning Theory*. MIT Press.
- [51] Kirsten, M., Wrobel, S., Dahmen, F.-W., und Dahmen, H.-C. (1998). Einsatz von Data-Mining-Techniken zur Analyse ökologischer Standort- und Pflanzendaten. *Künstliche Intelligenz Journal*, 12(2):39–42.
- [52] Klösgen, W. (1996). Explora: A multipattern and multistrategy discovery assistant. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., und Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*, chapter 10, pages 249–271. AAAI/MIT Press, Cambridge, USA.
- [53] Klösgen, W. (2000a). Change analysis. In Klösgen, W. und Zytkow, J., editors, *Handbook of Knowledge Discovery and Data Mining*. Oxford University Press, London. to appear.
- [54] Klösgen, W. (2000b). Deviation analysis. In Klösgen, W. und Zytkow, J., editors, *Handbook of Knowledge Discovery and Data Mining*. Oxford University Press, London. to appear.
- [55] Klösgen, W. (2000c). Subgroup patterns. In Klösgen, W. und Zytkow, J., editors, *Handbook of Knowledge Discovery and Data Mining*. Oxford University Press, London. to appear.
- [56] Koller, D. und Friedman, N., editors (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge, MA.
- [57] Lavrač, N. und Džeroski, S. (1994). *Inductive Logic Programming – Techniques and Applications*. Ellis Horwood, Hertfordshire.
- [58] Lunts, A. und Brailovskiy, V. (1967). Evaluation of attributes obtained in statistical decision rules. *Engineering Cybernetics*, 3:98–109.

- [59] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symp. on Mathematical Statistics and Probability*.
- [60] Mathewson, J. (2012). Three strategies for seo post google panda. IBM Press (online). <http://www.ibmpressbooks.com/articles/article.asp?p=1829428>.
- [61] Michalski, R. (1986). Understanding the nature of learning. In Michalski, Carbonell, und Mitchell, editors, *Machine Learning – An Artificial Intelligence Approach*. Morgan Kaufmann, Los Altos, California.
- [62] Michalski, R. S. und Stepp, R. E. (1983). Learning from observation: Conceptual clustering. In Michalski, R., Carbonell, J., und Mitchell, T., editors, *Machine Learning – An Artificial Intelligence Approach, Vol. I*, volume I, pages 331–363. Tioga, Palo Alto, CA.
- [63] Mingers, J. (1989). An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4:227–243.
- [64] Mitchell, T. M. (1997a). *Machine Learning*. McGraw Hill, New York.
- [65] Mitchell, T. M. (1997b). *Machine learning*. McGraw Hill series in computer science. McGraw-Hill.
- [66] Murphy, K. P. (2012, to appear). *Machine Learning: A Probabilistic Perspective*. MIT Press.
- [67] Murthy, S. K. (1998). Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2(4):345–389.
- [68] Murthy, S. K., Kasif, S., und Salzberg, S. (1994). A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32.
- [69] Natarajan, B. K. (1991). *Machine Learning. A Theoretical Approach*. Morgan Kaufmann, San Mateo, CA.
- [70] Nienhuys-Cheng, S.-H. und Wolf, R. d. (1997). *Foundations of Inductive Logic Programming*. LNAI Tutorial 1228. Springer Verlag, Berlin, New York.
- [71] Phua, C., Lee, V. C. S., Smith-Miles, K., und Gayler, R. W. (2010). A comprehensive survey of data mining-based fraud detection research. *Arxiv CoRR*, abs/1009.6119.
- [72] Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In Schölkopf, B., Burges, C., und Smola, A., editors, *Advances in Kernel Methods – Support Vector Learning*, chapter 12. MIT-Press.
- [73] Quinlan, J. (1990). Learning logical definitions from relations. *Machine Learning*, 5(3):239–266.
- [74] Quinlan, J. R. (1993). *C4.5 – Programs for Machine Learning*. Morgan Kaufman, San Mateo, CA. Accompanying software available.
- [75] Rencher, A. C. (1998). *Multivariate Statistical Inference and Applications*. John Wiley, New York.
- [76] Salzberg, S. (1991). A nearest hyperrectangle learning method. *Machine Learning*, 6:277–309.
- [77] Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5:197–227.
- [78] Schapire, R. E. und Freund, Y. (2012). *Boosting: Foundations and Algorithms*. MIT Press, Cambridge, MA.
- [79] Scheffer, T. und Joachims, T. (1999). Expected error analysis for model selection. In Bratko, I. und Džeroski, S., editors, *Proc. 16th Int. Conf. on Machine Learning*, San Mateo, CA. Morgan Kaufman.
- [80] Schoelkopf, B. und Smola, A. J. (2002). *Learning with Kernels*. The MIT Press, Cambridge, MA.

- [81] Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.
- [82] Shafer, J. C., Agrawal, R., und Mehta, M. (1996). SPRINT: A scalable parallel classifier for data mining. In Vijayaraman, T. M., Buchmann, A. P., Mohan, C., und Sarda, N. L., editors, *Proc. 22th Int. Conf. on Very Large Data Bases (VLDB96)*, pages 544–555, San Mateo, CA. Morgan Kaufman.
- [83] Shalev-Shwartz, S., Singer, Y., und Srebro, N. (2007). Pegasos: Primal estimated sub-gradient SOLver for SVM. In Ghahramani, Z., editor, *International Conference on Machine Learning (ICML)*, volume 227 of *ACM International Conference Proceeding Series*, pages 807–814. ACM.
- [84] Shannon, C. und Weaver, W. (1969). *The Mathematical Theory of Communication*, pages 20–21. Chapman and Hall, 4 edition.
- [85] Simon, H. A. (1983). Why should machines learn? In Michalski, R. S., Carbonell, J. G., und Mitchell, T. M., editors, *Machine Learning – An Artificial Intelligence Approach*, volume 1, chapter 2, pages 25–39. Morgan Kaufmann, Palo Alto, CA.
- [86] Srikant, R. und Agrawal, R. (1995). Mining generalized association rules. In Dayal, U., Gray, P. M. D., und Nishio, S., editors, *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11–15, 1995, Zurich, Switzerland*, pages 407–419. Morgan Kaufmann.
- [87] Srikant, R. und Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In Apers, P. M. G., Bouzeghoub, M., und Gardarin, G., editors, *Advances in Database Technology – EDBT'96, 5th International Conference on Extending Database Technology, Avignon, France, March 25–29, 1996, Proceedings*, volume 1057 of *Lecture Notes in Computer Science*, pages 3–17. Springer Verlag.
- [88] Steinhagen, D. und Langer, K. (1977). *Clusteranalyse*. Walter DeGruyter, Berlin, New York.
- [89] Sutton, R. und Barto, G. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge.
- [90] Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, 27(11): 1134–1142.
- [91] Vapnik, V. (1998). *Statistical Learning Theory*. Wiley, Chichester, GB.
- [92] Watkins, C. und Dayan, P. (1992). Technical note: Q-learning. *Machine Learning*, 8(3):279–292.
- [93] Wettschereck, D., Aha, D., und Mohri, T. (1997). A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11:273–314.
- [94] Wettschereck, D. und Dietterich, T. (1995). An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine Learning*, 19:5–27.
- [95] Wiering, M. und van Otterlo, M., editors (2012). *Reinforcement Learning: State-of-the-Art (Adaptation, Learning, and Optimization)*. Springer.
- [96] Wrobel, S. (1997). An algorithm for multi-relational discovery of subgroups. In Komorowski, J. und Zytkow, J., editors, *Principles of Data Minig and Knowledge Discovery: First European Symposium (PKDD 97)*, pages 78–87, Berlin, New York. Springer.
- [97] Wrobel, S. (1998). Data Mining und Wissensentdeckung in Datenbanken. *Künstliche Intelligenz*, 12(1):6–10.

13 Sprachverarbeitung

Wolfgang Menzel

Dieses Kapitel soll einen ersten Überblick über die spezifischen Probleme der Sprachverarbeitung, ihre typischen Lösungsansätze, sowie die Querbeziehungen zu den anderen Teilbereichen der KI vermitteln. Für eine grundlegende Einführung in die Methodik des Gebietes müssen sicherlich spezialisierte Lehrbücher herangezogen werden, z.B. [22], aber auch [53] bzw. [72]. In ihnen werden die hier vorgestellten Ansätze in einer Tiefe behandelt, für die im Rahmen eines Handbuches zu dem ohnehin schon sehr breiten Themenfeld der Künstlichen Intelligenz naturgemäß nicht ausreichend Platz zur Verfügung steht.

13.1 Sprache und sprachliche Beschreibungsebenen

Sprache und Denken sind zwei eng miteinander verzahnte Bereiche der menschlichen Kognition. Sprache ist schon wegen ihrer singulären Rolle als universelles Kommunikations- und Speichermedium eine fundamentale Voraussetzung für praktisch alle Prozesse der sozialen Interaktion, der kollektiven Erkenntnisgewinnung und der Wissensvermittlung. Darüber hinaus ist der Gebrauch von Sprache selbst eine Form ziel- und interessegeleiteten Handelns und ordnet sich somit nahtlos in das breite Spektrum menschlicher Intelligenzleistungen ein. Die enge Verquickung zeigt sich nicht nur im intersubjektiven Bereich, sondern auch beim Individuum selbst. So finden mentale Abläufe ihren Reflex oftmals im sprachlichen Ausdruck und gleichzeitig wächst Erkenntnis auch bei der Arbeit an der sprachlichen Formulierung. Wenngleich im Extremfall auch erhebliche Dissoziationen zwischen Sprachbeherrschung und allgemeiner Intelligenz beobachtet werden können, so entwickeln sich doch Sprache und Intelligenz in einer engen Wechselwirkung: Spracherwerb kommt einerseits ohne ein Minimum an kognitiven Fähigkeiten nicht aus, ist andererseits aber auch Voraussetzung für deutliche Fortschritte in der allgemeinen Intelligenzsentwicklung. Wichtige kognitive Leistungen wie Kategorisierung, Benennung und Abstraktion sind so nur schwer von einer hinreichenden Sprachbeherrschung zu trennen.

Angesichts dessen darf es nicht verwundern, dass sich zwischen sprachlichen Einheiten und kognitiven Kategorien ein weitgehend systematisches, wenngleich auch in vielen Fällen nicht eineindeutiges Abbildungsverhältnis herausgebildet hat. Dieser Zusammenhang wird von uns gemeinhin als so selbstverständlich betrachtet, dass wir für viele Zwecke die sprachliche Form als identisch mit dem durch sie bezeichneten Konzept ansehen können (z.B. im Falle der Wörter *Mensch*, *schlafen* und *hart*). Eine genauere Kennzeichnung wird meist erst dann erforderlich, wenn unterschiedliche semantische Lesarten einer Form betrachtet werden. Man unterscheidet dann etwa Fälle von Homonymie (zufälliger Formzusammenfall) und Polysemie (aus einer Kernbedeutung abgeleitete Bedeutungsvarianten), z.B.

Homonymie	<i>Hahn</i>	männliches Huhn vs. Absperrventil
Polysemie	<i>Schule</i>	Institution vs. Gebäude vs. organisatorischer Ablauf

Eine genauere Differenzierung ist oftmals auch dann erforderlich, wenn Unterschiede in der Konzeptualisierung in verschiedenen Sprachen hervorgehoben werden sollen. So steht dem deutschen Begriff „Sprache“ im Englischen die Differenzierung in „speech“ und „language“ gegenüber.

Über große Bereiche der Sprache hinweg kann jedoch einen weitgehend systematischen Zusammenhang zwischen mentalen Konzepten und ihrer sprachlichen Beschreibung beobachten. Gerade wegen dieser systematischen Bezüge dient uns Sprache als wichtiges Fenster auf die nicht direkt beobachtbaren kognitiven Prozesse der kategorialen Wahrnehmung, der Wissensstrukturierung, des Wissenserwerbs und des Schlussfolgerns. Hinzu kommt, dass Sprachbeherrschung auf dem durch den Menschen hervorgebrachten Niveau im Vergleich zu anderen Intelligenzleistungen (z.B. visuelle Wahrnehmung, räumliche Orientierung, Handlungsplanung usw.) eine spezifisch menschliche Fähigkeit darstellt, die in der sonstigen Biosphäre kein wirkliches Äquivalent kennt.

Betrachtet man sprachliche Äußerungen im Detail, so lassen sich in ihnen eine Reihe von elementaren Einheiten identifizieren, aus denen sich sukzessiv komplexere Konstruktionen ergeben. Beispiele hierfür sind etwa der Laut (Phon), die Silbe, die Wortform, der Satz und der Diskurs. Die Wahl dieser Einheiten ist nicht unabhängig von ihrem Verwendungszweck und so stehen Repräsentationen für gesprochene Sprache gleichberechtigt neben solchen für Schriftsprache. Zudem ist in einigen Fällen eine systematische Einordnung in die genannten Kategorien nicht immer intuitiv klar¹ und teilweise durchaus auch noch umstritten.

Orthogonal zu dieser hierarchischen Aggregation kann man drei Untersuchungsperspektiven ausmachen, aus denen sich die verschiedenen linguistischen Einheiten analysieren lassen. Dabei handelt es sich neben (a) der Form einer linguistischen Konstruktion (Syntax) um ihre (b) durch das Sprachsystem vermittelte Bedeutung (Semantik) und (c) die Funktion der Äußerung im kommunikativen Zusammenhang (Pragmatik).

Unter dem Formaspekt betrachtet, setzt sich der Satz

Ist das Fenster offen?

aus vier Wortformen zusammen, die sich weiter in Silben (bzw. Morphe) und Buchstaben dekomponieren lassen. Zusätzlich ist er durch das Interpunktionszeichen und die Anordnung der Satzbestandteile eindeutig als Interrogativkonstruktion gekennzeichnet. Seine Bedeutung besteht in einer Entscheidungsfrage über den Zustand eines Fensters, das in einem bestimmten Referenzkontext eindeutig identifizierbar sein muss (vgl. etwa den Gegensatz zu *Ist ein Fenster offen?*). Hinsichtlich seiner Funktion in einem kommunikativen Zusammenhang kann er sowohl als neutrales Informationsbegehr, als Aufforderung zur Verifikation einer Vermutung, aber auch als Vorwurf bzw. Handlungsaufforderung verstanden werden. In all diesen Fällen sind dann jeweils andere Reaktionen des Hörers angemessen (vgl. Tabelle 13.1).

¹ Was sind die Wörter in dem Satz: *Beim Ein- und Ausschalten tritt u.U. Gas aus.*

Tabelle 13.1: Kommunikative Funktionen für eine natürlichsprachliche Äußerung.

Außerung	kommunikative Funktion	angemessene Reaktion
<i>Ist das Fenster offen?</i>	neutrales Informationsbegehrten	<i>Ja, es ist offen.</i>
	Verifikation einer Vermutung	<i>Zieht es etwa?</i>
	Vorwurf/Handlungsaufforderung	<i>Ich mach es ja schon zu!</i>

Formaspekte finden sich auf allen Aggregationsebenen der Sprache. So beschreibt die Phono- bzw. Graphotaktik die Möglichkeiten sequenzieller Laut- bzw. Buchstabenverknüpfungen zu Sprech- bzw. Schreibsilben, die Morphotaktik widmet sich dem inneren Aufbau der Wortformen und die (Satz-)Syntax schließlich untersucht die Regularitäten der Satzbildung.

Bedeutungskonstituierende Elemente treten erst ab der Ebene der Wortbildung in Erscheinung. Ausgehend von der Annahme eines Kompositionalsprinzips wird hier untersucht, wie sich die Bedeutung komplexer sprachlicher Konstruktionen aus der Bedeutung ihrer jeweiligen Bestandteile zusammensetzt. Dementsprechend unterscheidet man

- die lexikalische Semantik: Was ist der Bedeutungsbeitrag einer lexikalischen Einheit?
- die Morphosemantik: Welchen Veränderungen unterliegt Bedeutung im Zusammenhang mit Wortbildungsprozessen?
- die Satzsemantik: Wie setzt sich die Bedeutung eines Satzes aus der Bedeutung seiner Bestandteile zusammen?
- die Diskurssemantik: Welche Mechanismen etablieren die satzübergreifenden Bedeutungsspekte in monologischen oder dialogischen Kommunikationshandlungen.

Wesentlich ist dabei, dass

- in die Bedeutung einer Äußerung sowohl sprachliche als auch außersprachliche Aspekte einfließen und
- die Bedeutung einer sprachlichen Konstruktion nicht extern und statisch vorgegeben ist, sondern unter den Angehörigen einer Sprechergemeinschaft ausgehandelt und somit von ihnen auch variiert werden kann [129].

Gut beobachtbar ist der Einfluss außersprachlicher Gegebenheiten etwa im Bereich der Referenz, wo beispielsweise durch Pronomina und definite Beschreibungen (*Sie schreibt noch immer an dem Text!*: Wer schreibt? An welchem Text?) externe Bezüge etabliert werden, die aus einer sprachlichen Äußerung, aber auch aus der Sprache insgesamt hinausweisen können. Da überdies jeder Sprecher einer Sprache aus Gründen der Sprachökonomie an möglichst sparsamen Verbalisierungen interessiert sein muss, sind zahlreiche Äußerungen in der alltäglichen Kommunikation semantisch hochgradig unterspezifiziert, weshalb ihre Bedeutung nur unter massiver Zuhilfenahme von Wissen aus dem sprachlichen bzw. situativen Kontext rekonstruiert werden kann. (*Wie wäre es am dritten nach der Chefrunde?*: Was wird geplant? Welcher dritte ist gemeint: Mai? Juni? Wochenende? Hauseingang? Was ist die Chefrunde? usw.). Hinzu kommt, dass in vielen Fällen eine Entscheidung zwischen mehreren Bedeutungen einer Äußerung (*Was ist mit dem Hahn los?*) Nur auf der Grundlage von außersprachlichen Tatbeständen entschieden werden kann.

Fragen der kommunikativen Funktion werden in der Pragmatik thematisiert. Dabei wird die Funktion mit Hilfe von Sprechakten beschrieben [104] und man unterscheidet Kategorien wie z.B. Vorschlag, Feststellung oder Aufforderung. Voraussetzung für die Zuordnung von Sprechakten zu natürlichsprachlichen Äußerungen ist aber in jedem Fall die Verfügbarkeit von explizitem Wissen über die Absichten, Bewertungen, Kenntnisse usw. von allen beteiligten Kommunikationspartnern.

Obwohl die saubere begriffliche Trennung zwischen den soeben skizzierten Perspektiven eine wichtige Rolle bei der systematischen Untersuchung sprachlicher Phänomene spielt, ist eine vollkommen isolierte Betrachtung auch nicht angemessen, da zahlreiche Wechselwirkungen und Querbezüge zu beachten sind: So befasst sich die (Satz-) Syntax strenggenommen nur mit dem Formaspekt sprachlicher Äußerungen, doch werden zweckmäßigerweise nur solche Beschreibungen der Satzstruktur betrachtet, die im Hinblick auf die Konstruktion der Satzbedeutung hilfreich sein können. Dieser starke Bezug zur Bedeutung zieht sich auch weiter hinunter bis auf die Ebene des Sprachsignals, wo man dann von Phonemen als den elementaren bedeutsendifferenzierenden Einheiten spricht, aus denen sich Morpheme, die elementaren bedeutungstragenden Einheiten, zusammensetzen. Während der Unterschied zwischen Phon und Phonem nur die Granularität der Kategorisierung betrifft (nicht alle Phone sind bedeutsendifferenzierend und daher nicht *per se* auch Phoneme), führt die Unterscheidung zwischen Silbe und Morphem durchaus zu unterschiedlichen Segmentierungen der Lautsequenz (*tau-schen* vs. *tausch-en*).

Zusätzlich gewinnt das Gesamtbild noch dadurch an Komplexität, dass das Sprachsignal in Gestalt der prosodischen Kategorien Betonung, Intonation und Rhythmus segmentübergreifende Informationen bereithält, deren Beitrag praktisch aus allen drei Perspektiven in Erscheinung tritt: Sie signalisieren Formaspekte, beispielsweise durch die Auszeichnung von Segmentgrenzen (*Zehn Finger hab ich an jeder Hand fünf und zwanzig ...*), sie erlauben die Unterscheidung zwischen Bedeutungsalternativen etwa durch die Vergabe eines Wortakzents (*übersetzen* vs. *über-setzen*, *Schülerin* vs. *Koffein*) oder sie geben Hinweise auf die pragmatische Funktion der Äußerung, indem sie z.B. die Bewertung der Aussage durch den Sprecher übermitteln (Hervorheben wesentlicher Information, Kennzeichnung von Ironie usw.).

13.2 Sprache und KI

Angesichts der zahlreichen systematischen Bezüge zwischen Sprache und allgemeiner Intelligenz ist es nicht verwunderlich, dass die maschinelle Verarbeitung natürlicher Sprache schon recht frühzeitig in das Blickfeld der KI geriet. So bezog sich bereits der erste Vorschlag für einen vollständig operationalisierbaren Nachweis von Intelligenz [112] unmittelbar auf ein sprachliches Imitationsspiel, auch wenn berücksichtigt werden muss, dass dieser Ansatz nur auf ganz bestimmte Aspekte der menschlichen Intelligenz abzielt und daher auch noch genügend Raum für völlig inadäquate Surrogatlösungen bietet.² Trotz dieser recht offensichtlichen Mängel ist bemerkenswert, dass TURING den Intelligenzbegriff schon damals in einen unauflösbar Bezug zur Frage der sprachlichen Kommunikationsfähigkeit stellte, anstatt auf anspruchsvolle aber

² Vgl. [127] und die im Rahmen des LÖBNER-Wettbewerbs überwiegend praktizierten Lösungsansätze: Bleibe dunkel und vage, reagiere ausweichend usw.

Siehe auch <http://www.loebner.net/Prizef/loebner-prize.html>.

artifizielle Aufgaben, wie etwa das Lösen von mathematischen Problemen oder die Teilnahme an Strategiespielen (Dame, Schach, Go usw.) zurückzugreifen.

Ca. 20 Jahre später versuchten dann erste experimentelle Systemlösungen die Brücke zwischen Sprache und zielgerichtetem Handeln zu schlagen [128]. In der Folge entstand eine Vielzahl von Forschungsprototypen mit dem Ziel, zunehmend anspruchsvollere Sprachverarbeitungsaufgaben einer maschinellen Verarbeitung zugänglich zu machen und daraus ggf. kommerzielle Produkte abzuleiten. Hierbei wurden gerade in den letzten Jahren eine Reihe bemerkenswerter Erfolge erzielt, unter anderem der spektakulären Sieg eines Computersystems in einer Fernseh-Quiz-Sendung im direkten Vergleich mit menschlichen Gegenspielern [36], aber auch sprachgesteuerte persönliche Assistenten, die nach langjähriger Vorlaufforschung in Verbindung mit der zunehmenden Verbreitung von Mobilfunktechnik inzwischen einen Massenmarkt erreicht haben [29, 35].

Trotz dieser beachtlichen Fortschritte bleibt aber die vertiefte Einsicht in die ungeheure Komplexität und Vielschichtigkeit des Phänomens Sprachbehandlung ein wichtiges Resultat der Forschungen in diesem Gebiet. Auch heute noch dienen die sprachlichen Fähigkeiten des Menschen als unerreichtes Ideal und Quelle der Inspiration für entsprechende technische Systemlösungen.

Wenngleich die Sprachverarbeitung innerhalb des Gebietes der Künstlichen Intelligenz eine Sonderstellung einnimmt und wie die Bildverarbeitung auch spezifische Forschungsziele und -methoden entwickelt hat, so bleibt sie doch durch zahlreiche gemeinsame Forschungs- und Anwendungsinteressen mit der generellen Frage nach den Voraussetzungen für intelligentes Systemverhalten verbunden. Zu diesen Gemeinsamkeiten zählt nicht zuletzt

- das interessegeleitete Abwägen zwischen verschiedenen Handlungsalternativen,
- die Notwendigkeit, Entscheidungen über das optimale Systemverhalten auch unter Zeitdruck zu treffen,
- die Nutzung von ähnlichen Formalismen zur Repräsentation und Verarbeitung von (insbesondere auch unsicherem) Wissen über die Sprache, die Welt und den Dialogpartner, sowie
- das Zusammenführen von möglicherweise widersprüchlichen Informationen aus verschiedenen Quellen und über verschiedene Sinnesmodalitäten hinweg.

Hinzu kommen fundamentale Fragestellungen aus dem Bereich des Spracherwerbs und der Sprachevolution, die immer stärker in den Mittelpunkt des Interesses rücken und von denen nicht zuletzt wichtige Impulse für die Forschungen in anderen Gebieten der Künstlichen Intelligenz erwartet werden können.

Dank ihrer starken Hinwendung zu Aufgaben der Mensch-Maschine-Kommunikation hat die sprachorientierte KI von ihren frühesten Versuchen an konsequent den pragmatischen Aspekt der maschinellen Sprachverarbeitung in den Mittelpunkt gestellt. Ging man dabei anfangs noch von Szenarien aus, bei denen die kommunikative Funktion der sprachlichen Äußerung eindeutig vorgegeben war, z.B.

- Frage-Antwort-Systeme [118]
- natürlichsprachlicher Zugriff zu relationalen Datenbanken oder Expertensystemen [2, 47, 124] bzw.
- Kommandosteuerungen für Roboter [128]

so traten später neben Anwendungen mit einem reichhaltigeren Inventar an Sprechakten zunehmend auch Systemlösungen zur expliziten Modellierung von Zielen, Plänen und Wertungen eines Sprechers in den Vordergrund. In diesem Zusammenhang wurden Szenarien vor allem im Bereich der Dialogsysteme [3, 62] und der nutzeradaptiven Informationspräsentation [5, 63] untersucht, wo in Abhängigkeit von den (jeweils zu ermittelnden) Zielen, Vorkenntnissen und Präferenzen eines Nutzers ein spezifisch aufbereitetes Informationsangebot generiert werden kann. Eine vergleichbare pragmatische Orientierung findet sich auch in ersten Ansätzen zum Gesprächsdolmetschen, bei denen im Gegensatz zur klassischen maschinellen Übersetzung nicht mehr ein möglichst wortgetreues Translat angestrebt wird, sondern ein solches, das die Sprecherintention korrekt wiedergibt [51].

Auch im Hinblick auf die Rolle des Bedeutungsaspekts eröffnet sich ein sehr breites Spektrum von Sprachverarbeitungsaufgaben, wobei der Schwierigkeitsgrad der im Einzelfall zu lösen den Probleme vor allem von der Art der jeweils noch tolerierbaren vereinfachenden Annahmen abhängig ist. Traditionell finden sich in diesem Spektrum zahlreiche wichtige und auch kommerziell erfolgreiche Anwendungen, die nur einen sehr schwachen Bezug zur Bedeutung aufweisen und bei denen daher eine für viele Zwecke zufriedenstellende Lösung bereits unter ausschließlichem Rückgriff auf Formaspekte gefunden werden kann. Hierzu zählen praktisch alle wortformbasierten Verfahren wie Silbentrennung, Rechtschreibprüfung und -korrektur, Lemmatisierung usw. Auch einfache Ansätze zur syntaxbasierten Maschinellen Übersetzung fallen in diese Klasse.

Im Gegensatz dazu orientiert sich die sprachorientierte KI vor allem an solchen Problemklassen, bei denen eine explizite Beschäftigung mit der Bedeutung sprachlicher Ausdrücke unverzichtbar ist und dementsprechend geeignete Komponenten zur Sprachproduktion aus konzeptuellen Strukturen bzw. zum Sprachverständigen erforderlich machen.

Die besondere Schwierigkeit besteht hierbei aber darin, dass weder Sprachproduktion noch -verständigen strikt informationserhaltende Prozesse sind, da ein Sprecher unter Ökonomiegesichtspunkten überwiegend nur solche Informationen kommuniziert, von denen er annehmen kann, dass sie beim Hörer noch nicht bekannt sind. Sprachproduktion ist damit vor allem ein Prozess der Informationsselektion und -reduktion, während es sich beim Sprachverständigen eher um eine Informationsrekonstruktion unter Einbeziehung des verfügbaren außersprachlichen Hintergrundwissens handelt. Eine solche pragmatisch begründete Verkürzung von Information lässt sich praktisch quer durch alle sprachlichen Genres nachweisen.³ Sie ist sogar in vielen Fällen unverzichtbare Voraussetzung für eine effiziente Kommunikation. So muss das sprachverarbeitende System zum Verstehen eines Satzes wie

Der X100 hat eine maximale Drehzahl von 5000 U/min.

mindestens die folgenden Bausteine bereitstellen:

Ein X100 ist ein Auto.

Autos haben Motoren.

Motoren erzeugen (meist) Drehbewegungen.

Eine Kenngröße für Drehbewegungen ist die maximale Drehzahl.

³ Interessanterweise wirken analoge Mechanismen auch auf der phonologischen Ebene, wo ähnliche Prozesse der Informationsreduktion durch Verschleifung und Klitisierung wirksam sind: *bei dem → beim, haben wir → hamwa*

Erst mit derartigen Inferenzdiensten aber verfügt das Sprachverarbeitungssystem über die notwendigen Voraussetzungen, um ausgesprochen unhandliche Äußerungen wie

Der X100 ist ein Auto, das einen Motor besitzt, der Drehbewegungen mit einer maximalen Drehzahl von 5000 U/min erzeugt.

vermeiden zu können. Tatsächlich müssen jedoch nicht alle diese Beiträge immer direkt im Hintergrundwissen verfügbar sein, da im Falle von unvollständigen Ableitungen die fehlenden Teile unter günstigen Umständen auch induktiv erschlossen werden können: Sollte dem System noch unbekannt sein, dass der X100 ein Auto ist, so lässt sich dies möglicherweise aus der Tatsache rekonstruieren, dass Autos Motoren besitzen, die ein Drehmoment erzeugen. Allerdings sind solche Schlüsse immer mit einer gewissen Unsicherheit behaftet: Der X100 kann sowohl der Motor selbst, als auch ein anderes motorisiertes Gerät sein.

Gerade das Problem der Informationsrekonstruktion stellt beim Sprachverständigen eine zentrale Herausforderung für die sprachorientierte KI dar, müssen hier doch Techniken und Methoden der Wissensrepräsentation und -verarbeitung geeignet in die Sprachverarbeitung integriert werden. Wesentliche Teilaufgaben sind dabei

- das Erkennen von Synekdoche und Metonymie, z.B. ein Objekt steht stellvertretend für eines seiner Teile⁴: *das Auto aufpumpen* statt *die Reifen des Autos aufpumpen* [74, 94], sowie
- die Rekonstruktion von metaphorischen Bedeutungen durch Aufdecken der relevanten Analogien wie z.B. in *das wachsende Interesse*, *sich Wissen aneignen* oder *eine Nachricht verdauen* [66].

Als wichtiger Spezialfall der Informationsrekonstruktion spielt die Auflösung von Referenzbeziehungen für sprachverarbeitende Systeme eine zentrale Rolle, wobei zwischen Mechanismen der sprachinternen und -externen Referenz zu unterscheiden ist [43]. Beide Fälle erfordern die (wenigstens partielle) Rekonstruktion des beim Sprecher vorliegenden Referenzsystems durch den Hörer.

Da im Fall der sprachinternen Referenzen ohnehin nur ein partielles Modell der Diskurswelt generiert werden kann, sind hier approximative Lösungen auf der Basis stark vereinfachter Modelle möglich. So werden für die Auflösung pronominaler Referenzen z.B. in

Die Frau₁ holt ihre Familienfotos₂ hervor. Sie₁ trägt sie₂ immer bei sich.

vielfach nur rein syntaktische Kriterien berücksichtigt, obwohl diese nicht in allen Fällen eine sichere Bestimmung des Bezugsnomens ermöglichen:

Das Mädchen₁ bedankt sich bei der Frau₂. Sie₂ hat ihr₁ sehr geholfen.

In solchen Fällen wird Wissen über die Welt benötigt: Üblicherweise richtet sich der Dank an die Helfer. Ähnlich ist die Lage bei Nominalanaphern, bei denen die Referenz ausschließlich über das Hintergrundwissen hergestellt werden kann:

⁴ Andere Beispiele sind: Instanz für Klasse (*einen X100 fahren*), Erzeuger für Erzeugnis (*Goethe lesen*), Gefäß für Inhalt (*einen Teller Suppe essen*) usw.

Herr Maier wurde vom amtierenden Direktor begrüßt.

Der bekennende Anhänger einer Expansionsstrategie wies darauf hin, dass ...

Hier muss der Hörer für eine korrekte Referenzauflösung entweder das Meinungsprofil der beteiligten Personen kennen oder aber bereits wissen, wer die Ansprache gehalten hat. Wegen dieser starken Abhängigkeit vom verfügbaren Hintergrundwissen ergeben sich weitreichende Analogien zu den bereits oben beschriebenen Mechanismen der Informationsrekonstruktion.

Im Gegensatz zur Behandlung sprachinterner Referenzen ist bei der Einbeziehung externer Verweise eine explizite Modellierung unvermeidbar. Je nachdem, ob die Referenz auf eine abstrakte Welt (z.B. Datenbank), eine simulierte und damit noch systemintern generierte Welt oder aber unmittelbar in die reale Welt erfolgt, ergeben sich Aufgaben mit steigendem Schwierigkeitsgrad und wachsenden Anforderungen an die jeweiligen Lösungsansätze (vgl. Tabelle 13.2). Unterschiede im Schwierigkeitsgrad folgen auch aus den dynamischen Eigenschaften der Welt, sowie aus der Verfügbarkeit deiktischer Konstruktionen (sprachliche Zeigegesten: *hier/dort, bisher/später, usw.*)

Aufbauend auf derartigen Verfahren zur Bedeutungsrekonstruktion erschließen sich dann zusätzlich zu den oben genannten noch eine Reihe weiterer Anwendungsbereiche etwa bei der intelligenten Informationssuche in Texten, beim Wissenserwerb aus textuellen Vorlagen [44] oder zur sprachlichen Kommunikation in komplexen multimodalen Interaktionsumgebungen [84]. Weitere Einsatzbereiche für eine stark semantisch fundierte Verarbeitung finden sich im Rahmen der wissensbasierten Übersetzung [88] oder aber auch bei der Realisierung von kommunikativ adäquaten Fehlerdiagnosen in Sprachlernsystemen [82].

Ein gleichfalls sehr wichtiger Berührungspunkt zwischen KI und Sprachverarbeitung ist schließlich mit dem großen Beitrag gegeben, den die KI im Bereich von Beschreibungsformalismen und Problemlösungsverfahren erbracht hat. Hierzu sind insbesondere zu rechnen

- logikbasierte Formalismen zur redundanzarmen Repräsentation von sprachlichem und außersprachlichem Wissen [20] (vgl. Abschnitt 13.4.2),
- effiziente Verfahren zur kombinatorischen Suche [55] sowie
- Architekturprinzipien zur Organisation sehr großer, heterogener Systeme [115] (vgl. Abschnitt 13.5).

13.3 Anwendungen der Sprachtechnologie

Die Palette möglicher Einsatzbereiche für Systeme der Sprachtechnologie ist sicherlich ebenso reichhaltig wie das breite Spektrum sprachlicher Tätigkeiten beim Menschen selbst. Von den in den vorangegangenen Abschnitten vorrangig diskutierten Forschungsansätzen unterscheiden sich die dabei betrachteten Lösungen nicht nur darin, dass sie hohen Anforderungen hinsichtlich der sprachlichen Abdeckung genügen müssen, vielmehr spielen unter dem Anwendungsaspekt auch Kriterien wie Verlässlichkeit, Benutzbarkeit und Integrierbarkeit in etablierte Arbeitsabläufe eine wichtige Rolle. Wegen dieser sehr strengen Forderungen kommen für einen Masseneinsatz vorerst nur vergleichsweise einfache Verfahren in Frage. In Tabelle 13.2 finden sich die entsprechenden Anwendungsfälle vornehmlich in den ersten drei Spalten, da hier noch eine recht einfache Modellierung der Referenzsemantik ausreichend erscheint.

Tabelle 13.2: Aufgabenbereiche der Sprachverarbeitung und die referentielle Einbettung der Äußerungen.

ohne Referenz	textinterne Referenz	externe Referenz				(zusätzlich) erforderliche Basistechnologie
		abstrakte Welten	simulierte Welten	dynamisch	reale Welt	
Rechtschreib- und Grammatikprüfung Spracherkennung Maschinelle Übersetzung	Frage-Antwort-Systeme Textverstehen und -zusammenfassung Maschinelle Übersetzung	Zugriff zu Datenbanken und Expertensystemen	virtuelle Stadtführung [Herzog, 1991] Steuerformular [Algayer et al., 1989]	simulierte Roboter („Klötzchenwelt“) [Winograd, 1972]	autonome Roboter [Moratz et al., 1995] Fahrerassistenzsysteme [Nagel, 1994]	Weltmodell
—	—	partielles Modell einer fiktiven Welt	„vollständiges“ und gesichertes Modell der Welt	partielle und unsichere Kenntnis der Welt	Sensorik (optisch, taktil, akustisch) begriffliche Abstraktion	(zusätzlich) erforderliche Basistechnologie
Lexikon ggf. Grammatik ggf. Transferregeln	referentielle Semantik Anaphernresolution textuelle Deixis	Weltmodell für real existierende Referenzobjekte	räumliche Relationen räumliche Deixis	temporale Relationen temporale Deixis Echtzeitdruck		

Im Hinblick auf die jeweils anzutreffende Kommunikationssituation kann man die unterschiedlichen Anwendungsklassen grob in Hilfsmittel für die Unterstützung der zwischenmenschlichen Kommunikation einerseits und Lösungen für die Mensch-Maschine-Interaktion andererseits einteilen. Obwohl sie sich natürlich strenggenommen auch der zwischenmenschlichen Kommunikation zurechnen ließen, sollen Werkzeuge zur Textproduktion und zum Informationsmanagement hier jedoch gesondert behandelt werden, da sie unter praktischen Gesichtspunkten eine herausragende Bedeutung besitzen.

13.3.1 Werkzeuge für die zwischenmenschliche Kommunikation

Die maschinelle Übersetzung [49] stellt den wohl ältesten tatsächlichen Anwendungsfall der maschinellen Sprachverarbeitung dar und kann damit auf einen umfangreichen technologischen und methodischen Erfahrungsschatz bei der Arbeit mit extrem umfangreichen linguistischen Wissens- und Datenbeständen zurückgreifen. Nachdem sich schon recht bald herausgestellt hat, dass eine qualitativ hochwertige Übersetzung nur in sehr speziellen Einsatzbereichen (z.B. beim Umgang mit weitgehend konventionalisierten Routinemeldungen, wie Wetterberichten) erreicht werden kann, sind heutzutage Lösungen verfügbar, die mit einem gewissen Maß an Vor- und Nachbereitungsaufwand zumindest bei der Übersetzung bestimmter Textsorten einen Effizienzgewinn bzw. Entlastung von Routinearbeiten versprechen. Mit einer Akzeptanz der Resultate kann vor allem dann gerechnet werden, wenn – wie etwa beim on-line Zugriff zu fremdsprachlichen Dokumenten – keine Alternative zur Informationsgewinnung zur Verfügung steht. In professionellen Anwendungen kommen darüberhinaus aber auch zunehmend Werkzeuge zur terminologischen Arbeit (Terminologiedatenbanken) und zur Wiederverwendung von Übersetzungsresultaten (Übersetzungsgedächtnisse) zur Anwendung, so dass der Übersetzer an seinem Arbeitsplatz aus einer ganzen Palette unterschiedlich leistungsfähiger Hilfsmittel auswählen kann.

Alle kommerziell angebotenen Übersetzungssysteme orientieren sich bei der Übertragung recht stark an der sprachlichen Oberflächenform. Von einem tiefergehenden Sprachverständigen kann nach wie vor kaum die Rede sein. Eine Ausnahme bilden hier wohl erste Prototypen zur Übersetzung gesprochener Sprache, die jedoch eher auf sehr kleine Anwendungsdomänen zugeschnitten sind [123].

Wesentliches Ziel bei der Weiterentwicklung der bestehenden Ansätze ist die kontinuierliche Qualitätsverbesserung für die wichtigsten Übersetzungsrichtungen zwischen den „großen“ Sprachen. Sie erfolgt vorrangig durch sukzessive Erweiterung und Verfeinerung der linguistischen Wissensbestände aufgrund einer systematischen Evaluation auf sprachlichen Massendaten. Eine weitere Herausforderung stellt sich mit dem stärker werdenden Wunsch nach Techniken zur schnellen Entwicklung von Übersetzungssystemen (rapid deployment) mit denen vor allem im Hinblick auf militärische Einsatzszenarien flexibel auf einen punktuellen Bedarf an Übersetzungsleistungen gerade für eine der zahlreichen „kleinen“ Sprachen reagiert werden kann.

Neben der Bereitstellung von Übersetzungsleistungen spielt in den modernen multilingualen Gesellschaften die Beherrschung von Fremdsprachen eine zunehmend stärker werdende Rolle. Für die Sprachtechnologie ergibt sich damit auch die Aufgabe, geeignete Hilfsmittel zur Unterstützung des Sprachunterrichts bereitzustellen [102]. Die Spannbreite reicht dabei von der morphologischen Analyse und der Unterstützung eines effizienten Wörterbuch-Zugriffs [86]

bis zur Diagnose und Erklärung von ungrammatischen Konstruktionen in einer beschränkten sprachlichen Domäne [80].

Die Entwicklung von Kommunikationshilfen für Behinderte [76, 87] stellt schließlich ein drittes Aufgabenfeld dar, in dem von der Sprachtechnologie wesentliche Beiträge erwartet werden. Je nach Art der vorliegenden Behinderung müssen hierbei Techniken der Wortformen- und Syntaxanalyse, sowie Komponenten zur Sprachsynthese und -erkennung in Gesamtlösungen für Formulierungs- und Aussprachehilfen, Vorlesegeräte, sowie Hilfsmittel zur visuellen Präsentation bzw. Verschriftlung gesprochener Sprache integriert werden.

13.3.2 Werkzeuge für die Textproduktion

Werkzeuge zur Textverarbeitung, wie Silbentrennung, Tippfehlersuche und -korrektur, sowie die Möglichkeit zum integrierten Wörterbuchzugriff gehören heutzutage zur selbstverständlichen Grundausstattung jedes einschlägigen Softwaresystems zur Textverarbeitung. Daher rücken zunehmend anspruchsvollere Hilfsmittel zur Textproduktion in den Mittelpunkt des Interesses. So wird etwa versucht, den Kreis der detektierbaren Fehler durch den Übergang von Techniken der Wortformenverarbeitung hin zu einer Satzstrukturanalyse beträchtlich zu erweitern [114, 119]. Darauf aufbauend können dann auch Hilfsmittel zur Überprüfung allgemeiner Aspekte der Grammatikalität und der stilistischen Wohlgeformtheit realisiert werden [111].

Zur Effektivierung der Verschriftlung spielen Diktiersysteme eine zunehmende Rolle [108]. Auch hier wirkt sich eine grobe thematische Spezialisierung (z.B. auf medizinische Befundungstexte oder juristische Gutachten) sehr vorteilhaft auf die Leistungsfähigkeit eines solchen Systems aus.

Insbesondere in multilingualen Kontexten werden große Erwartungen in Systeme zur automatischen Textgenerierung gesetzt, die in der Lage sind, den gewünschten Inhalt ausgehend von einer gemeinsamen formalen Repräsentation in unterschiedlichen Sprachen auszuformulieren [79]. Gerade bei der routinemäßigen Produktion von mehrsprachigen Wartungs- und Bedienungsunterlagen lassen sich auf diese Weise erhebliche Rationalisierungseffekte erzielen, die vor allem einer schnellen Verfügbarkeit der Dokumente zugute kommen sollen und somit auch eine attraktive Alternative zur Übersetzung darstellen.

13.3.3 Werkzeuge für das Informationsmanagement

Vor dem Hintergrund des explosionsartig wachsenden Volumens an unmittelbar zugänglicher Textinformation gewinnen Verfahren zum effizienten Umgang mit dieser Informationsflut zunehmend an Bedeutung. Während traditionell Techniken der systematischen Informationsrecherche [7] nur in wenigen spezialisierten Einrichtungen zum Einsatz kommen konnten, stehen sie in Form von Internetdienstleistungen nunmehr praktisch jedermann zur Verfügung. Auch hier ist in den letzten Jahren eine Reihe neuer Aufgabenstellungen mit neuen Verarbeitungsanforderungen hinzugekommen:

Informationsklassifikation und -filterung: Textdokumente (meist HTML-Seiten oder E-Mails) werden in eine bestimmte Anzahl, vom Nutzer vorzugebender Kategorien vorsortiert bzw. auf Wunsch auch ganz ausgeblendet [70]. Dies kann sowohl aufgrund von inhaltlichen Kriterien, aber auch auf der Basis von subjektiven Werturteilen erfolgen [91].

Zum Einsatz kommen vor allem Techniken des Maschinellen Lernens und der Mustererkennung, so dass das System auf beispielhaften Zuordnungen trainiert werden kann und durch fortlaufende Beobachtung sogar eine dynamische Anpassung an Änderungen im Nutzerverhalten möglich wird.

Informationsstrukturierung: Eine vorgegebene Menge von Textdokumenten (z.B. das Ergebnis einer Suchanfrage) soll nach inhaltlichen Kriterien (in der Regel thematische Ähnlichkeit) strukturiert werden [31].

Informationsextraktion: Hierunter fallen eine Reihe von Techniken zur Ermittlung spezieller Informationen aus Textdokumenten. Ausgangspunkt waren hierbei Verfahren, die auf verhältnismäßig reichhaltige Informationsstrukturen (z.B. über Terroranschläge oder Geschäftsabschlüsse) abzielten [30, 68]. Einsatzmöglichkeiten für derartige Verfahren gibt es in zahlreichen Gebieten, z.B. bei der Verwaltung von Patientendaten, Versicherungsanträgen oder Wartungs- und Reparaturberichten. Für all diese Einsatzfälle ist typisch, dass wegen des massenhaften Anfalls an Textdokumenten eine manuelle Bearbeitung nicht mehr in Frage kommt.

In den letzten Jahren hat sich das Interesse eher generischen Aufgaben zugewandt, die sich als Basisbausteine für komplexere Informationsextraktionsaufgaben nutzen lassen. Hierzu gehört die Detektion und Klassifikation von Eigennamen [109], die Ermittlung von Referenzen im Text, das Verfolgen von Entitäten, die Disambiguierung von Wortbedeutungen, die Ermittlung von semantischen Relationen und Prädikats-Argument-Strukturen [34, 42].

Textzusammenfassung: Wie die maschinelle Übersetzung auch, kann das Zusammenfassen eines Textes ausgehend von einem sehr unterschiedlichen Niveau des Textverständnisses erfolgen. Da eine tiefgehende inhaltliche Analyse unrestrictiver Texte noch weitgehend illusorisch ist, beschränken sich praktikable Verfahren auf die Auswahl möglichst aussagekräftiger Sätze aus dem Originaltext. Hierbei lässt sich die Qualität der Auswahl beträchtlich steigern, wenn dabei die Argumentationsstruktur des Textes angemessen berücksichtigt wird [73, 100]. Als erster Schritt auf dem Weg zu einer tatsächlichen Textzusammenfassung, wird auch die Kompression von Sätzen untersucht [61].

Domänenunabhängige Fragebeantwortung: Im Gegensatz zur Dokumentenrecherche besteht die Aufgabe hierbei darin, innerhalb einer großen Dokumentensammlung dasjenige Teilstück eines Dokumentes zu identifizieren, das die Antwort auf die gegebene Frage enthält. Betrachtet werden nicht nur faktenbezogene Anfragen (Wer? Was? Wann?) sondern auch solche nach komplexeren Ereignisbeschreibungen bzw. Kausalbeziehungen (Wie? Warum?).

Wesentlich für den Erfolg derartiger Anwendungen scheint zu sein, inwieweit es gelingt, die Textanalyse nahtlos mit anderen verfügbaren Wissensquellen zu verzahnen. Hierzu zählen beispielsweise Metainformationen über die Dokumente [97] oder aber die Vernetzung bzw. das Navigationsverhalten der Nutzer in Hypertextsammlungen. Im Falle der domänenunabhängigen Fragebeantwortung brachte bereits die Einbeziehung von Inferenztechniken, die auf einer sehr einfachen, automatisch aus natürlichsprachlichen Definitionen extrahierten Wissensbasis arbeiten, eine überraschend deutliche Steigerung der Antwortqualität auf ein Niveau, das weit über die Leistungsfähigkeit der anderen an der Evaluation beteiligten Systeme hinausreicht [45].

13.3.4 Mensch-Maschine-Kommunikation

Einsatzreife Anwendungen der Sprachtechnologie zur Mensch-Maschine-Kommunikation liegen bisher vor allem im Bereich der natürlichsprachlichen Anfrage an relationale Datenbanken vor [6]. Dieser Erfolg ist insbesondere der Tatsache zuzurechnen, dass mit dem Bezug auf das relationale Datenbankmodell eine relativ einfache Referenzsemantik vorliegt (vgl. Tabelle 13.2). Da eine tastaturbasierte Eingabe jedoch stets in unmittelbarer Konkurrenz zu den intuitiv bedienbaren grafischen Benutzungsschnittstellen steht, konnten sich natürlichsprachliche Systeme entgegen den ursprünglichen Erwartungen im Massenmarkt nicht durchsetzen. Eine Trendwende zeichnet sich hier allerdings mit der inzwischen erreichten Einsatzreife von Dialogsystemen für gesprochene Sprache ab [12, 122]. Gerade im Telefoniebereich, in dem alternative Kommunikationskanäle ohnehin nur eine untergeordnete Rolle spielen, eröffnen sich damit attraktive Perspektiven für eine Vielzahl ganz neuartiger Dienstleistungen.

13.4 Modelle und Verfahren zur Sprachverarbeitung

13.4.1 Strukturbeschreibungen

Es gilt als Konsens bei der Verarbeitung natürlicher Sprache, dass sich die gewünschten Verarbeitungsresultate nur in wenigen Spezialfällen in einem unmittelbaren Abbildungsschritt aus den jeweils gegebenen Ausgangsdaten ableiten lassen. Angesichts der kombinatorischen Vielfalt und hochgradigen inneren Komplexität sprachlicher Ausdrucksmittel müssen oftmals mehrstufige Verarbeitungsarchitekturen konzipiert werden, in denen unterschiedliche Repräsentationsformate sukzessive ineinander umgewandelt werden. Hierfür steht eine Vielzahl von Datenstrukturen zur Verfügung, mit denen sequentielle, hierarchische oder gar vernetzte Strukturen in sprachlichen Äußerungen modelliert werden können. Welche dieser Beziehungen relevante Information zur Lösung eines gegebenen Verarbeitungsproblems beitragen kann, muss in der Phase des Systementwurfs geklärt werden und kann bestenfalls in einer Gesamtevaluation unter vergleichbaren Bedingungen überprüft werden.

Grundlegendes Strukturierungsprinzip natürlicher Sprache ist die zeitliche Abfolge von Einheiten auf den unterschiedlichen Repräsentationsebenen. Für die phonetische Ebene sind dies etwa Sequenzen von (kontinuierlichen) Artikulationsbewegungen, und ihren (diskret wahrgenommenen) lautlichen Korrelaten. Eine Beschreibung sprachlicher Beobachtungsdaten durch Folgen von diskreten Phonsymbolen ist daher bereits mit einer wichtigen Abstraktion verbunden. Sie stellt den subjektiven Wahrnehmungsaspekt in den Vordergrund und vernachlässigt den asynchronen Charakter der menschlichen Artikulation. Das Problem tritt in anderer Form wieder auf, wenn auch prosodische Merkmale, wie Akzent, Rhythmus und (Satz-)Melodie, mit in die Verarbeitung einbezogen werden sollen. Diese Signalbestandteile sind inhärent suprasegmental und erfordern daher eine entsprechende Erweiterung der Repräsentation durch nichtlautliche Kategorien oder zusätzliche Beschreibungsebenen. Auf einer höheren Granularitätsebene gruppieren sich Phone zu Silben, als den elementaren Artikulationseinheiten einer Sprache, so dass sich bereits auf der phonetischen Ebene erste Ansätze für eine hierarchische Strukturierung ergeben. Sie kann aber für viele praktische Verarbeitungsaufgaben vernachlässigt werden. So greifen etwa Erkenner für gesprochene Sprache zumeist auf Wortmodelle zurück, die durch Verkettung von Phonmodellen entstehen. Hierbei erfolgt eine unmittelbare Abbildung aus einer Sequenz

von Signalbeobachtungen in eine Sequenz von Wortformen, die im Falle von Diktieranwendungen auch die gewünschte Zielstruktur darstellt.

Derartige Ganzwortmodelle ignorieren auch die Ebene der Morphologie, auf der untersucht wird, wie sich Wortformen aus ihren elementaren Bausteinen (Wurzeln, Flexionsendungen, Präfixen, Suffixen usw.) zusammensetzen. Sie sind daher vor allem für morphologisch relativ arme Sprachen geeignet, die kaum spontane Neubildungen zulassen.

Auch auf der morphologischen Ebene wird die lineare Abfolge durch eine hierarchische Struktur überlagert, die hier jedoch weitaus stärker in den Vordergrund tritt, da sie die syntaktischen Eigenschaften der Wortformen, sowie deren Bedeutung wesentlich beeinflussen kann. Damit bieten sich Baumstrukturen als geeignetes Beschreibungsmittel an, die hier durch eine entsprechende Klammerung illustriert sind:

Wurzel	(lieb)
Flexion	((lieb)t)en
Präfigierung	(ver(lieb))
Suffixierung	((lieb)lich)
Zirkumfigierung	(ge(lieb)t)

Zur Vereinfachung wird jedoch oftmals angenommen, dass sich zumindest die Zirkumfigierung auch auf eine Kombination von Prä- und Suffixierung zurückführen lässt, wobei Abstriche an der Modelladäquatheit im Interesse einer einfachen und effizient verarbeitbaren Repräsentation in Kauf genommen werden (müssen). Noch deutlicher wird dies im Fall der Infigierung, die sich einer systematischen und dennoch strikt hierarchischen Modellierung vollständig entzieht.

Wurzel	Präfigierung	Flexion
(arbeit)	(ab(arbeit))	((ab(arbeit))en) (ab[zu(arbeit))en]

Auch hier muss wieder mit approximativen Modellen gearbeitet werden.

Morphologische Prozesse unterscheiden sich in solche, die nur eine Formanpassung an die Bedingungen des syntaktischen Kontexts vornehmen (Flexion) und andere, die auch die Wortart bzw. die semantischen Eigenschaften verändern. Dabei sind Präfixe, Suffixe und Flexionsendungen morphologische Bausteine, die nur in Verbindung mit einer semantisch eigenständigen Einheit, der Wurzel, verwendet werden können.

Ein weiterer Wortbildungsprozess der (mehrere) eigenständige Formen miteinander kombiniert und der gravierende Auswirkungen auf die Bedeutung hat, ist die Kompositumsbildung, die gerade im Deutschen sehr produktiv ist und zahlreiche spontane Neubildungen ermöglicht. Auch hier bilden sich hierarchische Strukturen heraus, an denen sich die Bedeutungsermittlung orientieren muss:

((Eisen bahn) strecke)	(Güter (bahn hof))
-------------------------------	---------------------------

Für viele praktische Anwendungen lassen sich morphologische Strukturen bei der Sprachverarbeitung ohne gravierenden Verlust an problemrelevanter Information mit Hilfe von einfachen Sequenzen approximieren. In eingeschränkten Domänen kann zusätzlich noch angenommen werden, dass der Wortschatz ebenfalls begrenzt ist und daher sogar ein Vollformenlexikon ohne Berücksichtigung der internen Wortstruktur verwendet werden kann.

Derartige Vereinfachungen sind im Bereich der Syntax nicht mehr möglich, da man hier davon ausgehen muss, dass

- auch rekursive Einbettungsstrukturen auftreten und
- es selbst in beschränkten Anwendungsdomänen keine obere Schranke für die Menge der wohlgeformten Sätze gibt.

Für die Inhaltserschließung wird zudem das Kompositionalsprinzip angenommen, nachdem sich die Bedeutung eines Satzes kompositional aus der Bedeutung seiner Bestandteile ergibt [40], so dass die Kenntnis der hierarchischen Beziehungen im Satz Voraussetzung für seine semantische Analyse ist.

Zur Repräsentation hierarchischer Beziehungen in Sätzen haben sich zwei alternative Herangehensweisen herausgebildet. Sie zeichnen sich jeweils durch spezifische Vor- und Nachteile aus:

- *Konstituentenstrukturen* stellen den konfigurationalen Aufbau eines Satzes aus seinen (strukturell weniger komplexen) Bestandteilen in den Mittelpunkt, wohingegen
- *Abhängigkeitsstrukturen* (Dependenzstrukturen) sich eher an den funktionalen Beziehungen zwischen einzelnen Wortformen oder Teilbäumen eines Satzes orientieren.

Konstituentenstrukturen sind besonders gut geeignet, um bereichsabhängige Information darzustellen, wie sie etwa bei der Modellierung der Informationsstruktur (s.u.) auftreten. Typinformation kann dann ganzen Konstituenten und nicht nur einzelnen Wortformen zugewiesen werden. Durch die Vergabe von Namen für die Konstituenten eines Satzes, stehen zudem recht effiziente Lösungsverfahren für die Strukturanalyse zur Verfügung (vgl. Abschnitt 13.4.3).

Dependenzbeziehungen hingegen sind im Grunde anordnungsfrei und daher besser geeignet, die verschiedenen Linearisierungsvarianten in Sprachen mit freier Anordnung der Satzglieder zu berücksichtigen. Dies führt auch dazu, dass sich nicht-hierarchische Strukturbereihungen (Nichtprojektivitäten) ohne zusätzliche Operatoren adäquat beschreiben lassen.

Dependenzbeziehungen erfordern keinerlei Annahmen über nicht direkt beobachtbare Strukturlemente und deren Typ. In Verbindung mit Nichtprojektivitäten ruft dies zwar Effizienzprobleme bei der Analyse hervor, andererseits stehen dadurch aber auch alternative Lösungsverfahren zur Verfügung (vgl. Abschnitt 13.4.3, 13.4.4 bzw. 13.4.4). Für Dependenzbeziehungen gestaltet sich auch die Abbildung von den syntaktischen Valenzen (bei Verben etwa Subjekt, direktes Objekt usw.) auf die Argumente der korrespondierenden semantischen Struktur (Agens, Patiens usw.) erfreulich einfach. Hinzu kommt noch, dass sich Dependenzstrukturen zunehmender Popularität bei der Evaluation von syntaktischen Analyseverfahren erfreuen, da sie die Ähnlichkeit verschiedener Strukturvarianten weitaus besser widerspiegeln als Konstituentenstrukturen [69].

Noch deutlicher als bereits bei der Diskussion morphologischer Phänomene lässt sich im Bereich der Syntax beobachten, dass Strukturbeschreibungen immer nur bestimmte Aspekte eines linguistischen Phänomens widerspiegeln. Im Rahmen des Systementwurfs ist daher wieder

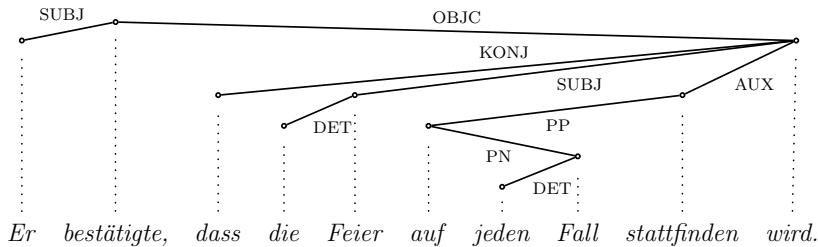
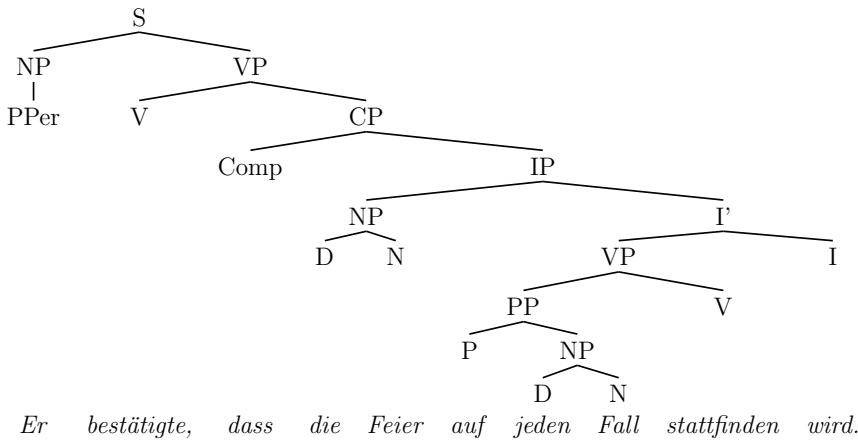


Abbildung 13.1: Konstituenten- und Dependenzstruktur für einen Satz (Abhängigkeitsbeziehungen sind von unten nach oben gerichtet).

zu entscheiden, ob eine bestimmte Repräsentation die Anforderungen der jeweiligen Aufgabe hinreichend gut erfüllt. Baumbanken, in denen für Forschungs- und Entwicklungszwecke authentisches Satzmaterial gesammelt und mit Strukturbeschreibungen annotiert wird, versuchen daher oftmals diese unterschiedlichen Aspekte angemessen zu berücksichtigen. Ein Beispiel hierfür ist das NEGRA-Korpus [107], in dem die Konstituentenstruktur zusätzlich durch Dependenzinformation angereichert ist, indem der Kopf einer Konstituente (von dem die anderen Bestandteile der Konstituente abhängig sind) ausgezeichnet wird.

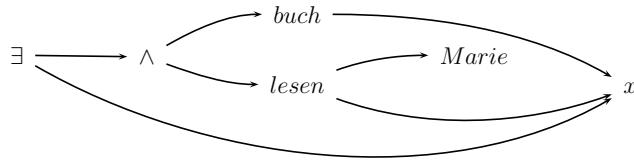
Zur Repräsentation der kompositionalen Bedeutung einer Äußerung wird zumeist auf das Beschreibungsinventar der formalen Logik zurückgegriffen. Damit ergeben sich auch hier strukturelle Beziehungen, die sich aber allein auf der Basis hierarchischer Einbettung nicht mehr angemessen darstellen lassen. Ursache dafür sind Koreferenzen zwischen den Individuenvariablen eines komplexen Ausdrucks:

Marie liest ein Buch.

$\exists x . \text{buch}(x) \wedge \text{lesen}(\text{Marie}, x)$

1

Eine adäquate Darstellung erfordert hier zwangsläufig den Rückgriff auf gerichtete azyklische Graphstrukturen:



Ähnlich gestaltet sich die Behandlung sprachlicher Koreferenzen. Derartige, oftmals Satzgrenzen überschreitende Bezugnahmen, können mit den Mitteln der Diskursrepräsentationstheorie (DRT) modelliert werden [54]. Sprachliche Koreferenz wird auch hier wieder auf Koreferenz zwischen den Variablen einer logischen Form, der Diskursrepräsentationsstruktur (DRS), abgebildet. Sie erfasst die Semantik einer Satzfolge durch Integration des kontextbezogenen Potentials von Einzelaussagen. Eine DRS besteht dabei immer aus einer Menge von Diskursreferenten und einer Menge von Bedingungen über diesen.

x,y
frau(x)
familienfotos(y)
hervorholen(x,y)
zugehörig(y,x)

Die Frau holt ihre Familienfotos hervor.

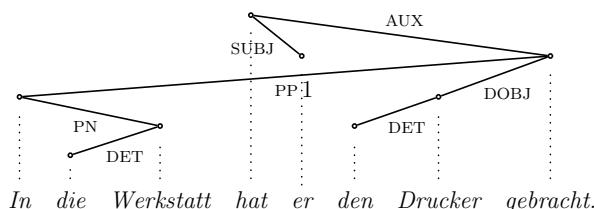
u,v
x = u
y = v
bei_sich_tragen(u,v)

Sie trägt sie immer bei sich.

Über die Modellierung des Referenzpotentials hinaus erfasst die DRT das satzinterne Zusammenspiel von Quantoren und referentiellen Bezügen und beschreibt auf der Basis der internen Strukturierung einer DRS auch die Zugänglichkeit von Diskursreferenten für eine referentielle Bezugnahme, z.B. mit Hilfe von Pronomen.

Stärker an der pragmatischen Ebene orientiert ist die *Informationsstruktur* eines Satzes. Hier unterscheidet man bereits vorerwähnte Information (Thema, Topik) von neu hinzukommender Information (Rhema, Fokus), wobei für die gegebene Information üblicherweise die besonders prominente, satzinitiale Position reserviert ist. Wie stark dieser Normalfall der Informationsstruktur im allgemeinen Sprachbewusstsein präsent ist, zeigt sich etwa am Beginn deutscher Märchen, wo eine Bezugnahme auf gegebene Information ja noch nicht möglich ist und daher ein semantisch leeres Füllpronomen gewählt werden muss: *Es war einmal ...*

Abweichend von der unmarkierten Reihenfolge kann in der satzinitialen Position jedoch auch solches syntaktisches Material untergebracht werden, das durch diese prominente Position besonders hervorgehoben werden soll, auch wenn dadurch die normalen Prinzipien der Konstituentenreihung im Satz durchbrochen werden. Solche Anordnungsvarianten haben dann in vielen Fällen eine systematische Nichtprojektivität zur Folge:



Im Deutschen bildet sich die Informationsstruktur unmittelbar auf die *topologischen Felder* eines Satzes ab, wobei zwischen Vorfeld-, Mittelfeld- und Nachfeldpositionen, sowie der linken und rechten Satzklammer unterschieden wird:

Vorfeld	linke Klammer	Mittelfeld	rechte Klammer	Nachfeld
<i>In die Werkstatt</i>	<i>hat</i>	<i>er den Drucker</i>	<i>gebracht.</i>	
<i>Er</i>	<i>packt</i>	<i>den Drucker</i>	<i>aus,</i>	<i>der in der Werkstatt war.</i>

Da sich die topologischen Felder im Satz vergleichsweise leicht identifizieren lassen, eignen sie sich hervorragend zur Vorstrukturierung des Suchraums für die syntaktische Analyse. Zudem stellen sie eine Repräsentationsebene dar, die sich gut für die Modellierung von Anordnungsphänomenen eignet [18].

Angesichts der Tatsache, dass es mit den topologischen Feldern und der Informationsstruktur, neben der eigentlichen syntaktischen Struktur weitere Beschreibungsebenen gibt, die ebenfalls zwischen der Form eines Satzes und seinem semantischen und pragmatischen Gehalt vermitteln, ergibt sich die Notwendigkeit, verschiedene strukturelle Aspekte zu repräsentieren und systematisch aufeinander zu beziehen. Ein Modellierungsansatz, der dies leistet, findet sich in z.B. in [32].

Für zahlreiche praktische Sprachverarbeitungsaufgaben, insbesondere im Umfeld der Dialogsysteme und der maschinellen Übersetzung, reicht eine Beschreibung der syntaktischen und semantischen Aspekte natürlichsprachlicher Äußerungen allein nicht aus, um qualitativ hochwertige Verarbeitungsresultate zu erhalten. Zusätzlich ist auch die Erfassung der Sprecherintentionen z.B. in Form von Sprechakten wesentlich. Im Kontext des Gesprächsdolmetschens sprechen Alexandersson et. al. [1] dann auch von Dialogakten.

Dialogakte bilden nicht die wörtliche Bedeutung, sondern den vom Sprecher intendierten kommunikativen Gehalt ab (Begrüßung, Zustimmung, Ablehnung). In einigen Fällen können Dialogakte auch durch den propositionalen Gehalt der jeweiligen Äußerung parametrisiert werden, etwa im Falle eines Vorschlags. Über die Modellierung von Abfolgebeschränkungen lassen sich Bedingungen für die Kohärenz eines Dialogs erfassen und in den Entscheidungsprozess über eine angemessene Systemreaktion einbringen.

Während Kohärenz den Textzusammenhalt auf der inhaltlichen Ebene beschreibt, orientiert sich die Kohäsion eines Textes stärker an den sprachlichen Ausdrucksmitteln für den Zusammenhang zwischen den elementaren Textbausteinen. Diese Zusammenhänge werden im Rahmen der Rhetorical Structure Theory (RST, Mann und Thompson [71]) mit Hilfe von rhetorischen Relationen beschrieben, die die funktionalen Beziehungen zwischen benachbarten, hierarchisch gegliederten Textelementen modellieren (siehe Abb. 13.2).

Die Repräsentation unterscheidet systematisch zwischen Nukleus (im Bild jeweils durch die Pfeilspitze markiert) und Satellit einer Relation, wobei die im Satelliten kodierte Information entfallen kann, ohne die Textverständlichkeit entscheidend zu beeinträchtigen. Somit stellt die rhetorische Struktur wichtige Grundlagen für die Textplanung (vgl. Abschnitt 13.4.6) und die Zusammenfassung von Texten bereit.

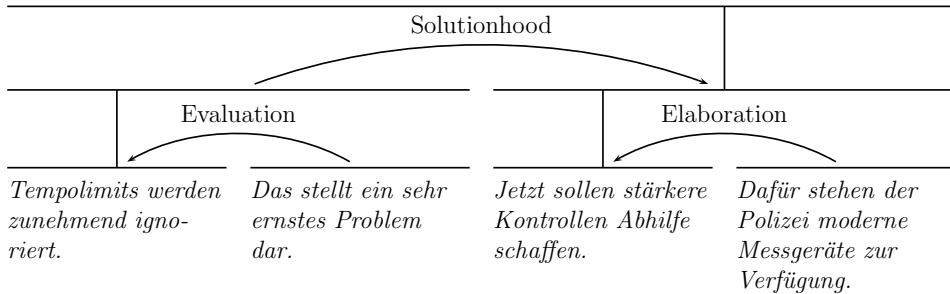


Abbildung 13.2: Rhetorisch Relationen zwischen Textelementen.

13.4.2 Wissensrepräsentation

Endliche Automaten

Solange sich die Anforderungen einer Verarbeitungsaufgabe auf die Modellierung von sequentiellen Beziehungen in der Sprache beschränken, können endliche Automaten oder deren probabilistische Varianten bzw. Erweiterungen (vgl. Abschnitt 13.4.4) zum Einsatz kommen. Da echt rekursive Einbettungsstrukturen auch im Bereich der Syntax nur in wenigen Fällen eine entscheidende Rolle spielen, erlauben solche Modelle eine gute Approximation der empirisch nachweisbaren Symbolfolgen.

Endliche Automaten gelten als effizient verarbeitbar, da die kritischen Fälle von Nichtdeterminismus in der natürlichen Sprache praktisch nicht vorkommen. Darüberhinaus existieren wohlfundierte Algorithmen zur Komposition komplexer Automaten aus einfacheren durch die Anwendung algebraischer Operatoren zur Verkettung, Vereinigung, Schnittbildung, Differenzbildung usw. Dies ermöglicht eine modulare Gestaltung der Wissensbasis und unterstützt die Wissensakquisition in optimaler Weise.

Wegen dieser Vorteile werden endliche Automaten in vielen Bereichen der Sprachverarbeitung eingesetzt, z.B. zur Vorverarbeitung und Normalisierung von sprachlichen Daten, zur Korpusrecherche, zur Modellierung morphologischer Bildungen, zur Extraktion standardisierter aber produktiver Fügungen (Orts-, Datums- und Zeitangaben), oder aber zur Modellierung zulässiger Wortfolgen und Dialogakten in Dialogsystemen für gesprochene Sprache. Darüberhinaus kommen zur Modellierung phonologischer Prozesse auch endliche Transducer [64] zum Einsatz, so z.B.

- bei der Rückführung orthografischer Varianten eines Morphems auf eine kanonische Wörterbuchrepräsentation: *ich handl+e* aber *er handel+t*, *wir handel+n* aber *wir sprech+en*
- bei der Generierung von Aussprachevarianten eines Morphems: *der Tag* ([ta:k]) aber *die Tage* ([ta:gə]), *lustig* ([löstic]) aber *lustige* ([lustigə])

Phrasenstrukturgrammatiken

Müssen über die rein sequentiellen Beziehungen hinaus auch hierarchische (Phrasen-)Strukturen berücksichtigt werden, können *kontextfreie Grammatiken* oder Varianten davon verwendet werden. Die Regelstruktur der Grammatik bildet sich dabei unmittelbar in eine hier-

archische Strukturbeschreibung ab. So lässt sich z.B. die Strukturbeschreibung des Satzes in Abbildung 13.1 durch folgende kontextfreie Ersetzungsregeln erzeugen:

S	\Rightarrow	NP VP
NP	\Rightarrow	D N
NP	\Rightarrow	PPer
VP	\Rightarrow	V CP
CP	\Rightarrow	Comp IP
IP	\Rightarrow	NP I'
	...	
PPer	\Rightarrow	er
V	\Rightarrow	bestätigte
Comp	\Rightarrow	dass
D	\Rightarrow	die
	...	

Dabei bildet das Vokabular einer Sprache, die Menge der *Terminalsymbole*, während alle übrigen Symbole Metazeichen zur Strukturbeschreibung darstellen und als *Nichtterminalsymbole* bezeichnet werden. Jedes Nichtterminalsymbol muss in mindestens einer Regel auf der linken Regelseite auftreten.

Wie die endlichen Automaten auch sind kontextfreie Grammatiken deklarative Formalismen, d.h. sie können gleichermaßen zur Generierung, wie zur Strukturanalyse sprachlicher Ausdrücke verwendet werden. Dabei wird im Falle der Generierung das Startsymbol der Grammatik (in unserem Fall das Symbol S) durch eine Regel, die das Symbol auf der linken Seite enthält, expandiert, d.h. es wird durch die Symbole auf der rechten Regelseite ersetzt. Enthält die so erzeugte Symbolfolge noch Nichtterminalsymbole, wird der Prozess rekursiv fortgeführt, bis keine weiteren Symbole mehr expandiert werden können.

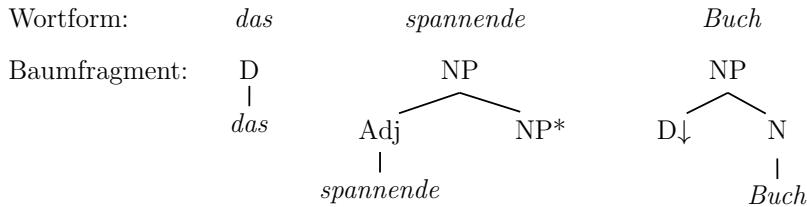
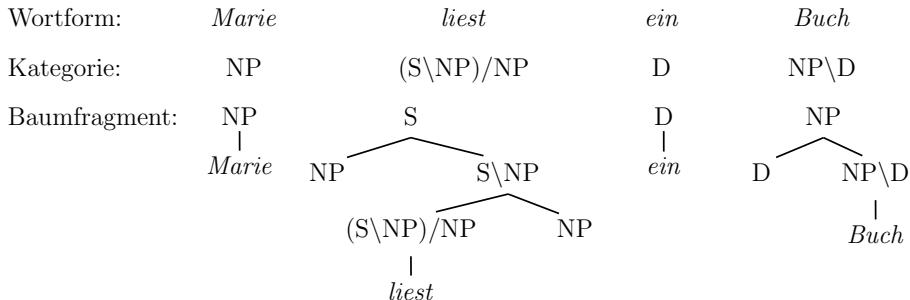
Bei der Strukturanalyse hingegen, werden die Ersetzungsregeln beginnend mit der rechten Seite auf die sprachlichen Ausdrücke angewendet, bis nur noch das Startsymbol der Grammatik verbleibt (vgl. Abschnitt 13.4.3). Die Ableitungsgeschichte stellt dann einen Phrasenstrukturbaum für die betreffende Äußerung dar.

Neben den kontextfreien Grammatiken existieren mehrere alternative Formalismen mit denen ebenfalls Phrasenstrukturen erzeugt werden können, z.B.

- die Kategorialgrammatik (CG, Bar-Hillel [10]), die bereits im Wörterbuch mit ganzen Baumfragmenten als komplexen lexikalischen Beschreibungen arbeitet, aus denen sich letztendlich auch die Satzstruktur zusammensetzt (s. Abb. 13.4).
- die Baumadjunktionsgrammatik (Tree-Adjoining Grammar, TAG, Joshi [52]), die in ihrer lexikalierten Form ebenfalls mit komplexen Baumfragmenten arbeitet, aber zusätzlich zu dem normalen Substitutionsmechanismus (Knoten mit der Markierung \downarrow) auch über Adjunktionsregeln (Knoten mit der Markierung $*$) verfügt, mit denen nicht-obligate Elemente (Adjunkte) in eine Baumstruktur eingefügt werden können (s. Abb. 13.3).

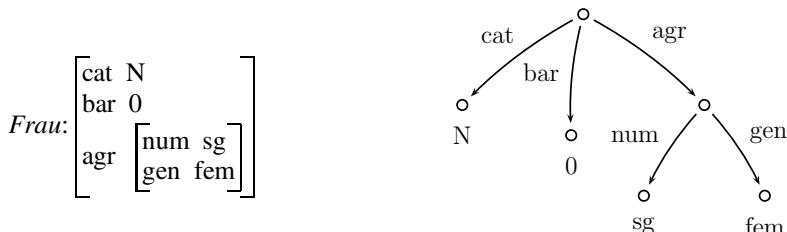
Unifikationsgrammatiken

Es hat sich herausgestellt, dass die atomaren Nichtterminalsymbole einer kontextfreien Grammatik ein zu grobes Beschreibungsinventar für die Syntax natürlichsprachlicher Äußerungen

**Abbildung 13.3:** Baumadjunktionsgrammatik.**Abbildung 13.4:** Kategorialgrammatik.

darstellen. Um zu verhindern, dass bei der Generierung auch ungrammatische Sätze gebildet werden bzw. bei der Analyse sinnlose strukturelle Interpretationen auftreten, ist man gezwungen, feinere Unterscheidungen etwa zu den morphosyntaktischen Merkmalen Person, Genus, Kasus und Numerus vorzunehmen. Auf der Grundlage atomarer Nichtterminale (wie in der kontextfreien Grammatik) führt dies jedoch zwangsläufig zu einem Verlust an generalisierbarer Information, der sich letztendlich in einem exponentiellen Anwachsen der Regelmenge bemerkbar macht.

Einen Ausweg bieten hier die Unifikationsgrammatiken [105], bei denen die atomaren Symbole der Grammatik durch komplexe Merkmalsstrukturen ersetzt werden. Dabei handelt es sich um Mengen von Attribut-Wert-Paaren, die eine im mathematischen Sinne funktionale Abbildung aus der Menge der Attribute in die Menge der Werte realisieren. Werte können wiederum komplexe Symbole, also ebenfalls Merkmalsstrukturen sein. Die resultierenden Beschreibungen lassen sich als gerichtete azyklische Graphen mit den Attributen als Kantenbeschriftung und den zugehörigen Werten als Terminalsymbole veranschaulichen, z.B. für den Lexikoneintrag der Wortform *Frau*:



Da Merkmalsstrukturen rekursiv eingebettet werden können, stehen auch beliebige rekursive Datenstrukturen (inbesondere Listen) für die linguistische Modellierung zur Verfügung.

Der Grammatikautor ist völlig frei in der Wahl der benötigten Attribute. Dementsprechend bieten Unifikationsgrammatiken eine hohe Flexibilität bei der Umsetzung linguistischer Intuition. So werden etwa in der Lexical Functional Grammar (LFG) spezielle Attribute für die grammatischen Funktionen der Satzglieder (Subjekt, direktes Objekt usw.) reserviert, wodurch zusätzlich zur Phrasenstrukturgliederung auch eine funktionale Sicht auf die Satzstruktur modelliert werden kann [16].

Die zentrale Operation zur Arbeit mit Merkmalsstrukturen ist die Unifikation, durch die zwei Merkmalsstrukturen auf Verträglichkeit geprüft werden, und die im Erfolgsfall die Vereinigungsmenge der in den ursprünglichen Merkmalsstrukturen enthaltenen Information berechnet. Dadurch vereinigt Unifikation drei Berechnungsaspekte in sich, die für die Arbeit mit komplexen Strukturbeschreibungen essentiell sind:

- Sie kann als *Test* für das Vorhandensein bestimmter Eigenschaften an den Knoten einer Strukturbeschreibung dienen.
- Über sie wird der selektive *Zugriff* auf bestimmte Teile einer Merkmalsstruktur ermöglicht.
- Sie kann zur *Konstruktion* neuer Merkmalsstrukturen verwendet werden.

Attribute in Merkmalsstrukturen können *underspezifiziert* sein, wie etwa das Kasusmerkmal in der oben angegebenen Merkmalsstruktur für die Wortform *Frau*. Unterspezifizierte Merkmale können im Verlauf der Berechnung durch *Informationsanreicherung* mit einem Wert versehen werden, wenn etwa aufgrund von Kongruenz mit anderen Satzbestandteilen die Wertebelegung für das betreffende Merkmal erschlossen werden kann.

$$\left[\begin{array}{c} \text{agr} \\ \text{gen fem} \\ \text{cas acc} \end{array} \right] \sqcup \left[\begin{array}{c} \text{agr} \\ \text{num sg} \\ \text{gen fem} \end{array} \right] = \left[\begin{array}{c} \text{agr} \\ \text{num sg} \\ \text{gen fem} \\ \text{cas acc} \end{array} \right]$$

die_{acc} *Frau* *die_{acc} Frau_{acc}*

Derartige Wohlgeformtheitsbedingungen lassen sich durch Koreferenz zwischen zwei Attributen beschreiben, auch wenn die konkrete Merkmalsbelegung zum Zeitpunkt der Systementwicklung noch gar nicht bekannt sein kann. Ebenfalls durch Koreferenz wird der Transport von Merkmalen durch den Strukturabaum etwa von den Lexikoneintragungen zur Phrasenebene (Projektion) vermittelt. Eine Regel zur Bildung einer Nominalgruppe aus einem Artikel und einer artikellosen Nominalgruppe könnte dann etwa wie folgt aussehen:

$$\left[\begin{array}{c} \text{cat N} \\ \text{bar 2} \\ \text{agr } \boxed{1} \end{array} \right] \Rightarrow \left[\begin{array}{c} \text{cat D} \\ \text{bar 0} \\ \text{agr } \boxed{1} \end{array} \right] \quad \left[\begin{array}{c} \text{cat N} \\ \text{bar 1} \\ \text{agr } \boxed{1} \end{array} \right]$$

Hier wird durch die Koreferenz (dargestellt durch die Markierung $\boxed{1}$) sowohl die Kongruenz zwischen den beiden Schwesternknoten D^0 und N^1 gefordert, als auch das Unifikationsresultat auf die Phrasenebene (Knoten $N^2 = NP$) projiziert.

Ungeachtet der Tatsache, dass sich die Form derartiger Regeln an das Regelformat kontextfreier Grammatiken anlehnt, kann die resultierende Grammatik durchaus bereits kontextsensitiv sein.

Der Grund hierfür ist die Möglichkeit zur Verwendung rekursiv eingebetteter Merkmalsstrukturen als komplexe Nichtterminalsymbole, wodurch die Symbolmenge potenziell unendlich viele Elemente enthalten kann.

Semantikkonstruktion

Das Potenzial der Unifikation zur Wertzuweisung an unterspezifizierte Attribute lässt sich auch dazu verwenden, komplexe Strukturbeschreibungen zu konstruieren. Anwendungen dafür finden sich vor allem im Bereich der Semantik, wenn etwa die semantische Beschreibungen für die Subjekts-Nominalgruppe und die Verbgruppe eines Satzes zu einer vollständigen Prädikats-Argumentstruktur für den Satz zusammengefügt werden soll. Gleichzeitig wird dem Subjekt der strukturelle Kasus Nominativ zugewiesen:

$$\begin{bmatrix} \text{cat S} \\ \text{bar 1} \\ \text{sem } \boxed{1} \end{bmatrix} \Rightarrow \begin{bmatrix} \text{cat N} \\ \text{bar 2} \\ \text{agr } [\text{cas nom}] \\ \text{sem } \boxed{1} \end{bmatrix} \quad \begin{bmatrix} \text{cat V} \\ \text{bar 2} \\ \text{sem } \boxed{1} \end{bmatrix}$$

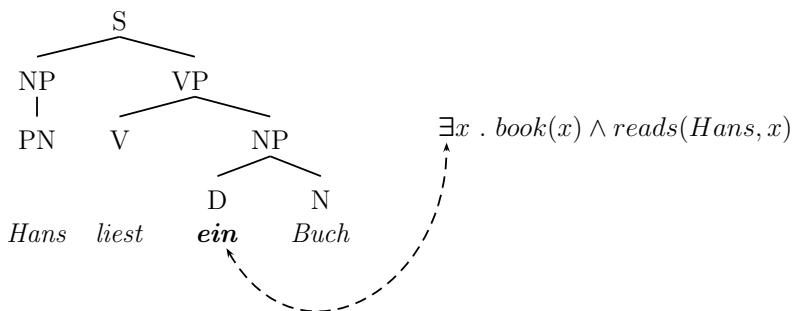
Die in dieser Regel enthaltenen Koreferenzforderungen für die sem-Merkmale sorgen dafür, dass die Teilbeschreibungen z.B. für die Bestandteile des Satzes *Die Firma verkauft Computer*, durch Unifikation zusammengesetzt und auf der Satzebene zur Verfügung gestellt werden:

$$\begin{bmatrix} \text{sem } [\text{ag firma}] \end{bmatrix} \sqcup \begin{bmatrix} \text{sem } [\text{pred verkaufen} \\ \text{theme computer}] \end{bmatrix} = \begin{bmatrix} \text{sem } [\text{pred verkaufen} \\ \text{ag firma} \\ \text{theme computer}] \end{bmatrix}$$

die Firma *verkauft Computer* *Die Firma verkauft Computer.*

Ein solches Vorgehen setzt jedoch voraus, dass die Komponenten der jeweils vorliegenden semantischen Struktur unabhängig von Anordnungsvarianten immer den gleichen strukturellen Positionen im Sysntaxbaum zugewiesen werden. Dies lässt sich etwa durch geeignet zu definierende Bewegungsoperationen erreichen. Alternativ dazu müssten die unterschiedlichen Strukturvarianten in der Grammatik explizit aufgezählt werden. Dies gilt auch für die verschiedenen semantischen Valenzmuster und Formen der Verben (z.B. Aktiv bzw. Passiv)

Etwas schwieriger gestaltet sich die Behandlung von sprachlichen Quantoren (zumeist Determiner), die oftmals tief in die syntaktische Struktur eingebettet sind, in der logischen Bedeutungsrepräsentation aber als äußere Strukturelemente auftreten:



Die hier erforderliche Strukturtransformation kann unter Rückgriff auf den Mechanismus der λ -Konversion, einer speziellen Form der Funktionsapplikation erreicht werden. Als Semantik des unbestimmten Artikels *ein* wird hierzu der Ausdruck

$$\lambda F \lambda G \exists x . F(x) \wedge G(x)$$

und für das Verb *lesen* der Ausdruck

$$\lambda H \lambda w H(\lambda x \text{ reads}(w, x))$$

im Lexikon repräsentiert. Die Reihenfolge, in der diese Ausdrücke zu kombinieren sind, ergibt sich aus einer Typtheorie, die auf Montague [83] zurückgeht. Danach sind Entitäten (*Hans*) von Typ e , und Aussagen vom Typ t (Wahrheitswert). Prädikatskonstanten bekommen daher den Typ $\langle e, t \rangle$, d.h. sie binden einen Ausdruck vom Typ e an sich und ordnen ihm den Typ t zu. Demgegenüber können zweistellige Prädikate (z.B. *liest*) einen der Typen $\langle e, \langle e, t \rangle \rangle$, $\langle e, \langle \langle e, t \rangle, t \rangle \rangle$, $\langle \langle \langle e, t \rangle, t \rangle, \langle e, t \rangle \rangle$ oder $\langle \langle \langle e, t \rangle, t \rangle, \langle \langle e, t \rangle, t \rangle \rangle$ annehmen, d.h. sie binden in zwei Teilschritten jeweils einen Ausdruck vom Typ e (Eigenname) oder $\langle \langle e, t \rangle, t \rangle$ (Nominalgruppe) an sich, um daraus insgesamt einen Ausdruck vom Typ t zu bilden. Ein Determinator ist vom Typ $\langle \langle e, t \rangle, \langle \langle e, t \rangle, t \rangle \rangle$. Damit ergibt sich für den Satz *Hans liest ein Buch*. die Typstruktur in Abb. 13.5.

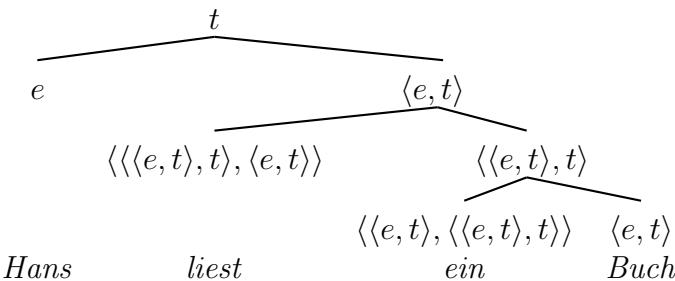


Abbildung 13.5: Typstruktur zu „Hans liest ein Buch“.

Der Aufbau der semantischen Beschreibung erfolgt dann in einer Serie von Funktionsapplikationen, die sich strikt an der syntaktischen Struktur des Satzes orientieren und so das von Frege postulierte Kompositionalsprinzip wahren s. Abb.⁵ 13.6.

Wesentliche Aspekte dieses Abbildungsprozesses lassen sich ebenfalls mit Hilfe der Unifikation von Merkmalsstrukturen implementieren und dank der strikten Kompositionallität in die Regeln einer unifikationsbasierten Grammatik integrieren. Kontextuell bedingte Einflüsse auf die Interpretation von Quantoren sind wiederum Gegenstand einer satzübergreifenden Modellierung mit den Mitteln der Diskursrepräsentationstheorie (vgl. Abschnitt 13.4.1).

Constraint-basierte Grammatiken

Unifikationsgrammatiken gehören wie alle generativen Grammatiken zu den deklarativen Formalismen für die Wissensrepräsentation und sind daher neutral bezüglich der Verarbeitungs-

⁵ In der Tabelle bedeutet ein Ausdruck der Art (*liest*)(*ein Buch*), dass der funktionale Ausdruck für *liest* auf den Ausdruck für *ein Buch* angewendet wird

Satzfragment	partielle semantische Beschreibung
<i>ein</i>	$\lambda F \lambda G \exists x . F(x) \wedge G(x)$
<i>Buch</i>	$\lambda y book(y)$
<i>(ein)(Buch)</i>	$\lambda F (\lambda G \exists x . F(x) \wedge G(x))(\lambda y book(y))$ $\equiv \lambda G \exists x . \lambda y book(y)(x) \wedge G(x)$ $\equiv \lambda G \exists x . book(x) \wedge G(x)$
<i>liest</i>	$\lambda H \lambda u H(\lambda w reads(u, w))$
<i>(liest)(ein Buch)</i>	$\lambda H (\lambda u H(\lambda w reads(u, w))) (\lambda G \exists x . book(x) \wedge G(x))$ $\equiv \lambda u (\lambda G \exists x . book(x) \wedge G(x)) (\lambda w reads(u, w))$ $\equiv \lambda u \exists x . book(x) \wedge reads(u, x)$
<i>Hans</i>	Hans
<i>(liest ein Buch)(Hans)</i>	$\lambda u (\exists x . book(x) \wedge reads(u, x))(Hans)$ $\equiv \exists x . book(x) \wedge reads(Hans, x)$

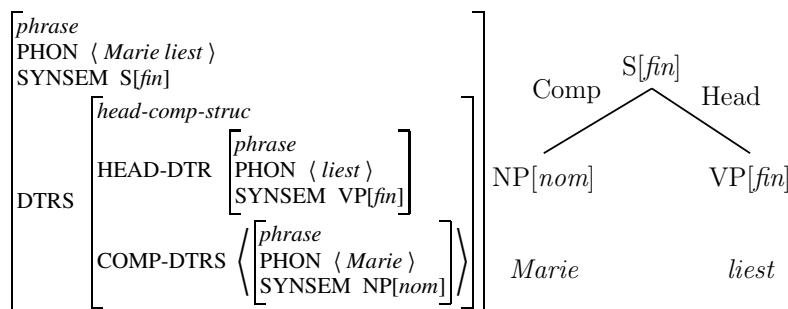
Abbildung 13.6: Semantische Satzbeschreibung durch Funktionsapplikationen.

richtung. Somit lassen sich Unifikationsgrammatiken auch als verallgemeinerte Transducer zwischen beliebigen Strukturbeschreibungen auffassen. Allerdings bildet die durch die Regelstruktur vorgegebene hierarchische Gliederung eines Satzes immer noch das Rückgrat der Modellierung. Das Fehlen einer geeigneten Regel in der Grammatik hat unter diesen Umständen stets ein systematisches Scheitern der gesamten Verarbeitung zur Folge.

Einen Schritt weiter gehen hier die *constraint-basierten Grammatiken*, die eine deutliche Trennung zwischen Strukturbeschreibung einerseits und Wohlgeformtheitsbedingungen für sprachliche Strukturen andererseits vollziehen. Ihr prominentester Vertreter ist die Head-driven Phrase Structure Grammar (HPSG, Pollard und Sag [92]).

Um die gewünschte Trennung zu erreichen, wird die Struktur für ein phrasales Zeichen direkt in seiner Merkmalsstruktur kodiert. In Anlehnung an die X-bar-Theorie [50] bestehen phrasale Zeichen jeweils aus einem lexikalischen Tochterknoten, dem *Kopf der Phrase*, sowie einem oder mehreren eingebetteten phrasalen Zeichen. Da die Kopftochter stets bestimmte Eigenschaften der anderen phrasalen Tochterknoten determiniert, schließt die entstehende hybride Modellierung auch Aspekte von Abhängigkeitsbeziehungen ein.

Für kophaltige phrasale Zeichen wird der Kopf in der Merkmalsstruktur des phrasalen Zeichens explizit ausgewiesen. Hierfür ist entsprechend den Konventionen der HPSG das Merkmal HEAD-DTR reserviert während sich die Informationen über die übrigen Tochterknoten in einer Liste unter dem Merkmal COMP-DTRS befinden.



Für die phrasalen Zeichen wird eine relativ kleine Menge von sehr allgemein formulierten Strukturschemata definiert, die durch die lexikalische Information der Wortformen eines Satzes so instanziert werden müssen, dass die resultierenden Merkmalsstrukturen den gegebenen Constraints genügen.

Constraints sind Implikationen über Merkmalsstrukturen, wobei die Prämisse die Relevanz des Constraints für eine bestimmte strukturelle Konfiguration beschreibt und die Conclusio zusätzliche Wohlgeformtheitsbedingungen für das jeweilige phrasale Zeichen bereit hält. So fordert etwa das folgende Constraint für alle Kopfstrukturen die Unifikation der lexikalischen Kopfmerkmale mit den Kopfmerkmalen des übergeordneten phrasalen Zeichen, d.h. die Projektion der kategorialen Eigenschaften des lexikalischen Kopfes auf die Phrasenebene, wo sie dann mit den Schwesterknoten in Kongruenz treten können:

$$\left[\text{DTRS} \left[\text{head-struc} \right] \right] \rightarrow \left[\begin{array}{l} \text{SYNSEM} | \text{LOC} | \text{CAT} | \text{HEAD } \boxed{1} \\ \text{DTRS} | \text{HEAD-DTR} | \text{SYNSEM} | \text{LOC} | \text{CAT} | \text{HEAD } \boxed{1} \end{array} \right]$$

Formale Grundlage für die Definition einer Implikation ist hierbei die Typisierung von Merkmalsstrukturen [20], wobei jeder Typ durch die Merkmale, die er lizenziert, charakterisiert ist und überdies die Typen der Werte bestimmt, die diese Merkmale annehmen können. Typen sind in einer Typhierarchie angeordnet, wobei ein Typ alle Merkmale von allen seinen Supertypen auf monotone Weise ererbt, zusätzlich aber auch eigene Merkmale einführen kann.

Der allgemeinste Typ ist der des sprachlichen Zeichens. Er stellt z.B. die Merkmale für die kategoriale Information des Zeichens (nominal, verbal, ...) bereit. Seine beiden Subtypen (lexikalisches Zeichen bzw. phrasales Zeichen) führen weitere Merkmale ein, z.B. für die Tochterknoten eines phrasalen Zeichens.

Mit den Operatoren Konjunktion, Disjunktion, Negation und Implikation, sowie der Möglichkeit zur rekursiven Einbettung von Merkmalsstrukturen entspricht die Semantik des der HPSG zugrundeliegenden Unifikationsformalismus unmittelbar einer Programmiersprache zur Logikprogrammierung. In ihr lassen sich die grundlegenden Verarbeitungsaufgaben als deduktive Prozesse über den Axiomen der Wissensbasis (lexikalische Information, Strukturschemata und Constraints) formulieren:

- Generierung: Gegeben ist ein partiell instanziertes phrasales Zeichen (für den Satz), das unter Zuhilfenahme der Axiome in der Wissensbasis maximal mit Information angereichert werden soll.
- Strukturanalyse: Gesucht ist das allgemeinste phrasale Zeichen, das sich aus den Axiomen der Wissensbasis ableiten lässt.

Die entscheidende Innovation, die die Arbeit mit getypten Merkmalsstrukturen in die maschinelle Sprachverarbeitung eingebracht hat, liegt vermutlich auf methodischem Gebiet. Analog zu Entwicklungen in der Softwaretechnik werden über die Typhierarchie Konzepte der Objektorientierung für den Grammatikentwurf erschlossen, während auf der anderen Seite die Constraints eine modulare Organisation des linguistischen Wissens ermöglichen. Sprachliche Regularitäten z.B. zur Projektion oder Kongruenz müssen nicht mehr in einer Vielzahl von Regeln als individuelle Koreferenzbedingungen kodiert werden, sondern können in abstrakter Form in

einem einzigen Constraint zusammengefasst werden. Constraints spielen daher in der HPSG eine Rolle, die vergleichbar ist mit den Prinzipien universalgrammatisch angelegter Theorien der kognitiven Linguistik [27].

13.4.3 Strukturanalyse

Chart-Parsing

Von den verschiedenen Verfahren zur Strukturanalyse natürlicher Sprache, sind diejenigen, die auf kontextfreien Grammatiken basieren, am weitesten verbreitet und am besten untersucht. Das Grundidee besteht dabei darin, für einen gegebenen Satz durch sukzessive Anwendung von Grammatikregeln einen Strukturbau zu ermitteln, so dass der Spitzenknoten des Baumes dem Startsymbol der Grammatik entspricht und die Blattknoten den Wortformen im Satz. Grundsätzlich stehen dabei mehrere Entwurfsalternativen zur Auswahl, deren Kombination zu verschiedenen Analysestrategien führen:

- Die Regeln der Grammatik können von links nach rechts abgearbeitet werden oder von rechts nach links. Im ersten Fall werden *Erwartungen* über den Typ der zu analysierenden Konstituente *top down* expandiert, bis sie sich auf den Wortformen des Eingabesatzes verifizieren lassen. Im zweiten Fall werden Folgen von Nichtterminalsymbolen *datengetrieben (bottom up)* solange zusammengefasst (reduziert), bis im Erfolgsfall das Startsymbol der Grammatik übrigbleibt.
- Da in einer realistischen Grammatik stets mehrere Regeln zur *top down*-Expansion bzw. zur *bottom up*-Reduktion zur Verfügung stehen, ergibt sich ein Suchproblem. Die Analyseverfahren unterscheiden sich dann hinsichtlich der Reihenfolge in der die Alternativen überprüft werden: Werden erst alle Regeln angewendet, ehe der nächste Expansions/Reduktionsschritt erfolgt, führt dies zu einer Breitensuche, während die weitere Expansion/Reduktion der aktuellen Analysevariante eine Tiefensuche ergibt.
- Der Eingabesatz kann sowohl von links nach rechts, als auch von rechts nach links konsumiert werden, wobei jedoch die letztere Variante nur in Spezialanwendungen (Inselparsing für gesprochene Sprache) eine Rolle spielt.

Keine der möglichen Strategien ist unter allen Umständen vorteilhaft. Tiefensuche benötigt linearen Speicheraufwand, naive Breitensuche hingegen exponentiellen. Dafür erlaubt es aber die Breitensuche, verschiedene partielle Analysen miteinander zu vergleichen und, falls eine Auswahlheuristik existiert, eine Bestensuche zu realisieren.

Andere Kriterien für die Wahl zwischen verschiedenen Analysestrategien sind grammatisch abhängig. Erwartungsgesteuerte (links-rechts) Tiefensuche etwa kann zu Terminierungsproblemen führen, falls die Grammatik (links-)rekursive Regeln enthält. Sie lassen sich zwar systematisch in rechtsrekursive Regeln umwandeln, eine Linksrekursion in der linguistisch gewünschten Strukturbeschreibung muss dann jedoch mit den Mitteln der Unifikation erzeugt werden (s. Abb. 13.7).

Enthält die Grammatik Nullexpansionen, die insbesondere für die systematische Behandlung von Fragesatz- und Topikalisationkonstruktionen vorteilhaft sein können, führt dies bei *bottom up*-Analyse zu Schwierigkeiten, da die erforderlichen, aber unsichtbaren Nullelemente praktisch an beliebigen Stellen im Satz hypothetisiert werden müssten.

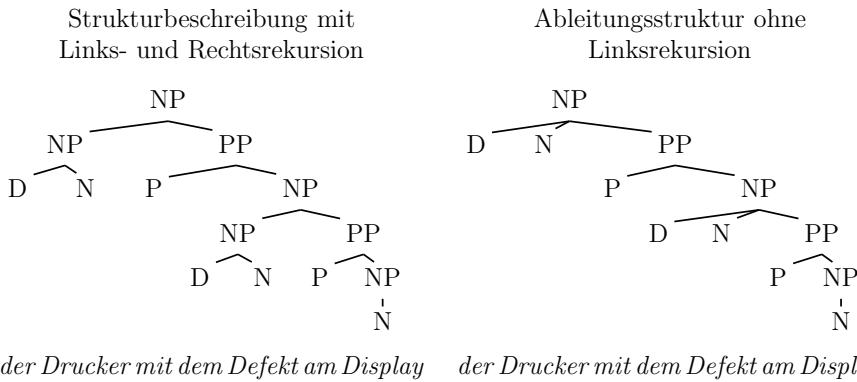


Abbildung 13.7: Strukturbeschreibung mit und ohne Linksrekursion.

Der wohl schwerwiegendste Nachteil naiver Analyseverfahren ist der exponentielle Rechenzeitbedarf. Dieser kann durch Memoisierungstechniken und die Wiederverwendung partieller Analyseergebnisse bis auf einen kubischen Aufwand reduziert werden. Grundlage dafür ist eine Datenstruktur zur Speicherung von Zwischenergebnissen, die *Chart* genannt wird. Eine Chart ist ein gerichteter azyklischer Graph mit einer totalen Präzedenzrelation zwischen den Knoten, die den Wortzwischenräumen im Satz entsprechen. Kanten sind entweder lexikalisch annotiert, oder tragen einen Vermerk, durch welche Regel sie jeweils erzeugt wurden und in welchem Umfang diese Regel bereits abgearbeitet wurde. Dabei wird der aktuelle Abarbeitungszustand einer Regel durch einen Punkt auf der rechten Regelseite notiert, z.B. $VP \Rightarrow V . NP PP$.

Eine Kante $\langle i, j, VP \Rightarrow V . NP PP \rangle$ von Knoten i zu Knoten j mit der angegebenen Beschriftung zeigt demnach an, dass beginnend mit dem Knoten i eine phrasale Kategorie VP hypothetisiert werden kann, wobei von den drei geforderten Kategorien auf der rechten Regelseite erst eine und zwar zwischen i und j mit der Kategorie V gefunden wurde. Zur Bestätigung der Hypothese, wird jedoch rechts von j noch eine Kantenfolge mit zwei aufeinanderfolgenden Kategorien NP und PP benötigt. Damit repräsentiert eine solche *aktive* Kante drei verschiedenen Arten von Information: eine Hypothese über eine mögliche Kategorie rechts von i , eine bereits zwischen i und j gefundene partielle Bestätigung für die Hypothese, sowie Erwartungen an den Kontext rechts von j . Kanten, die für vollständig bestätigte Hypothesen stehen und daher keine offenen Erwartungen an den rechten Kontext mehr haben, werden *inaktive* Kanten genannt.

Erwartungen und Daten können durch *Kantenexpansion* aufeinander abgebildet werden. Enthält die Chart eine aktive Kante $\langle i, j, A \Rightarrow w_1 . B w_2 \rangle$, und eine inaktive Kante $\langle j, k, B \Rightarrow w_3 . \rangle$ wobei die w_i für möglicherweise auch leere Symbolfolgen stehen, kann eine neue Kante $\langle i, k, A \Rightarrow w_1 B . w_2 \rangle$ in die Chart eingefügt werden (Fundamentalregel). In diesem Falle wurde eine Erwartung an den rechten Kontext (hier die Kategorie B) durch die vorgefundenen Daten bestätigt.

Durch zwei unterschiedliche Regeln zur *Kanteneinführung* werden die beiden Analysevarianten *top down* bzw. *bottom up* realisiert:

- *top down*-Kanteneinführung:

Für jede neu hinzukommende Kante $\langle i, j, A \Rightarrow \psi_1 . B w_2 \rangle$ füge für jede Regel $B \Rightarrow w_m$ aus der Grammatik eine weitere Kante $\langle j, j, B \Rightarrow . w_m \rangle$ zur Chart hinzu. Der Prozess wird

mit der globalen Erwartung für das Startsymbol der Grammatik initialisiert, indem für alle Regeln $S \Rightarrow w_n$ mit dem Startsymbol auf der linken Seite eine Kante $\langle 0, 0, S \Rightarrow w_n \rangle$ in die Chart eingefügt wird. Die Kanteneinführungsregel perkoliert dann diese Erwartung durch die Baumstruktur nach unten, bis ein Abgleich mit den Daten im Satz möglich ist.

- *bottom up*-Kanteneinführung:

Für jede neu hinzukommende inaktive Kante $\langle i, j, A \Rightarrow w_1 . \rangle$ füge für jede Regel $C \Rightarrow A w_2$ aus der Grammatik eine Kante $\langle i, j, C \Rightarrow A . w_2 \rangle$ in die Chart ein.

Durch das Wechselspiel von Kanteneinführung und Kantenexpansion werden, wie im Bild 13.8 für die *top down*-Analyse dargestellt, monoton neue Kanten in die Chart eingetragen, bis ein Fixpunkt erreicht ist.

Die Strukturbeschreibung für den Satz kann anschließend als konsistente Teilmenge von inaktiven Kanten aus der Chart extrahiert werden. Dadurch, dass alle (auch die partiellen) Zwischenergebnisse in der Chart protokolliert sind, wird Mehrfachaufwand durch wiederholte Überprüfung gleicher Hypothesen an gleicher Position im Satz vermieden. Diese Wiederverwendung von Zwischenergebnissen kann jedoch erschwert und im Extremfall ganz verhindert werden, wenn an die Stelle der atomaren Symbole der kontextfreien Grammatik, die komplexen Merkmalsstrukturen einer Unifikationsgrammatik treten.

Die Menge aller aktiven Kanten einer Chart bildet die *Agenda* der Strukturanalyse. Sie kann auf verschiedene Weise organisiert werden (Kellerspeicher, Warteschlange, sortierte Liste), um so die unterschiedlichen Suchstrategien (Breite-zuerst, Tiefe-zuerst, Bestensuche) zu realisieren.

Parsing als Constraint Satisfaction

Auch wenn die constraint-basierten Unifikationsgrammatiken, wie etwa die HPSG (vgl. Abschnitt 13.4.2) mit der Unterscheidung zwischen Strukturschemata und Constraints bereits eine formale Trennung von Strukturrepräsentation und Wohlgeformtheitsbedingungen einführen, werden die beiden Bestandteile des grammatischen Wissens immer noch im Rahmen eines einheitlichen Formalismus modelliert und überdies mit konsistenzbasierten Ableitungsmechanismen verarbeitet. Genau diese enge Verbindung zwischen zwei völlig unterschiedlichen Aspekten ist es aber, die erhebliche Nachteile im Hinblick auf ein robustes Verhalten der Analyse mit sich bringt: Scheitert ein Constraint beim Vorliegen von extra- oder agrammatischen Eingabedaten, so steht auch das zugrundeliegende Strukturschema nicht mehr zur Verfügung. Im günstigsten Fall wird als Resultat eine fehlerhafte Strukturbeschreibung ermittelt, oft genug scheitert die Analyse jedoch ganz.

Dieser Effekt kann nur dadurch vermieden werden, dass der Hypothesenraum für die Ermittlung der Strukturbeschreibungen vollständig von den Wohlgeformtheitsbedingungen getrennt wird. Erst dann lassen sich Constraints, die eine Analyse zum Scheitern bringen können, gezielt außer Kraft setzen, so dass auch im Fehlerfall noch ein zumindest partiell korrektes Analyseergebnis vorliegt.

Von den strukturellen Beschreibungsmitteln eignen sich Dependenzstrukturen besonders gut für ein solches Herangehen: Die Wortformen werden als Variable des CSP etabliert, denen als Werte jeweils Paare, bestehend aus einer der anderen Wortform im Satz und einem Label, zugewiesen werden [75]. Die Wertepaare können dann als Unterordnungsrelationen einer Dependenzstruktur interpretiert werden (vgl. Bild 13.9).

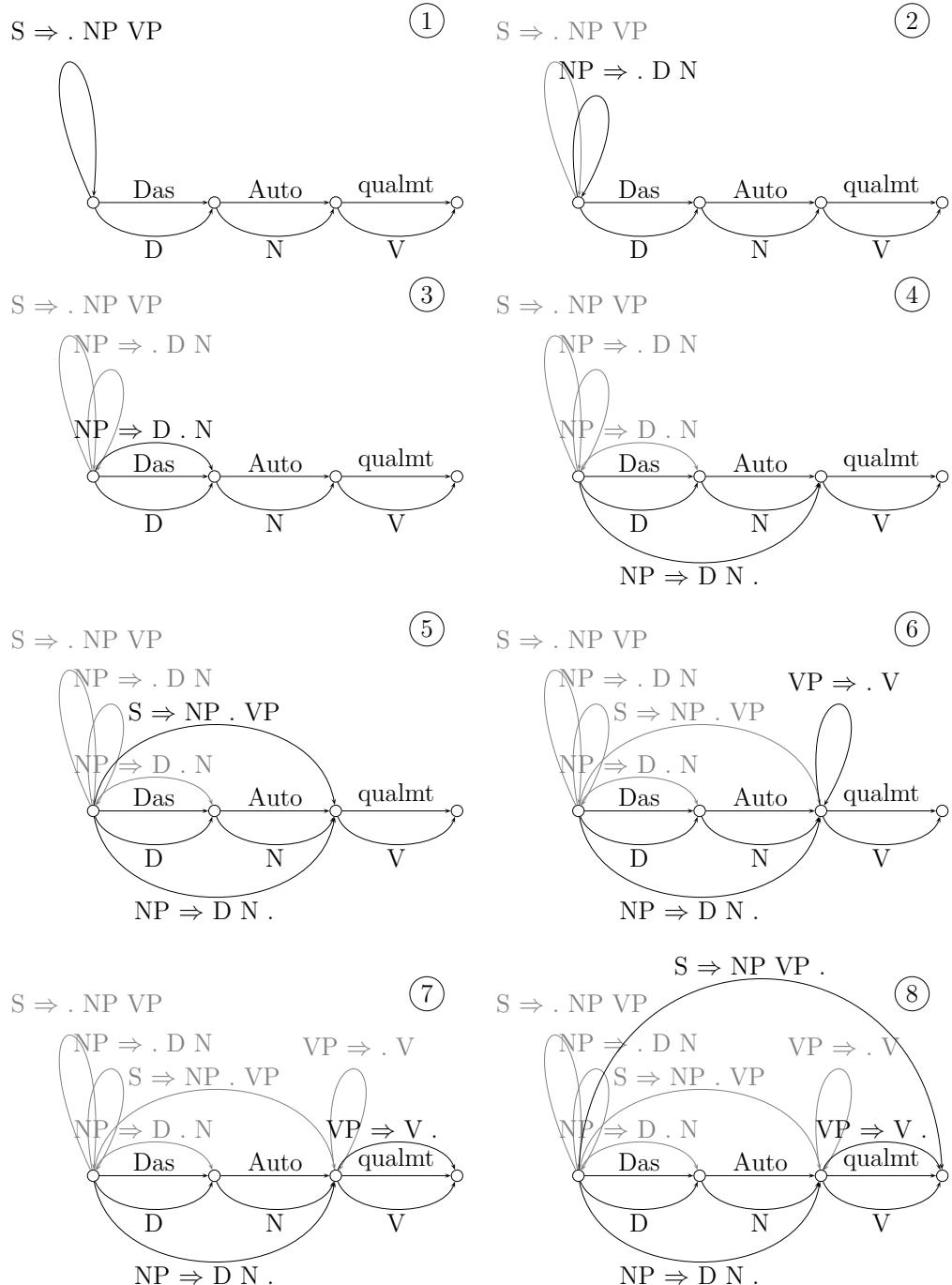


Abbildung 13.8: Top down-Analyse für einen Satz. Ältere aktive Kanten sind jeweils grau dargestellt.

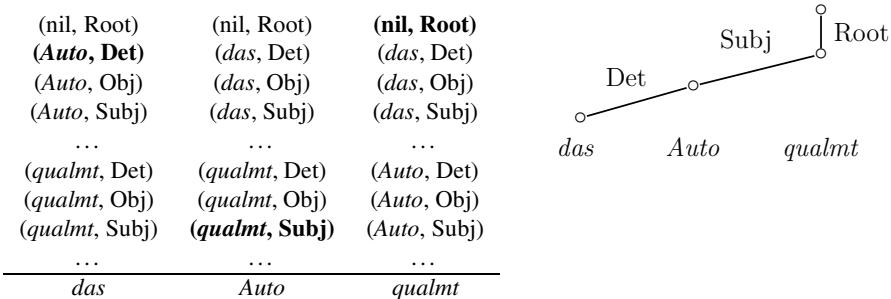


Abbildung 13.9: Hypothesenraum (links) und ausgewählte Dependenzbaum (rechts) für ein Constraint-Satisfaction-Problem. Die Wertebindungen für die ausgewählte Strukturbeschreibung sind in der Tabelle fett hervorgehoben.

Die Constraints der Grammatik beschreiben zulässige lexikalische Eigenschaften an den Knoten einer Unterordnungsbeziehung, lineare Reihenfolgebeziehungen zwischen den Knoten oder auch Verträglichkeitsbedingungen für die Kombination mehrerer Knoten im Baum. So fordern etwa die beiden einstelligen Constraints

$$\{X\} : \text{DetNomCategory} : \text{Det} : 0.0 : X \downarrow \text{cat} = \text{D} \rightarrow X \uparrow \text{cat} = \text{N} \wedge X.\text{label} = \text{Det}$$

$$\{X\} : \text{DetNomPrecedence} : \text{Det} : 0.0 : X.\text{label} = \text{Det} \rightarrow X \downarrow \text{pos} < X \uparrow \text{pos},$$

dass ein Artikel (Kategorie D) nur ein Nomen (Kategorie N) mit dem Label Det modifizieren darf und dass bei einer solchen Unterordnung der Artikel links vom Nomen stehen muss.

Zur Steigerung der Robustheit des Verfahrens gegen extra- und agrammatische Eingabedaten können Constraints gewichtet werden. Dadurch werden sie prinzipiell verletzbar, wobei durch das Gewicht eines Constraints die Kosten für seine Verletzung beschrieben werden. Hierdurch wird das Constraint-Satisfaction-Problem zu einer Optimierungsaufgabe, die diejenige Struktur identifizieren soll, die die geringsten Gesamtkosten verursacht.

Mit Hilfe von gewichteten Constraints können eine Vielzahl von schwachen Präferenzen und sogar widersprüchliche Wohlgeformtheitsbedingungen in die Modellierung einbezogen werden, die bei einer strikt binären Repräsentation (wie im Falle einer Unifikationsgrammatik) unberücksichtigt bleiben müssen. Ein ähnliches Herangehen findet sich etwa bei der Optimalitäts-theorie [93], in der Constraints in eine Hierarchie eingeordnet werden und diejenige Strukturbeschreibung präferieren, die die am wenigsten schwerwiegende Constraintverletzung hervorruft. Im Unterschied zu Constraintgewichten lassen sich so aber keine kumulativen Effekte beschreiben, die beim Auftreten mehrfacher Constraintverletzungen durchaus nachweisbar sind [58].

Ein wesentlicher Vorzug der Trennung von Strukturbeschreibung und Wohlgeformtheitsbedingungen besteht darin, dass sich zahlreiche bekannte Lösungsverfahren für Constraint-Satisfaction-Probleme (CSP) mit ihren teilweise komplementären Berechnungseigenschaften auf die Strukturanalyse natürlicher Sprache übertragen lassen (vgl. Abschnitt 13.4.3). Möglicherweise nachteilig ist hingegen die Tatsache, dass – obwohl die Constraints selbst eine deklarative Semantik besitzen – die prinzipielle Umkehrbarkeit der Verarbeitung von der Analyse zur Generierung nicht mehr ohne weiteres gegeben ist.

13.4.4 Robuste Verfahren

Trotz erheblicher Fortschritte bei der Entwicklung von Repräsentationsformalismen für sprachliches Wissen, sowie von entsprechenden Analyseverfahren, existiert bis heute kein einsatzfähiges Analysesystem auf der Basis eines binären Wohlgeformtheitsbegriffes, das in der Lage wäre, eindeutige Strukturanalysen für unrestringierte natürliche Sprache mit hoher Zuverlässigkeit zu ermitteln. Der Grund hierfür scheint darin zu liegen, dass die Annahme, die Grammatik einer natürlichen Sprache ließe sich auf der Basis eines widerspruchsfreien logischen Axiomensystems modellieren, eine für praktische Zwecke zu starke Idealisierung darstellt, die sich offenbar nur für kleine Ausschnitte einer Sprache aufrecht halten lässt.

Mit zunehmender Abdeckung der Grammatik wächst hingegen die Ergebnismehrdeutigkeit der Analyse zwangsläufig an. Die lange Zeit gehegte Hoffnung, dass man diesem Trend durch eine möglichst feine linguistische Modellierung wirksam begegnen könnte, hat sich letztendlich nicht erfüllt, so dass auch elaborierte Unifikationsgrammatiken im Regelfall noch mehrere Dutzend Analysen pro Satz produzieren [65, 67]. Voraussetzung für einen Einsatz in praktischen Anwendungen ist daher immer noch eine zusätzliche und stark fehleranfällige Prozedur zur Auswahl der plausibelsten Strukturhypothese.

Um trotz dieses Dilemmas zu praktikablen Lösungen zu kommen, wird an Alternativen gearbeitet, die entweder mit flachen oder unterspezifizierten Strukturrepräsentationen arbeiten, eine plausibilitätsgestützte Auswahl zwischen alternativen Hypothesen unterstützen oder aber widersprüchliches grammatisches Wissen explizit zulassen.

Flache Repräsentationen

Flache Repräsentationen bieten sich als Alternative zu einer vollständigen Strukturanalyse immer dann an, wenn die Aussicht besteht, dass man eine Verarbeitungsaufgabe (zumindest approximativ) auch mit weniger anspruchsvollen Annahmen über die vorliegenden linguistischen Strukturen lösen kann. In vielen Fällen dienen die flachen Strukturen auch nur als Vorstufen für eine nachgeschaltete tiefergehende Strukturanalyse, die dann über einem reduzierten Hypothesenraum arbeiten kann.

Den einfachsten Fall einer Prozedur zur flachen Analyse stellt das *Tagging*, eine (kontextabhängige) Klassifikation elementarer linguistischer Einheiten (zumeist Wortformen) unter Verwendung eines vorgegebenen Kategorieninventars dar. Bekanntester Vertreter hierfür ist die Zuordnung von Wortarten zu den Wortformen eines Satzes (*part-of-speech-Tagging*, Brants [15]), die oftmals zur Vorverarbeitung eingesetzt wird, um die Strukturanalyse von der Wortartenmehrdeutigkeit zu entlasten. Sehr ähnliche Analysetechniken können aber auch auf komplexe lexikalische Beschreibungen (*Supertags*, Bangalore und Joshi [9]), auf aufgabenbezogene Kategorien (z.B. verschiedene Arten von Eigennamen) oder die semantische Disambiguierung von Wortformen angewendet werden.

Bewährt haben sich hierbei endliche Automaten, constraint-basierte Verfahren, probabilistische Modelle, Trennflächenklassifikatoren oder aber transformations-basierte Ansätze. Letztere überführen eine initiale Annahme über die Tag-Sequenz durch die Anwendung kontextsensitiver Ersetzungsregeln schrittweise in eine weniger fehlerhafte Beschreibung. Ebenfalls zur Klasse der flachen Verfahren gehört das *Chunking* [106], bei dem ein Satz in elementare Phrasen (z.B. Nominal- oder Präpositionalgruppen) segmentiert wird, denen ggf. auch eine Kategorie zugewiesen werden kann.

Aggregierende Beschreibungen

Zu den robusten Techniken sind auch aggregierende Ansätze zu rechnen, die ein zusammenfassendes Urteil über einen Text bzw. eine Textpassage ermitteln, ohne dabei eine detaillierte Analyse der syntaktisch-semantischen Zusammenhänge durchzuführen. Klassische Vertreter hierfür sind inhaltliche Beschreibungen mit Hilfe von Schlüsselwörtern [7]. Zunehmend konzentriert sich aber das Interesse der Forschung auf eine Einschätzung der subjektiven Einstellung des Autors (*opinion mining, sentiment detection*). Dabei handelt es sich um eine Aufgabenstellung, die insbesondere zur Analyse von Produktbewertungen im Kontext von e-Business-Anwendungen von erheblicher Relevanz ist [91].

Grundlage für die Ermittlung von subjektiven Einstellungen sind Polaritätswörterbücher, die Auskunft über eine mehr oder weniger positive bzw. negative Färbung eines Wortes geben. So deutet *haltbar* etwa auf eine positive Einstellung hin, während *nutzlos* eher eine negative Grundstimmung signalisiert. Zu beachten ist dabei, dass solche Bewertungen in vielen Fällen kontext- und aufgabenspezifisch sein können (*scharfes Messer* vs. *scharfe Kanten*, *lange Haltbarkeit* vs. *lange Wartezeit* usw.). Polaritätsinformation kann durch Intensifikatoren (*sehr, ziemlich, kaum, etwas, ...*) verstärkt oder abgeschwächt werden. Negatoren (*kein, nicht, niemals*) invertieren einen Polaritätsbeitrag.

Das Zusammenwirken dieser Faktoren kann sowohl durch manuell optimierte Heuristiken [110] oder aber mit Hilfe von trainierten Modellen [91] beschrieben werden. In beiden Fällen erfolgt die Bewertung aber aufgrund von rein quantitativen Kriterien (Stärke und Häufigkeit von Polaritätsindikatoren), die nur schlecht geeignet sind, rhetorische Wendungen (kontrastive Aufzählungen, Ironie, Sarkasmus usw.) oder Aspekte der Textstruktur angemessen zu berücksichtigen.

Unterspezifikation

Eng verwandt mit dem Ansatz der flachen Repräsentationen ist die Idee der Unterspezifikation, die in einer einfachen Form bereits im Zusammenhang mit Merkmalsstrukturen als komplexe Kategorien aufgetreten war. So enthält etwa der lexikalische Eintrag für die Wortform *Frau* im Abschnitt 13.4.2 keine Information zum Merkmal Kasus, da die weiblichen Nomina im Deutschen bezüglich dieses Merkmals nicht differieren. Die dem zugrunde liegende Idee, mehrere Lesarten für ein sprachliches Zeichen in einer gemeinsamen Repräsentation darzustellen, und eine Differenzierung zwischen den verschiedenen Varianten erst bei Bedarf vorzunehmen, lässt sich auch auf strukturelle Mehrdeutigkeiten übertragen. Der Vorteil im Hinblick auf die Robustheit besteht dann darin, dass dem Analyseverfahren keine Entscheidungen mehr zugemutet werden, die erfahrungsgemäß nur schwer zu treffen sind. Ein solches Vorgehen ist immer dann sinnvoll, wenn eine strukturelle Mehrdeutigkeit aus der zu analysierenden Äußerung ohne Informationsverlust in die Zielstruktur der Verarbeitung übernommen werden kann. Häufig ist dies bei der Übersetzung in eine andere Sprache der Fall, z.B. im Falle der strukturellen Anbindung einer Präpositionalgruppe (PP attachment). So kommen in komplexen Nominalgruppen wie

der Mann mit dem Koffer vor der Tür

mehrere Unterordnungsmöglichkeiten in Frage: Anbindung der Präpositionalgruppe *vor der Tür* als Lokation an die Nominalgruppe *der Mann* oder aber an die vorangehende Präpositionalgruppe *mit dem Koffer*. Die Übersetzung z.B. ins Englische bleibt jedoch unbeeinflusst davon, für welche der beiden Varianten man sich hierbei entscheidet. Daher ist es auch nicht sinnvoll, die Analyse mit einer Mehrdeutigkeit zu belasten, die das Verarbeitungsresultat nicht beeinflusst. Dem

trägt die Unterspezifikation im folgenden Beispiel Rechnung, die zwar die Anbindungsrichtung und die Kategorie der übergeordneten Wortform spezifiziert, den genauen Anbindungspunkt aber offen lässt [57].

<i>der</i>	<i>Mann</i>	<i>mit</i>	<i>dem</i>	<i>Koffer</i>	<i>vor</i>	<i>der</i>	<i>Tür</i>
@DN>	@SUBJ	@<NOM	@DN>	@<P	@<NOM	@DN>	@<P

Dabei bedeutet @DN> die Unterordnung eines Artikels unter ein Nomen rechts davon, @<NOM die nach links gerichtete Unterordnung unter ein Nomen und @<P die nach links gerichtete Unterordnung unter eine Präposition. Insgesamt sind auf diese Weise 18 Strukturvarianten in einer einzigen, strikt sequenziellen Repräsentation kodiert.

Ein ähnliches Vorgehen ist auch auf der semantischen Ebene möglich, etwa im Falle der sprachlichen Quantoren, deren Zusammenwirken oftmals eine systematische Skopusmehrdeutigkeit mit sich bringt [13]. Für den Satz

Jeder Mann hat einen Traum.

lassen sich zwei Lesarten identifizieren, die sich vor allem im Skopuss der Quantoren unterscheiden. Im ersten Fall, hat der sprachliche Allquantor *jeder* einen weiten Skopuss über die gesamte Bedeutungsrepräsentation und damit auch über den Existenzquantor:

$$\forall x . \text{mann}(x) \rightarrow \exists y . \text{traum}(y) \wedge \text{haben}(x, y)$$

d.h. jeder Mann hat einen anderen Traum, während im zweiten Fall sein Skopuss den Existenzquantor von *einen Traum* nicht mit einschließt:

$$\exists y . \text{traum}(y) \wedge \forall x . \text{mann}(x) \rightarrow \text{haben}(x, y)$$

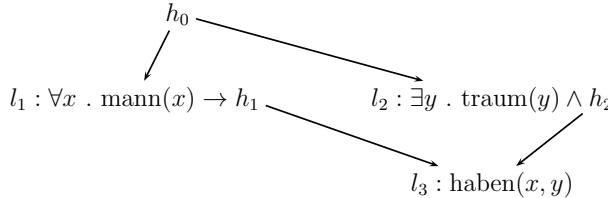
d.h. alle Männer träumen den gleichen Traum. Diese beiden Lesarten können in einer gemeinsamen Repräsentation zusammengefasst werden, indem sie in ihre gemeinsamen (parametrisierten) Bestandteile zerlegt werden:

$$\begin{aligned} l_1 &: \forall x . \text{mann}(x) \rightarrow h_1 \\ l_2 &: \exists y . \text{traum}(y) \wedge h_2 \\ l_3 &: \text{haben}(x, y) \end{aligned}$$

Dabei stellen die l_i eindeutige Bezeichner für die Teilausdrücke dar, während die Parameter h_i als Platzhalter dienen, die durch andere Teilausdrücke substituiert werden können. Die kombinatorischen Möglichkeiten, solche Bausteine erneut zu kompletten Formeln zusammenzusetzen, werden durch zusätzliche Constraints beschränkt, wobei ein weiterer Parameter h_0 angenommen wird, der für den zu konstruierenden Gesamtausdruck steht:

$$l_1 \leq h_0, \quad l_2 \leq h_0, \quad l_3 \leq h_1, \quad l_3 \leq h_2.$$

Dabei bezeichnet der Operator \leq die Möglichkeit, den Ausdruck für den das l_i steht, an der Stelle des entsprechenden Parameters h_i einzusetzen. Dies bedeutet im gegebenen Beispiel, dass die Teilausdrücke l_1 und l_2 nur für den Parameter h_0 substituiert werden dürfen, der Teilausdruck l_3 nur für die Parameter h_1 und h_2 :



Darüberhinaus gilt, dass das die Einsetzung nur dann einen zulässigen Gesamtausdruck ergibt, wenn alle Teilausdrücke genau einmal verwendet wurden. Damit können außer den beiden oben angegebenen Repräsentationen keine weiteren Gesamtausdrücke gebildet werden. Wesentlich ist jedoch, dass eine Expansion in die Lesarten im Normalfall nicht erforderlich ist, da auch die sich anschließenden Inferenzprozesse der semantischen Auswertung auf derartigen unterspezifizierten Strukturen arbeiten können.

Diese Technik zur Unterspezifikation lässt sich auch auf die Semantik von Diskurspartikeln, wie z.B. das *noch* im Satz *Dann sollten wir noch eine Runde drehen*. übertragen. Aufgrund der unterschiedlichen Betonungsvarianten (auf *noch* oder aber auf *Runde*) ergeben sich auch hier wieder zwei Lesarten, die in einer unterspezifizierten Beschreibung zusammengefasst werden können.

Probabilistische Verfahren

Eine weitere Möglichkeit zur Verbesserung der Robustheit von Analyseverfahren besteht in der Verwendung stark übergenerierender Grammatikmodelle, die neben den wohlgeformten Sätzen einer Sprache auch zahlreiche ungrammatische Konstruktionen akzeptieren. Dabei wird versucht, die bei einem solchen Vorgehen zwangsläufig entstehende, extrem hohe Mehrdeutigkeit durch die Auswahl der *optimalen* Strukturbeschreibung auf der Basis eines geeigneten Plausibilitätsmaßes wieder zu kompensieren. Typische Vertreter hierfür sind die probabilistischen Modelle, die inzwischen in allen Bereichen der Sprachverarbeitung starke Verbreitung gefunden haben. Ausschlaggebend hierfür ist sicherlich die prinzipielle Fähigkeit probabilistischer Ansätze, z.B. auf der Basis von (Hidden) Markov Modellen oder Dynamischen Bayes'schen Netzen [131], Entscheidungen über die optimale (sequentielle oder auch hierarchische) *Struktur* für sprachliche Beobachtungsdaten herbeizuführen.

Für ein probabilistisches Modell ergibt sich aufgrund der Bayes'schen Regel die fehleroptimale Entscheidungsvorschrift stets zu

$$x_o = \arg \max_{x_i \in X} P(x) P(o|x)$$

wobei

- o eine Sequenz von Beobachtungsdaten (Merkmalsvektoren in der Sprachsignalverarbeitung oder Wortformen in der Textverarbeitung),

- x eine Sequenz von Ausgabedaten (in der Regel symbolische Kategoriezuordnungen: Wortformen, Wortarttags, Supertags, semantische Klassen usw.),
- X die Menge aller zulässigen Sequenzen aus den verwendeten Kategorien,
- $P(x)$ die *a priori* Wahrscheinlichkeit der jeweiligen Ergebnissequenz und
- $P(o|x)$ die *a posteriori* Wahrscheinlichkeit der jeweiligen Ergebnissequenz

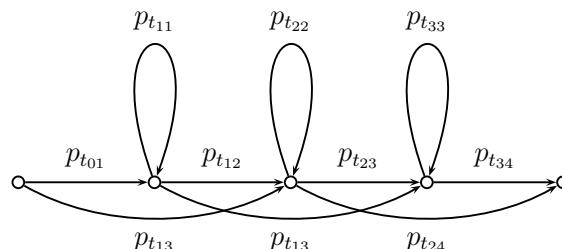
ist.

Für die Zwecke der Spracherkennung und der maschinellen Übersetzung etwa wird die *a priori* Wahrscheinlichkeit $P(x)$ durch ein *Sprachmodell* erfasst, das die Wahrscheinlichkeit beliebiger Wortformsequenzen einer Sprache z.B. durch ein Markov-Modell erster oder zweiter Stufe (Bis bzw. Trigram-Modell) approximiert. Ein solches Modell ist im Hinblick auf eine hohe Ergebnisqualität besonders wirksam, wenn es eine geringe *Perplexität* (eine Exponentialfunktion des Informationsgehalts) besitzt. Aktuelle Untersuchungen zielen darauf ab, über rein sequenzielle Abhängigkeiten hinaus auch strukturelle Aspekte in die Sprachmodellierung einzubeziehen. Hierfür kommen dann z.B. probabilistische kontextfreie Grammatiken zum Einsatz (s.u.).

Die Modellierung der *a posteriori* Wahrscheinlichkeit erfordert ein generatives Modell, das die Wahrscheinlichkeit von Beobachtungssequenzen aufgrund von gegebenen Ergebnissequenzen vorhersagen kann. Wegen ihrer großen Flexibilität bei der Anpassung an die in den Daten verborgenen Strukturen haben sich hierfür insbesondere Markov-Modelle mit verborgenen Zuständen (hidden markov models, HMM) bewährt, bei denen die Beobachtung nicht unmittelbar mit den Modellzuständen verknüpft ist, sondern erst durch einen weiteren stochastischen Prozess mit den Zuständen in Verbindung gebracht wird.

Aufgrund dieser Trennung unterscheidet man in einem HMM drei verschiedene Wahrscheinlichkeitstypen:

- die Menge der Initialwahrscheinlichkeiten, die für jeden Modellzustand angibt, mit welcher Wahrscheinlichkeit er den Startzustand der Verarbeitung bildet,
- die Menge der Transitionswahrscheinlichkeiten und
- die Menge der Emissionswahrscheinlichkeiten.



Für die Spracherkennung verwendet man üblicherweise Phonmodelle, die einen Laut im Kontext seiner Nachbarn modellieren (Triphone). Phonmodelle sind stets vorwärtsverkettet und erlauben es, Modellzustände zu wiederholen (Loops) und gegebenenfalls auch zu überspringen (Skips). Die Details der Modelltopologie (wieviele Zustände, welche Transitionen) sind Gegenstand des Systementwurfs.

Für jeden Modellzustand geben die Emissionswahrscheinlichkeiten an, mit welcher Wahrscheinlichkeit eine Beobachtung durch den jeweiligen Zustand erzeugt worden sein könnte. Hierfür werden im Falle der Spracherkennung überwiegend Gauss'sche Mischverteilungen benutzt, die die Lage der zugehörigen Merkmalsvektoren in einem hochdimensionalen Merkmalsraum durch die Überlagerung mehrerer Normalverteilungen beschreiben. Die Zustände eines HMM stellen somit nur eine indirekte Vermittlung zwischen der symbolischen Repräsentation für einen Signalabschnitt (Wortformen, Phonsequenzen) und der Signalrealität (Merkmalsvektoren) her. Sie selbst tragen keinerlei Bedeutung und sind auch nicht unmittelbar beobachtbar, d.h. sie bleiben einem externen Beobachter „verborgen“.

Phonmodelle werden über die im Aussprachewörterbuch kodierten Phonsequenzen zu Wortmodellen verkettet und gehen so in die Berechnung der optimalen Wortformsequenz ein. Durch die Existenz alternativer Worthypothesen und die Möglichkeit über Loops und Skips die Zuordnung zwischen Modellzuständen und Merkmalsvektoren zu variieren, ergibt sich ein Suchraum, in dem die optimale Zuordnung durch dynamische Programmierung [11] ermittelt werden kann. Dabei kommt in der Regel der Viterbi-Algorithmus zur Anwendung [116].

Die Qualität von Spracherkennungsergebnissen ist von zahlreichen Parametern abhängig, u.a. von der Wortschatzgröße, dem Sprachregister (vorgelesene Sprache, Diktiersprache, Spontansprache), der Kanalqualität (Nahbesprechungsmikrophon, Festnetztelefon, Mobiltelefon, Raumakustik, Störgeräusche usw.), dem Nutzerkreis (Dialekte, Slang, Nichtmuttersprachler usw.), den zur Verfügung stehenden Trainingsdaten zum Schätzen der Modellparameter, sowie der zu erkennenden Sprache. Erkennungsraten zwischen 70% für spontan gesprochene Sprache und 99% für die Ziffernerkennung unter idealen akustischen Bedingungen sind durchaus typisch.

Obwohl die Idee zur Verwendung von HMMs im Kontext der Erkennung gesprochener Sprache entstanden ist, hat sich gezeigt, dass sie sich erfolgreich auf zahlreiche andere Sprachverarbeitungsaufgaben übertragen lässt. Beim Tagging, etwa für Wortarten, Supertags oder semantische Kategorien, kommen ergodische Markov-Modelle zum Einsatz, bei denen alle Knoten des Modells durch Transitionen miteinander verbunden sind. Ansonsten muss das Grundschema nur geringfügig variiert werden. Da im Falle des Taggings eine eindeutige Abbildung zwischen Beobachtungen und Ergebniskategorien gesucht ist, können die Kategorien den Modellzuständen direkt zugeordnet werden. Obwohl die Zustände dadurch eine Bedeutung erhalten und somit im strengen Sinne nicht mehr „verborgen“ sind, wird wegen der offensichtlichen Parallelen in der Modellstruktur die Bezeichnung HMM beibehalten. Typische probabilistische Wortartentagger erreichen eine Genauigkeit von über 96%.

Für die maschinelle Übersetzung muss das generative Modell für die *a posteriori* Wahrscheinlichkeit weiter modifiziert werden. Man unterscheidet hier zwei Komponenten [19]

- ein Fertilitätsmodell, das angibt, mit welcher Wahrscheinlichkeit eine Zielsprachenwortform aus mehreren Quellsprachenwortformen entstanden sein könnte und
- ein Transfermodell, das angibt, mit welcher Wahrscheinlichkeit eine Zielsprachenwortform aus einer Quellsprachenwortform entstanden sein könnte.

Da Markov-Modelle ausschließlich sequentielle Beziehungen erfassen können, sind sie für die strukturelle Analyse natürlicher Sprache nicht geeignet. Hierfür werden im allgemeinen probabilistische kontextfreie Grammatiken verwendet, die den einzelnen Regeln $A \Rightarrow w$ bedingte

Wahrscheinlichkeiten der Art $P(w|A)$ zuweisen, wobei w die Symbolsequenz auf der rechten Regelseite darstellt [24]. Unter der Annahme der statistischen Unabhängigkeit für Regelanwendungen ergibt sich die Wahrscheinlichkeit für einen Strukturbaum t als Summe über die Wahrscheinlichkeiten für alle Ableitungen d , durch die er erzeugt werden kann:

$$P(t) = \sum_{d \in t} \prod_{r \in d} P(r)$$

Leider erweist sich die dieser Berechnungsvorschrift zugrunde liegende Unabhängigkeitsannahme als in erheblichem Maße unbegründet. Dieser Tatsache versucht man z.B. dadurch Rechnung zu tragen, dass die Regelwahrscheinlichkeiten durch zusätzliche stochastische Abhängigkeiten etwa vom Kopf der übergeordneten oder benachbarten Knoten im Syntaxbaum erweitert werden [26, 28]. Eine Alternative besteht im Ansatz des datenorientierten Parsings (DOP-Model), wo statt der Wahrscheinlichkeiten für Regeln (die elementaren Bäumen der Tiefe eins entsprechen), Wahrscheinlichkeiten für Baumfragmente mit einer Tiefe von bis zu sechs Hierarchieebenen verwendet werden [14].

Ein weiteres Problem stochastischer Phrasenstrukturgrammatiken besteht darin, dass auch die aus einer Baumbank extrahierten Grammatikregeln eine für sprachliche Daten charakteristische Verteilung aufweisen: Wenigen sehr häufig auftretenden Regeln steht eine große Anzahl sehr selten beobachtbarer Strukturen gegenüber. So treten von den 10605 Regeln, die sich aus einer Baumbank für das Englische extrahieren lassen, 3943 nur ein einziges Mal im Korpus auf, so dass eine vernünftige Schätzung der entsprechenden Wahrscheinlichkeiten praktisch unmöglich ist. [25]. Bessere Ergebnisse werden daher mit Modellen erreicht, die die Wahrscheinlichkeit für die Erzeugung einer Regel auf der Basis von Markov-Modellen berechnen [28]. Da das resultierende Modell auf diese Weise sogar potentiell unendlich viele Grammatikregeln generieren kann, steigt seine Generalisierungsfähigkeit deutlich an. Auf englischen Daten wird dann eine Genauigkeit von ca. 91%, gemessen auf der Basis von reinen Abhängigkeitsbeziehungen, erreicht.⁶

Trennflächenklassifikatoren

Neben den probabilistischen Modellen haben in der Sprachverarbeitung zunehmend auch Entscheidungsverfahren an Bedeutung gewonnen, die auf der direkten Optimierung der Grenze zwischen zwei Entscheidungsgebieten im Merkmalsraum beruhen (z.B. *support vector machines* bzw. *online large margin learning*). Sie approximieren die Zuordnung von Strukturbeschreibungen durch einfache, oftmals sogar lineare Abbildungsvorschriften, wodurch sich die aus den Trainingsdaten extrahierten Regularitäten gut auf ungesehene Fälle übertragen lassen. Auf diese Weise wird in vielen Fällen ein sehr hohes Maß an Robustheit gegenüber der hochgradigen sprachlichen Varianz erreicht.

Trennflächenklassifikatoren können immer dann zum Einsatz kommen, wenn sich das Berechnungsproblem auf eine Klassifikationsaufgabe in einem hochdimensionalen Merkmalsraum reduzieren lässt. Dies ist z.B. beim Wortarttagging der Fall. Die Entscheidung über die optimale Sequenz von Wortarttags wird hierbei durch lokale Entscheidungen approximiert, die eine Vielzahl von Indikatoren aus dem Kontext der aktuell betrachteten Wortform berücksichtigen [41]. Dies wird dadurch ermöglicht, dass Trennflächenklassifikatoren ohne weiteres mehrere Millionen, zumeist binarisierte Merkmale verarbeiten können. Ein wesentlicher Faktor für

⁶ Zur Methodik der Qualitätsmessung für hierarchische Strukturbeschreibungen vgl. auch [21].

den erfolgreichen Einsatz derartiger Verfahren besteht dann darin, die am besten geeigneten Merkmale für die korrekte Zuordnung der Wortartentags auszuwählen.

Dieser Ansatz kann auf strukturelle Beschreibungen erweitert werden, wenn man als zusätzliche Randbedingung für die Auswahl der optimalen Beschreibung noch die formalen Eigenschaften einer Baumstruktur hinzunimmt. Dann lassen sich Trennflächenklassifikatoren auch für das (syntaktische) Parsing einsetzen. Besonders gut geeignet sind hierfür Dependenzrepräsentationen, da bei ihnen die Entscheidung über den optimalen Syntaxbaum in separate Klassifikationsaufgaben zerlegt werden kann: die Unterordnung einer Wortform unter eine andere, sowie die Wahl einer geeigneten Kantenbeschriftung, wobei jedoch bei der Wahl des korrekten Anbindungspunktes die „Klassenanzahl“ mit der Satzlänge variiert.

Diese Idee wird in [77] verfolgt, wo eine lokale Bewertung auf der Basis lokaler, d.h. kantenspezifischer Merkmale ermittelt wird, die dann in eine globale Suche nach dem optimalen, den gesamten Eingabesatz überspannenden Dependenzbaum einfließen (MSTParser). Alternativ dazu kann die Entscheidung über die optimale Strukturbeschreibung auch auf die Wahl der jeweils nächsten Parseraktion (e.g. *shift*, *reduce*, *attach to the right*, *attach to the left*) zurückgeführt werden. Greift dabei das Modell ausschließlich auf Merkmale aus der Verarbeitungshistorie des Parsers zurück, ergibt sich ein deterministisches Entscheidungsverfahren, das den Strukturbau in einem einzigen links-rechts-inkrementellen Durchlauf erzeugen kann (MaltParser, [89]).

Obwohl beide Ansätze ein vergleichbar hohes Qualitätsniveau aufweisen, zeigen sie jedoch gleichzeitig auch ein hohes Maß an Komplementarität in ihren Fehlentscheidungen, so dass durch Kombination der Verfahren in einer sequentiellen Architektur (*classifier stacking*) deutliche Synergieeffekte realisiert werden können. Dabei werden dem Parser zusätzlich zu seinen bisherigen Merkmalen auch Merkmale aus der Strukturentscheidung des jeweils alternativen Systems zur Verfügung gestellt. Auf diese Weise lässt sich Genauigkeit der vom Parser vorhergesagten Kanten (labelled accuracy ohne Berücksichtigung der Interpunktions) auf einem deutschsprachigen Korpus von 87,43% auf 88,46% bzw. von 85,82% auf 87,66% steigern [78].

Inkonsistente Regelsysteme

Probabilistische Modelle gehen generell von einer logisch konsistenten Repräsentation grammatischen Wissens beispielsweise in Form eines endlichen Automaten bzw. einer kontextfreien Grammatik aus. Die empirisch ermittelten Regelwahrscheinlichkeiten dienen dann bei der Analyse als Selektionskriterium zwischen konkurrierenden Analysealternativen. Formalismen, die auf gewichteten Constraints beruhen (WCDG bzw. Optimalitätstheorie) erfordern hingegen eine explizite Behandlung von *Konflikten*, die zwischen den verschiedenen Wohlgeformtheitsbedingungen der Grammatik auftreten können.

Hierfür sind dann Verfahren zur Constraint-Optimierung erforderlich, die unter Bezug auf ein gegebenes Bewertungsschema die optimale Lösung für ein Constraint-Satisfaction-Problem anstreben. Variablenbelegungen, die ein Constraint verletzen, werden mit Kosten belegt, können aber immer noch Bestandteil der optimalen Lösung sein, solange keine besser bewertete Alternative zur Verfügung steht. Damit besteht die Möglichkeit, dass auch extra- bzw. agrammatische Sätze noch eine Strukturbeschreibung zugewiesen bekommen. Zudem sind durch die systematische Berücksichtigung von Constraint-Verletzungen die Voraussetzungen gegeben, neben schwachen Wohlgeformtheitsbedingungen auch *unsichere* Informationsquellen direkt in die Entscheidung über die optimale Strukturbeschreibung einbeziehen zu können. Dazu gehören praktisch alle oben angesprochenen flachen Analyseverfahren. Fehlentscheidungen in ei-

ner dieser Informationsquellen führen zwar normalerweise zu einem Konflikt, der aber nicht notwendigerweise auch eine fehlerhaften Strukturzuweisung zur Folge haben muss. In einer derartigen Architektur liefern flache Analyseverfahren keine Vorentscheidung über das Gesamtresultat mehr, sondern stellen Evidenz bereit, die mit der Evidenz aus dem Kontext der Konfliktsituation abgeglichen werden muss. Unter Verwendung einer Vielzahl von flachen, probabilistischen Analysekomponenten (Wortartentagger, Chunker, Supertagger, PP-Attacher, einfacher Shift-Reduce-Parser) konnte die Analysegenauigkeit für deutsche Sätze von 68,3% (ohne Wortartentagger) über 87,7% (mit Wortartentagger) auf 91,1% (mit allen Prädiktoren)⁷ gesteigert werden [37]. Bemerkenswert ist dabei, dass dieses Ergebnis erzielt werden konnte, obwohl die Einzelkomponenten partiell widersprüchliche Hypothesen bereitstellen (z.B. der Wortartentagger gegenüber dem Supertagger oder der PP-Attacher gegenüber dem Shift-Reduce-Parser). Zudem konnte gezeigt werden, dass alle Einzelkomponenten, trotz ihrer teilweise sehr geringen Zuverlässigkeit einen positiven Beitrag zum Gesamtergebnis erbracht haben. Nimmt man als zusätzlichen Prädiktor noch den MSTParser (vgl. Abschnitt 13.4.4) hinzu, der auf den gleichen Daten eine ohnehin schon sehr hohe Genauigkeit von 89,4% erreicht, so steigt die Genauigkeit des Gesamtsystems auf 92,6% und übertrifft damit die beiden ursprünglichen Systeme deutlich [59].

Konflikte aufgrund widersprüchlicher Evidenz ergeben sich jedoch nicht nur auf einer einzelnen Ebene des sprachverarbeitenden Systems. Sie treten ebenso zwischen den sprachlichen Ebenen (Syntax vs. Semantik vs. Informationsstruktur) oder aber zwischen der sprachlichen Information und dem zur Verfügung stehenden Hintergrundwissen auf. Ihre systematische Behandlung ist insbesondere in multimodalen Umgebungen erforderlich, wo sich die Konsistenz der Informationsbeiträge aus den verschiedenen Eingabekanälen ohnehin nicht erzwingen lässt.

Über diesen Beitrag zur Steigerung der Robustheit hinaus, können verletzbare Constraints auch Diagnoseinformation über grammatische Abweichungen bzw. über Unzulänglichkeiten in der derzeitigen Strukturbeschreibung bereitstellen. Diese Information lässt sich dann sowohl in diesbezüglichen Anwendungen, etwa zum computergestützten Sprachunterricht, als auch zur Steuerung der Analyse bzw. zur Grammatikentwicklung nutzen.

Grundsätzlich lassen sich die Verfahren zur Constraint-Optimierung in drei Gruppen aufteilen:

- *konstruktive* Algorithmen, die versuchen, eine partielle Strukturbeschreibung durch sukzessive Hinzunahme zusätzlicher Strukturelemente zu kompletieren. Da der Suchraum endlich ist, kann das optimale Resultat garantiert werden, jedoch lässt sich der zu seiner Ermittlung erforderliche Rechenzeitaufwand nicht vorhersagen.
- *eliminative* Algorithmen, die davon ausgehen, dass inkonsistente bzw. gering bewertete Strukturelemente aus dem Hypothesenraum entfernt werden können. Sie erlauben zwar eine gute Vorhersage des Disambiguierungsfortschritts, allerdings hat sich herausgestellt, dass die erforderlichen lokalen Heuristiken im Falle von Grammatiken mit vielen verletzbaren Constraints nur eine sehr unsichere Entscheidungsgrundlage darstellen.
- *transformative* Algorithmen, die eine Annäherung an die optimale Lösung durch lokale Reparaturen an einer Strukturhypothese für die gesamte Äußerung erreichen.

Transformationsbasierte Verfahren sind besonders deshalb von Interesse, weil sie nicht nur eine gute Approximation des optimalen Resultats liefern, sondern darüberhinaus auch über attraktive

⁷ Jeweils *labelled accuracy* mit Berücksichtigung der Interpunktions

anytime-Eigenschaften verfügen: Dadurch, dass während der Transformationsphase zu jedem beliebigen Zeitpunkt eine komplette, wenngleich vermutlich auch noch verbesserungsfähige Lösung vorliegt, ist die Verarbeitung jederzeit unterbrechbar und die Analysequalität kann (auch dynamisch zur Laufzeit) gegen Rechenaufwand eingetauscht werden [38].

13.4.5 Maschinelles Lernen zur Wissensakquisition

Das manuelle Zusammenstellen des lexikalischen und grammatischen Wissens, das für die Verarbeitung natürlicher Sprache benötigt wird, ist nicht nur arbeitsaufwändig, sondern auch sehr fehleranfällig und kann nur bedingt arbeitsteilig erfolgen. Dabei steht der verhältnismäßig geringe Aufwand, mit dem man für stark beschränkte Sprachausschnitte rechnen muss, in deutlichem Kontrast zu den Anforderungen, die sich für größere Anwendungsdomänen bzw. gar für unrestringierte Sprachdaten ergeben. Ausgehend von diesen Erfahrungen, ist in den letzten Jahren die Wissensakquisition mit Methoden des maschinellen Lernens zur (semi-)automatischen Informationsgewinnung immer stärker in den Mittelpunkt des Interesses gerückt.

Die wesentliche Voraussetzung für eine Anwendung von Verfahren des maschinellen Lernens ist die Verfügbarkeit großer Datenmengen für das Training der Modelle. Zwar stehen natürlichsprachliche Texte inzwischen für zahlreiche Sprachen in fast unbegrenzter Menge zur Verfügung, doch eignen sich diese nur für relativ einfache Aufgaben der Sprachverarbeitung, die vorrangig auf sequentielle Abfolgebeziehungen zurückgreifen, z.B. die Sprachmodellierung (vgl. Abschnitt 13.4.4) oder das Auffinden von gemeinsamen Wortformenvorkommen (Mehrwortlexeme bzw. Kollokationen). Sprachstrukturverarbeitung hingegen erfordert überwachte Lernverfahren, für die große Mengen an Korpusdaten benötigt werden, die auf den verschiedenen sprachlichen Ebenen mit Annotationen versehen sind. Entsprechende Annotationsstandards wurden inzwischen für praktisch alle Bereiche linguistischer Strukturbeschreibungen entwickelt, angefangen von laut- bzw. schriftsprachlichen Transkriptionen über syntaktische Kategorien und Strukturen (Baumbanken), semantische Klassen, Relationen und Koreferenzbeziehungen bis hin zu den pragmatischen Aspekten der Sprecherintention (Dialogakte) und der Textstruktur.

Unter Zuhilfenahme von stochastischen oder symbolischen Lernverfahren können aus solchen Daten Modelle oder Modellkomponenten automatisch abgeleitet werden [130]. Typische Vertreter hierfür sind etwa

- stochastische Markov-Modelle bzw. konnektionistische Architekturen für die akustische Realisierung von Lauten, sowie für Wort-, Wortart- und Dialogaktsequenzen,
- stochastische Phrasenstrukturgrammatiken bzw. Trennflächenklassifikatoren für die Satzsyntax, sowie
- Entscheidungsbäume bzw. Transformationsregelmengen für unterschiedliche Disambiguierungsaufgaben.

Unter den verwendeten Lernparadigmen dominierten traditionell die stochastischen Verfahren. In den letzten Jahren sind hierzu auch noch die Trennflächenklassifikatoren hinzugekommen, während sich Arbeiten zu den anderen Methoden (z.B. genetische Algorithmen oder neuronale Netze) nur vereinzelt finden. Da die derzeit erfolgreich trainierbaren Verarbeitungskomponenten in vielen Fällen nur Teilespekte der sprachlichen Realität modellieren, besteht eine wichtige

Aufgabe in ihrer Zusammenführung zu einer komplexen, möglicherweise hybriden Gesamtarchitektur (vgl. Abschnitt 13.5).

Bei der Auswahl der auf den gegebenen Daten zu trainierenden Modelle ergibt sich in allen Fällen ein systematischer Widerspruch zwischen der gewünschten (linguistischen) Ausdruckskraft einerseits und ihrer durch die Menge der verfügbaren Daten begrenzten Trainierbarkeit. Um hier zu tragfähigen Kompromissen zu kommen, werden neben flachen Strukturbeschreibungen auch Techniken der Parameterapproximation eingesetzt. Letztere ersetzen die im Falle von ungesesehenen Ereignissen unbekannten Wahrscheinlichkeiten durch eine Interpolation benachbarter Werte (Glättung, *smoothing*) und greifen dabei auch auf generellere Beobachtungsklassen zurück (*back-off*, Kneser und Ney [60]). So kann z.B. die bedingte Wahrscheinlichkeit für ein im Training nicht beobachtbares Wortformen-Trigram auf der Basis entsprechender Bigram-Information geschätzt werden. Darüberhinaus ist es möglich, zusätzliche Trainingsdaten mit Hilfe des bereits trainierten Modells zu erzeugen (*boosting*, z.B. Wang et. al. [125]).

In einfachsten Fall (z.B. bei den Sprachmodellen) erfolgt das Schätzen der Modellparameter auf der Basis relativer Häufigkeiten für die verschiedenen Beobachtungsklassen, die sich direkt auf den Trainingsdaten auszählen lassen. Für Modelle mit verborgenen Zuständen, ist ein solches Vorgehen jedoch nicht mehr möglich. Hier kommen Verfahren zur rekursiven Parameterapproximation zum Einsatz, die auf der Grundidee der Erwartungsmaximierung (*expectation maximization*, EM-Algorithmus, Dempster et. al. [33]) beruhen. Ausgehend von einer initialen Schätzung wird über mehrere Trainingszyklen hinweg die Wahrscheinlichkeit für die Erzeugung der Trainingsdaten durch das jeweilige Modell maximiert (Baum-Welch-Training, Rabiner [95]). Der Zuwachs an Modellqualität ergibt sich dann vor allem aus einer verbesserten Zuordnung zwischen den Modellzuständen und den Beobachtungsdaten. Allerdings basiert der Ansatz auf einem Gradientenabstiegsverfahren, das die optimale Parameterschätzung nicht garantieren kann und überdies empfindlich ist gegenüber einer zu starken Anpassung an die Gegebenheiten der Trainingsdaten (*overfitting*).

13.4.6 Generierung

Auf den ersten Blick stellt sich das Generierungsproblem für natürliche Sprache als relativ einfach dar. Anders als bei der Sprachanalyse, die keinen Einfluss auf die Auswahl der sprachlichen Ausdrucksmittel hat und daher der Vielfalt der an kommenden sprachlichen Daten angemessen Rechnung tragen muss, liegt im Falle der Generierung die Initiative im sprachverarbeitenden System selbst. Daher können bereits mit relativ einfachen Mitteln, wie der Verwendung von parametrisierten Standardnachrichten oder Textbausteinen recht brauchbare Lösungen für einfache Verarbeitungsaufgaben implementiert werden.

Die spezifischen Schwierigkeiten der Sprachgenerierung treten erst dann deutlicher hervor, wenn die Verarbeitungsaufgabe eine gewisse Variabilität und damit auch Natürlichkeit der sprachlichen Ausgaben erfordert. Dies ist etwa im Bereich natürlichsprachlicher Dialogsysteme für anspruchsvolle Aufgaben (Beratung, Verhandlungsführung usw.), bei der maschinellen Übersetzung, sowie bei der Generierung (auch multimedialer) Präsentationen, Dokumente, Berichte, Zusammenfassungen oder kontextualisierter Hilfestellungen der Fall.

Ausgangspunkt der Sprachgenerierung ist immer ein zu realisierendes kommunikatives Ziel und eine geeignete Repräsentation der jeweils zu versprachlichenden Information. Dafür kommen relativ sprachnahe Strukturen in Frage, wie im Falle der maschinellen Übersetzung bzw.

der Textzusammenfassung, oder aber nichtsprachliche Informationen, wie sie als Ausgabe von Datenbank-, Informations-, Experten- oder Bildverarbeitungssystemen auftreten. Insbesondere im Zusammenhang mit Kommunikationsaufgaben im Bereich der Robotik ist hier auch mit komplexen Situationsbeschreibungen als Eingabestruktur zu rechnen.

Grundsätzlich muss jedes Sprachgenerierungssystem darüber entscheiden, welcher Inhalt versprachlicht werden muss, um das kommunikative Ziel zu erreichen, und in welcher Form dies passieren soll. Im einzelnen sind dabei die folgenden Teilschritte zu bearbeiten [96]:

- Inhaltsauswahl: Vor dem Hintergrund der Kommunikationssituation und unter Annahmen über das Vorwissen des Kommunikationspartners und seiner Perspektive auf den Kommunikationsgegenstand müssen die für das Kommunikationsziel relevanten konzeptuellen Einheiten und die Relationen, in denen sie stehen, ausgewählt werden.
- Textplanung: Die zu versprachlichen konzeptuellen Einheiten müssen in geeigneter Weise gruppiert und strukturiert werden, um einen konsistenten und nachvollziehbaren Kommunikationsbeitrag zu erzeugen. Hierbei spielt z.B. die rhetorische Struktur eine entscheidende Rolle (vgl. Abschnitt 13.4.1).
- Aggregation: Um irritierende und das Verständnis negativ beeinflussende Aufzählungen zu vermeiden, sollten einzelne Informationsbausteine zusammengefasst werden. Dies kann sowohl auf der sprachlichen Ebene durch Koordination, als auch konzeptuell durch Wahl einer angemessenen Abstraktionsebene erfolgen, z.B. im Falle einer Datenbankabfrage:

ohne Aggregation: *In der Hausbachstraße ist ein Chinarestaurant. In der Bahnhofsstraße sind zwei Chinaresteaurants. In der Gartenstraße ist ein Chinarestaurant. In der Bergstraße ist ein Chinarestaurant.*
...

sprachliche Aggregation: *In der Hausbach-, der Bahnhofs-, der Garten-, ... und der Bergstraße sind Chinaresteaurants.*

konzeptuelle Aggregation: *Im Bahnhofsviertel sind zahlreiche Chinaresteaurants.*

Zur konzeptuellen Aggregation muss dann natürlich auch entsprechendes Weltwissen hinzugezogen werden. Im Resultat dieses Verarbeitungsschrittes liegt ein Satzplan vor.

- Lexikalisierung: Als erster Schritt der Oberflächengenerierung sind die konzeptuellen Einheiten auf konkrete Lexeme der Zielsprache abzubilden. Dieser Teilschritt muss relativ frühzeitig erfolgen, da seine Ergebnisse die syntaktische Struktur des zu erzeugenden Satzes stark beeinflussen können. Zu den Entscheidungen, die dabei zu treffen sind, gehört die Auswahl zwischen unterschiedlichen Perspektiven (z.B. *kaufen* oder *verkaufen*), der Einsatz von sprachlichen Wendungen (Kollokationen, z.B. *in Gang bringen*; idiomatische Wendungen, z.B. *die Kurve kriegen*), die Auswahl einer geeigneten Granularitätsebene (*Auto* oder *Stufenhecklimousine*) und die Berücksichtigung von individuellen Einstellungen (*Sportwagen*, *Bolide* oder *Prollkarosse*).
- Festlegung von Referenzausdrücken: Die Verwendung von Referenzausdrücken ist ein wichtiges stilistisches Ausdrucksmittel, um die Textverständlichkeit zu erhöhen und Eintönigkeit zu vermeiden. Dabei ist grundsätzlich immer abzuwagen, zwischen der Forderung nach Sprachökonomie und dem Risiko von Mehrdeutigkeit. Referenzen können durch Eigennamen (z.B. *Linux*), definite Beschreibungen (z.B. *das Open-Source-Betriebssystem*) oder Proformen (z.B. *es*) etabliert werden. Welche dieser Möglichkeiten genutzt werden sollte, ist

abhängig von der Salienz des Referenzobjekts, d.h. von seiner Verfügbarkeit im Diskurskontext.

- Realisierung der Oberflächenform: Den Abschluss der Generierung bilden alle syntaktischen und morpho-syntaktischen Anpassungsoperationen, die erforderlich sind, um eine korrekte Oberflächenform für die zu versprachlichen Einheiten herzustellen. Hierzu zählen: die lineare Anordnung, das Hinzufügen von Funktionswörtern, die Flexion der Wortformen und ggf. auch ihre phonologische Anpassung an den Kontext (z.B. *hinter dem* → *hinterm*)

13.5 Architekturen für die Sprachverarbeitung

13.5.1 Modularisierung

Geht es um die Modularisierung komplexer Sprachverarbeitungsaufgaben so bietet oftmals die Aufteilung in sprachliche Beschreibungsebenen und die zwischen ihnen vermittelnden Transformationsschritte eine erste Orientierung. Dementsprechend gelten Komponenten für Syntax und Semantik gemeinhin als Standard, ggf. sind auch solche für Morphologie, Pragmatik, Prosodie usw. vorgesehen (vgl. z.B. die Systemstruktur in einem System zum Gesprächsdolmetschen [56, 103, 120]).

Eine solche Modularisierung ist vor allem aus methodischen Gründen unverzichtbar. Sie ist gleichermaßen Voraussetzung für linguistischen Erkenntnisgewinn und die Modellierung sprachlicher Phänomene, wie auch für die Organisation von Systementwurf, -evaluation und -weiterentwicklung. Gleichzeitig birgt aber eine zu rigorose Modularisierung auch ein erhebliches Risikopotential, da bei der Wahl zu stark eingeschränkter Schnittstellenprotokolle ebenenübergreifende Aspekte der Sprache nur noch unzureichend berücksichtigt werden können. Ein Beispiel für ein solches Querschnittsphänomen ist etwa mit der Prosodie gegeben, deren Einfluss auf praktisch allen Ebenen der Sprache nachweisbar ist. Ein weiteres Problemfeld ist die Abgrenzung zwischen Semantik und Pragmatik, da in vielen Fällen ohne die Kenntnis der Bedeutung einer Äußerung keine Analyse ihrer kommunikativen Funktion möglich ist, umgekehrt aber auch die pragmatische Funktion als Voraussetzung für eine erfolgreiche Bedeutungsanalyse angesehen werden muss.

Ähnlich zirkuläre Beziehungen findet man auch in anderen Bereichen der Sprachverarbeitung, etwa bei der Auflösung pronominaler Referenzen, die gleichermaßen stark auf syntaktische und semantische Information zurückgreift. So muss für die deutschen Personalpronomen einerseits eine Übereinstimmung mit dem Bezugsnomen in den syntaktischen Merkmalen Genus, Numerus und Person gegeben sein, andererseits sollte aber auch semantische Passfähigkeit vorliegen. Liegen mehrere Kandidaten für das Bezugsnomen vor, so kann sowohl eine Übereinstimmung in der syntaktischen Funktion als auch in der semantischen Rolle disambiguierend wirken. Da alle diese Kriterien nur einen mehr oder weniger stark ausgeprägten präferentiellen Charakter besitzen, müssen Syntax und Semantik hier und in vielen anderen Zusammenhängen offensichtlich als wechselseitige Voraussetzungen für einander betrachtet werden.

Weiterhin können Modularisierungsentscheidungen einen erheblichen Einfluss auf die Effizienz einer Systemlösung haben, wenn etwa wegen einer lokal nicht verfügbaren Information die Entscheidung über alternative Hypothesen auf die globale Ebene des Systems verlagert werden muss [117]. Als Ausweg aus diesem Dilemma wird oftmals eine sehr enge informationelle

Kopplung der Komponenten wie in der HPSG gewählt, die dann jedoch *de facto* auf eine monolithische Architektur hinausläuft, die nicht nur unter dem Gesichtspunkt einer arbeitsteiligen Systementwicklung, sondern auch im Hinblick auf die robuste Verarbeitung problematisch ist (vgl. Abschnitt 13.4.4).

Die Schwierigkeiten, die mit Modularisierungentscheidungen verbunden sein können, werden besonders deutlich, wenn es bei den zu verarbeitenden Daten um unsichere Hypothesen über die sprachlichen Eingaben handelt. Dies ist etwa bei der Verarbeitung gesprochener Sprache der Fall, die in den letzten Jahren verstärkt in den Mittelpunkt des Interesses gerückt ist. Dabei stellt Unsicherheit in diesem Zusammenhang eine inhärente Eigenschaft des (akustischen) Eingabekanals dar und kann nicht als ausmerzbare Artefakt betrachtet werden, wie dies etwa bei Tippfehlern oftmals der Fall ist. Die Unsicherheit betrifft überdies verschiedene Aspekte der sprachlichen Form, wie die Segmentierung des Eingabestroms, die Identität der sprachlichen Einheiten und das Vorliegen suprasegmentaler Eigenschaften (Prosodie). Zusätzlich muss bei der Verarbeitung spontan gesprochener Sprache mit dem massiven Auftreten von Performanzphänomenen (Häsitationen (*ähm*), Abbrüchen, Korrekturen usw.) gerechnet werden, die erhebliche Abweichungen vom Ideal wohlgeformter sprachlicher Äußerungen mit sich bringen: *Ich möchte am äh, dreizehnten, nein zwölften, also übermorgen, nach Berlin fahren, über Hannover.*

Rein sequentielle Architekturen, bei denen die Daten nacheinander eine Reihe von Verarbeitungsstufen durchlaufen, führen zwangsläufig auf das Problem der Fehlerfortpflanzung und -verstärkung. Zudem sind für die Verarbeitung allein schon wegen des ebenenübergreifenden Charakters prosodischer Information nicht mehr angemessen. Überdies weiß man vom menschlichen Vorbild, dass eine wirksame Strategie zur Kompensation der gravierenden Unsicherheit in den Eingabedaten in einer stark erwartungsgesteuerten Verarbeitung besteht. Ein derartiger Verarbeitungsmodus wird derzeit allerdings nur in monolithischen Systemen recht gut beherrscht. So basieren z.B. die üblichen Verfahren zur Worterkennung auf Erkennungshypothesen, die aus dem Aussprachewörterbuch (ggf. unter weiterer Einschränkung durch ein Sprachmodell) generiert und anschließend am Sprachsignal verifiziert werden. Solange hierbei die verfügbaren Erwartungen an die Eingabedaten stark genug sind, können auf diese Weise sensorische Unsicherheit und extragrammatische Konstruktionen in erheblichem Umfang kompensiert werden. Ein ähnliches Verarbeitungsprinzip liegt dem Top-Down-Parsing (siehe Abschnitt 13.4.3) bei der syntaktischen Analyse von idealem sprachlichen Input zugrunde.

Weitgehend unklar ist jedoch noch, inwieweit man eine solche Verarbeitungsstrategie auch auf komplexe Systeme übertragen kann. Hier hat man es üblicherweise mit mehreren potentiellen Quellen für Erwartungen zu tun (Dialogmodell, Weltmodell, Grammatik usw.) und es ist nicht ohne weiteres klar, wie man diese (möglicherweise widersprüchlichen) Informationsbeiträge geeignet kombinieren und zu den relevanten Entscheidungspunkten propagieren kann. Ein populärer Ansatz besteht in der Kommunikation von Entscheidungsalternativen etwa in Form von Worthypothesengraphen [90] (vgl. Bild 13.10). Ein solcher Graph enthält dann leicht einige zehntausend Erkennungshypothesen pro Äußerung und seine Weiterverarbeitung erfordert sehr leistungsfähige Algorithmen zur strukturellen Analyse [98]. Eine Alternative hierzu besteht möglicherweise in der gezielten Etablierung geeigneter Interaktionspfade zwischen den betroffenen Komponenten [17]. Überzeugende Lösungen stehen aber noch aus.

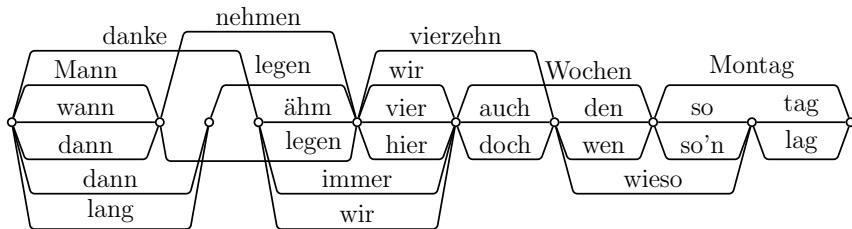


Abbildung 13.10: Ein einfacher Worthyhypothesengraph für die Äußerung *Dann nehmen wir doch den Montag*.

Eine wesentliche Voraussetzung für robustes Systemverhalten ist schließlich das Vorhandensein struktureller Redundanz innerhalb einer lose gekoppelten Systemarchitektur. Hierfür sind mehrere alternative und voneinander unabhängige Verarbeitungsstränge erforderlich wie z.B.

- syntaktisch vs. semantisch orientierte Analyse,
- probabilistische vs. klassische Ansätze,
- regelbasierte vs. beispielbasierte Verfahren oder
- akustischer vs. visueller Eingabekanal,

aus deren im Einzelfall differierenden Berechnungsresultaten eine geeignete Auswahl getroffen werden muss [121]. Kann mit komplementären Informationsbeiträgen gerechnet werden, sollte eine Integration von Teilergebnissen angestrebt werden [39, 81, 99]. In einigen Fällen konnte so die Leistungsfähigkeit des Gesamtsystems über die der Einzelkomponenten hinaus gesteigert werden [113]. Ein solcher Synergieeffekt stellt sich jedoch nicht zwangsläufig ein [23].

13.5.2 Inkrementelle Verarbeitung

Ein weiterer Aspekt des Architekturentwurfs, der sich in der Komponentenaufteilung nicht direkt widerspiegelt, ist die zeitliche Charakteristik der Modulinteraktion. Klassische Ansätze zur Mensch-Maschine-Interaktion greifen hier ausnahmslos auf explizite zeitliche Grenzmarker (Enter-Taste, Mausklick, Sprechtaste usw.) zurück, wodurch insbesondere die Phase der Informationseingabe von der Phase der Informationsverarbeitung klar getrennt werden soll. Abgesehen davon, dass ein solches Vorgehen im Vergleich zur zwischenmenschlichen Kommunikation recht artifiziell erscheint, können leistungsfähige Lösungen zur Echtzeitverarbeitung erst erreicht werden, wenn Eingabe, Verarbeitung und Ausgabe auf allen Beschreibungsebenen der Sprache zeitlich überlappend erfolgen kann. Sprachverarbeitende Systeme, die dieser Forderung nach strikter temporaler Inkrementalität genügen, weisen dann eine Reihe von Vorteilen auf [4, 46, 126].

- Strukturelles Echtzeitverhalten: Dank der zeitlichen Überlappung von Eingabe und Analyse kann die Sprechzeit des Dialogpartners bereits zur Verarbeitung (Sprachverständnis und Handlungsplanung) genutzt werden. Gleichermaßen gilt auch für den Ausgabekanal: Frühe Teile einer Äußerung stehen bereits zur Versprachlichung bereit, während für spätere noch Verbalisierungsprozesse ablaufen. Längere monologische Beiträge lassen sich wegen des prinzipiell beschränkten Speicherplatzes ohnehin nur inkrementell produzieren und analysieren.

- Teilnahme an natürlichen Dialogen: Unmittelbare Reaktion und ggf. auch die Übernahme der Gesprächsinitiative ist nur möglich, wenn zumindest Sprachverständigen und Handlungsplanung nebenläufig erfolgen. Wegen der informationellen Abhängigkeiten zwischen den Komponenten einer Gesamtarchitektur, gilt diese Forderung dann natürlich für alle Ebenen der Sprachverarbeitung.
- Effiziente Verarbeitung: In modularen Architekturen können bestimmte Teilaufgaben von Sprachverständigen und -produktion partiell nebenläufig realisiert werden (z.B. Worterkennung, syntaktische Analyse, semantische Interpretation, Transfer, Generierung).

Problematisch ist jedoch, dass bei der Verarbeitung der rechte Kontext nicht mehr verfügbar ist. Daher entfällt auch die Möglichkeit zur globalen Optimierung für eine komplette Äußerung. Das somit unvermeidlich erhöhte Risiko für lokale Fehlentscheidungen kann nur durch stark erweiterte Suchräume bzw. geeignete Korrekturmechanismen kompensiert werden. Auf der anderen Seite bietet erst eine inkrementelle Analyse überhaupt die Voraussetzungen, um dynamisch generierte Erwartungen an den rechten Kontext zur Steigerung der Robustheit einzusetzen.

Schlangen und Skantze [101] beschreiben eine generelle Architektur für die inkrementelle Verarbeitung in einem Dialogsystem. Sie basiert auf der Möglichkeit zur Kommunikation und expliziten Revision von Hypothesen in einem System aus mehreren Komponenten, die über referentiell gekoppelte Ein-/Ausgabepuffer miteinander verbunden sind. Die Architektur wurde bereits erfolgreich für einfache interaktive Sprachdialogaufgaben (z.B. zur Eingabe von Ziffernfolgen mit verzögerungsfreier Korrekturmöglichkeit bzw. Feinpositionierung von grafischen Objekten in einer virtuellen Umgebung) eingesetzt. In diesen Fällen konnte die Natürlichkeit der Mensch-Computer-Interaktion deutlich gesteigert werden. Eine Einbeziehung von Erwartungshypothesen ist in der konkreten Umsetzung derzeit aber noch nicht möglich.

13.5.3 Multimodale Kommunikation

Eine wesentliche Herausforderung für die Realisierung sprachverarbeitender Systeme ist deren erfolgreiche Einbettung in komplexe Nutzungsschnittstellen zur multimodalen Kommunikation. In einem solchen Umfeld ist es nicht mehr möglich, von bestimmten Aspekten der Sprache zu abstrahieren, sondern es ist im Gegenteil eine explizite Behandlung für eine Reihe von Fragen erforderlich, z.B.

- die Fusion von sensorischer Evidenz aus mehreren Eingabekanälen,
- die Repräsentation von sprachlichen Einheiten und Domänenobjekten,
- die Modellierung des Dialogpartners im Hinblick auf sein Wissen, sowie seine Interessen, Intentionen und Bewertungen,
- die Auflösung bzw. Herstellung sprachlicher und außersprachlicher Referenzen,
- die zeitliche Koordination von Ereignissen in unterschiedlichen Informationskanälen,
- die Auswahl der optimalen Modalität für die Informationsvermittlung usw.

Daher zählen solche Szenarien auch zu den großen Herausforderungen der KI-Forschung insgesamt. Eine typische Beispielanwendung ist etwa die Kommunikation mit einem autonomen Roboter, der neben der Spracheingabe über alternative sensorische Eingabekanäle verfügt [84].

Auf der Generierungsseite ergeben sich vergleichbare Anforderungen bei der multimodalen Informationspräsentation [5, 122].

Über den unmittelbaren praktischen Aspekt hinaus sind derartige Untersuchungen von ganz prinzipieller Bedeutung für solch grundlegende Fragestellungen wie die nach der Repräsentation des eigenen Körpers im Verhältnis zur Umwelt (Embodiment), die z.B. für eine adäquate Behandlung des Phänomens der Metaphorik ganz entscheidend ist [8].

Literaturverzeichnis

- [1] Alexandersson, J., Engel, R., Kipp, M., Kocha, S., Küssner, U., Reithinger, N., und Stede, M. (2000). Modeling negotiation dialogs. In Wahlster, W., editor, *Verbomobil: Foundations of Speech-to-Speech Translation*, pages 441–451. Springer-Verlag.
- [2] Allgayer, J., Harbusch, K., Kobsa, A., Reddig, C., Reithinger, N., und Schmauks, D. (1989). XTRA: A natural-language access system to expert systems. *Int. Journal of Man-Machine Studies*, 31:161–195.
- [3] Allgayer, J., Kobsa, A., Reddig, C., und Reithinger, N. (1990). PRACMA: PRocessing Arguments between Controversially-Minded Agents. In *Proc. of the Fifth Rocky Mountain Conference on Artificial Intelligence: Pragmatics in Artificial Intelligence*, pages 63–68, Las Cruces, NM.
- [4] Amstrup, J. W. (1999). *Incremental Speech Translation*, volume 1735 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin.
- [5] André, E., Müller, J., und Rist, T. (1996). WIP/PPP: Automatic generation of personalized multimedia presentations. In *Proc. of Multimedia 96, 4th ACM International Multimedia Conference*, pages 407–408, Boston, MA.
- [6] Androutsopoulos, I., Ritchie, G. D., und Thanisch, P. (1995). Natural language interfaces to databases – an introduction. *Natural Language Engineering*, 1(1):29–82.
- [7] Baeza-Yates, R. und Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. ACM Press, New York.
- [8] Bailey, D., Feldman, J., Narayanan, S., und Lakoff, G. (1997). Modelling embodied lexical development. In *Proc. 19th Annual Meeting of the Cognitive Science Society*, Stanford.
- [9] Bangalore, S. und Joshi, A. K. (1999). Supertagging: an approach to almost parsing. *Computational Linguistics*, 25(2):237–265.
- [10] Bar-Hillel, Y. (1953). A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58.
- [11] Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Boston.
- [12] Bernsen, N. O., Dybkjær, H., und Dybkjær, L. (1997). *Designing Interactive Speech Systems*. Springer-Verlag, London.
- [13] Blackburn, P. und Bos, J. (2005). *Representation and Inference for Natural Language – A First Course in Computational Semantics*. CSLI publications, Stanford University.
- [14] Bod, R. (1998). *Beyond Grammar: An Experience-Based Theory of Language*. CSLI Publications, Cambridge University Press.
- [15] Brants, T. (2000). TnT – A statistical part-of-speech tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, pages 224–231, Seattle, WA.
- [16] Bresnan, J. (2001). *Lexical-Functional Syntax*. Blackwell Publishers, Oxford.

- [17] Briscoe, E. J. (1987). *Modelling Human Speech Comprehension. A Computational Approach*. Ellis Horwood, Chichester.
- [18] Bröker, N. (1998). Separating surface order and syntactic relations in a dependency grammar. In *Proc. 17th Int. Conference on Computational Linguistics, COLING-ACL-1998*, pages 174–180.
- [19] Brown, P. F., Cocke, J., Pietra, S. A. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L., und Roossin, P. S. (1990). A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.
- [20] Carpenter, B. (1992). *The logic of typed feature structures*. Cambridge Univ. Press, Cambridge.
- [21] Carroll, J., Briscoe, E., und Sanfilippo, A. (1998). Parser evaluation: A survey and a new proposal. In *Proc. 1st Int. Conference on Language Resources and Evaluation*, pages 447–454, Granada, Spain.
- [22] Carstensen, K.-U., Ebert, C., Endriss, C., Jekat, S., Klabunde, R., und Langer, H., editors (2010). *Computerlinguistik und Sprachtechnologie – Eine Einführung*. Spektrum Akademischer Verlag, Heidelberg, Berlin, 3. auflage edition.
- [23] Cavar, D., Küssner, U., und Tidhar, D. (2000). From off-line evaluation to on-line selection. In Wahlster, W., editor, *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 599–612. Springer-Verlag, Berlin etc.
- [24] Charniak, E. (1993). *Statistical Language Learning*. MIT Press, Cambridge, MA.
- [25] Charniak, E. (1996). Tree-bank grammars. Technical Report CS-96-02.
- [26] Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proc. 1st Meeting of the North American Chapter of the ACL, NAACL-2000*, Seattle, WA.
- [27] Chomsky, N. (1981). *Lectures on Government and Binding*. Kluwer, Dordrecht.
- [28] Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Processing*. PhD dissertation, University of Pennsylvania, Philadelphia.
- [29] Comeford, L., Frank, D., Gopalakrishnan, P., Gopinath, R., und Sedivy, J. (2001). The IBM personal speech assistant. In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, ICASSP-2001*, pages I1–I4.
- [30] Cowie, J. und Lehnert, W. (1996). Information extraction. *Communications of the ACM*, 39(1):51–87.
- [31] Cutting, D., Karger, D., Pedersen, J., und Tukey, J. W. (1992). Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proc. 15th Annual International ACM/SIGIR Conference*.
- [32] Debusmann, R., Duchier, D., und Kuhlmann, M. (2004). Multi-dimensional graph configuration for natural language processing. In *Proceedings of the International Workshop on Constraint Solving and Language Processing*, Roskilde, Denmark. Springer.
- [33] Dempster, A., Laird, N., und Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.
- [34] Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L., Strassel, S., und Weischedel, R. (2004). The automatic content extraction (ace) program – tasks, data, and evaluation. In *Proc. 4th International Conference on Language Resources and Evaluation, LREC-2004*, Lisbon, Portugal.
- [35] Feng, J., Johnston, M., und Bangalore, S. (2011). Speech and multi-modal interaction in mobile search. *IEEE Signal Processing Magazine*, 28(4):40–49.

- [36] Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J., Schlaefler, N., und Welty, C. (2010). Building Watson: An overview of the DeepQA project. *AI Magazine*, 31(3):59–79.
- [37] Foth, K. und Menzel, W. (2006). Hybrid parsing: Using probabilistic models as predictors for a symbolic parser. In *Proc. 21st Int. Conf. on Computational Linguistics, Coling-ACL-2006*, Sydney.
- [38] Foth, K. A., Menzel, W., und Schröder, I. (2000). A Transformation-based Parsing Technique with Anytime Properties. In *4th Int. Workshop on Parsing Technologies, IWPT-2000*, pages 89–100, Trento, Italy.
- [39] Frederking, R. und Nirenburg, S. (1994). Three heads are better than one. In *Proceedings 4th Conf. on Applied Natural Language Processing*, pages 95–100, Stuttgart.
- [40] Frege, G. (1892). Über Sinn und Bedeutung. In Agnelli, I., editor, *1967, Kleine Schriften*, pages 143–162. Darmstadt.
- [41] Giménez, J. und Màrquez, L. (2004). SVMTool: A general POS tagger generator based on Support Vector Machines. In *Proc. 4th International Conference on Language Resources and Evaluation, LREC-2004*, pages 43–46, Lisbon, Portugal.
- [42] Grishman, R. und Sundheim, B. (1996). Message understanding conference – 6: A brief history. In *Proceedings 16th International Conference on Computational Linguistics, Coling '96*, pages 466–471, Copenhagen, Denmark.
- [43] Habel, C. (1986). *Prinzipien der Referentialität : Untersuchungen zur propositionalen Repräsentation von Wissen*. Springer-Verlag, Berlin.
- [44] Hahn, U. und Schnattinger, K. (1998). A text understander that learns. In *Proc. 36th Annual Meeting of the ACL and 17th Int. Conf. on Computational Linguistics*, pages 476–482, Montreal.
- [45] Harabagiu, S., Moldovan, D., Pasca, M., Mihalcea, R., Surdeanu, M., Bunescu, R., Girju, R., Rus, V., und Morarescu, P. (2000). Falcon – boosting knowledge for answer engines. In *9th Text Retrieval Conference (Trec-9)*, Gaithersburg, Maryland.
- [46] Harbusch, K., Finkler, W., und Schauder, A. (1991). Incremental syntax generation with tree adjoining grammars. In Brauer, W. und Hernandez, D., editors, *Verteilte Künstliche Intelligenz und kooperatives Arbeiten: 4. Internationaler GI-Kongreß Wissensbasierte Systeme, Proc.*, pages 363–374. Springer, Berlin, Heidelberg.
- [47] Harris, L. R. (1977). Robot: A high performance natural language data base query system. In *Proc. of the 5th IJCAI*, pages 903–904, Cambridge, MA.
- [48] Herzog, O., editor (1991). *Text understanding in LILOG: Integrating computational linguistics and artificial intelligence*. Springer-Verlag, Berlin.
- [49] Hutchins, W. J. und Somers, H. L. (1992). *An introduction to machine translation*. Academic Press, London.
- [50] Jackendoff, R. (1977). *X Syntax: A Study of Phrase Structure*. Linguistic Inquiry. MIT Press, Cambridge MA.
- [51] Jekat, S. J. und Klein, A. (1996). Machine interpretation: Open problems and some solutions. *Interpreting: International Journal of Research and Practice in Interpreting*, 1.
- [52] Joshi, A. K. (1987). Introduction to tree adjoining grammar. In Ramer, A. M., editor, *The Mathematics of Language*, pages 87–114. J. Benjamins.
- [53] Jurafsky, D. und Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, Upper Saddle River, NJ, 2nd edition edition.
- [54] Kamp, H. und Reyle, U. (1992). *From Discourse to Logic*. Kluwer, Dordrecht.

- [55] Kanal, L. und Kumar, V. (1988). *Search in Artificial Intelligence*. Springer-Verlag, Berlin.
- [56] Karger, R. und Wahlster, W. (1997). Verbmobil: Multilinguale Verarbeitung von Spontansprache. *KI*, (4):41–45.
- [57] Karlsson, F., Voutilainen, A., Heikkilä, J., und Anttila, A., editors (1995). *Constraint Grammar – A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter, Berlin, New York.
- [58] Keller, F. (2000). *Gradience in Grammar: Experimental and Computational Aspects of Degrees of Grammaticality*. PhD thesis, University of Edinburgh.
- [59] Khmylko, L., Foth, K., und Menzel, W. (2009). Co-parsing with competitive models. In *Proc. 11th Int. Conf. on Parsing Technologies, IWPT-2009*, pages 99–107, Paris.
- [60] Kneser, R. und Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Proc. IEEE Int. Conference on Acoustics, Speech and Signal Processing*, pages 181–184.
- [61] Knight, K. und Marcu, D. (2002). Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1).
- [62] Kobsa, A. (1985). *Benutzermodellierung in Dialogsystemen*. Springer, Berlin, Heidelberg.
- [63] Kölln, M. E. (1997). *Textproduktion als intentionaler, benutzerorientierter Prozess*. infix, Sankt Augustin.
- [64] Koskenniemi, K. (1983). Two-level model for morphological analysis. In *IJCAI*, pages 683–685.
- [65] Kuhn, J. und Rohrer, C. (1997). Approaching ambiguity in real-life sentences – the application of an optimality theory-inspired constraint ranking in a large-scale lfg grammar. In *Proc. 6. Fachtagung der Sektion Computerlinguistik der DGfS*, Heidelberg.
- [66] Lakoff, G. (1987). *Women, fire, and dangerous things: What categories reveal about the mind*. University of Chicago Press, Chicago.
- [67] Langer, H. (2001). *Parsing-Experimente: praxisorientierte Untersuchungen zur automatischen Analyse des Deutschen*. Peter Lang, Frankfurt am Main u.a.
- [68] Lehnert, W. und Sundheim, B. (1991). A performance evaluation of text-analysis technologies. *AI Magazin*.
- [69] Lin, D. (1998). Dependency-based evaluation of minipar. In *LREC Workshop on the Evaluation of Parsing Systems*, Granada, Spain.
- [70] Maes, P. (1994). Agents that reduce work and information overload. *Communications of the ACM*, 37(7).
- [71] Mann, W. C. und Thompson, S. A. (1988). Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.
- [72] Manning, C. D. und Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.
- [73] Marcu, D. (1997). From discourse structures to text summaries. In *Proc. ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization*, pages 82–88, Madrid, Spain.
- [74] Markert, K. und Hahn, U. (1997). On the interaction of metonymies and anaphora. In *Proc. of the 15th International Conference on Artificial Intelligence, IJCAI-97*, pages 1010–1015, Nagoya/Japan.
- [75] Maruyama, H. (1990). Structural disambiguation with constraint propagation. In *Proceedings 28th Annual Meeting of the ACL*, pages 31–38.

- [76] McCoy, K. F. (1998). Interface and language issues in intelligent systems for people with disabilities. In Mittal, V. O., Yanko, H. A., Aronis, J., und Simpson, R., editors, *Assistive Technology and Artificial Intelligence*, pages 1–11. Springer-Verlag, Berlin.
- [77] McDonald, R. (2006). *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. PhD thesis, University of Pennsylvania.
- [78] McDonald, R. und Nivre, J. (2011). Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(1):197–230.
- [79] Mellish, C., Reiter, E., und Levine, J. (1996). Natural language generation applications to technical documentation: A view through IDAS. In Adorni, G. und Zock, M., editors, *Trends in Natural Language Generation: An Artificial Intelligence Perspective. Proc. 4th European Workshop on Natural Language Generation, EWNLG '93*, Lecture Notes in Artificial Intelligence, pages 368–382, Berlin. Springer-Verlag.
- [80] Menzel, W. (1992). *Modellbasierte Fehlerdiagnose in Sprachlehrsystemen*, volume 24 of *Sprache und Information*. Niemeyer, Tübingen.
- [81] Menzel, W. (1995). Robust parsing of natural language. In *KI-95: Advances in Artificial Intelligence*, pages 19–34, Berlin. Springer-Verlag.
- [82] Menzel, W. (2004). Errors, intentions, and explanations: Feedback generation for language tutoring systems. In *Proc. Int. Conf. InSTIL/ICALL-2004*, pages 75–82, Venice, Italy.
- [83] Montague, R. (1974). Formal philosophy. In Thomason, R. H., editor, *Selected Papers of Richard Montague*. New Haven and London.
- [84] Moratz, R., Eikmeyer, H., Hildebrandt, B., Kummert, F., Rickheit, G., und Sagerer, G. (1995). Integrating speech and selective visual perception using a semantic network. In *AAAI-95 Fall Symposium on Computational Models for Integrating Language and Vision*.
- [85] Nagel, H.-H. (1994). AI approaches towards sensor-based driver support in road vehicles. In *KI-94: Advances in Artificial Intelligence*, pages 1–15, Berlin. Springer-Verlag.
- [86] Nerbonne, J. und Smit, P. (1996). GLOSSER-RuG: in support of reading. In *Proceedings 16th International Conference on Computational Linguistics, Coling '96*, pages 830–835, Copenhagen, Denmark.
- [87] Newell, A., Langer, S. A., und Hickey, M. (1998). The role of natural language processing in alternative and augmentative communication. *Natural Language Engineering*, 4(1):1–16.
- [88] Nirenburg, S., Carbonell, J., Tomita, M., und Goodman, K. (1992). *Machine Translation: A Knowledge-Based Approach*. Morgan Kaufmann, San Mateo, CA.
- [89] Nivre, J. (2006). *Inductive Dependency Parsing*, volume 34 of *Text, Speech and Language Technology*. Springer-Verlag, Dordrecht, The Netherlands.
- [90] Oerder, M. und Ney, H. (1993). Word graphs: An efficient interface between continuous-speech recognition and language understanding. In *Int. Conf. on Acoustics, Speech and Signal Processing*, pages 119–122, Minneapolis.
- [91] Pang, B. und Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135.
- [92] Pollard, C. und Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago.
- [93] Prince, A. und Smolensky, P. (1993). Optimality theory: Constraint interaction in generative grammar. Technical Report 2.

- [94] Pustejovsky, J. und Boguraev, B. (1993). Lexical knowledge representation and natural language processing. *Artificial Intelligence*, 63:193–223.
- [95] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*.
- [96] Reiter, E. und Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press.
- [97] Ronthaler, M. (1998). Osiris: Qualitative Fortschritte bei der Literaturrecherche. In Dassow, J. und Kruse, R., editors, *Informatik '98: Informatik zwischen Bild und Sprache*, pages 171–180. Springer, Berlin.
- [98] Ruland, T., Rupp, C. J., Spilker, J., Weber, H., und Worm, K. L. (1998). Making the most of multiplicity: A multi-parser multi-strategy architecture for the robust processing of spoken language. In *Proc. Int. Conf. on Spoken Language Processing*, Sydney.
- [99] Rupp, C. J., Spilker, J., Klärner, M., und Worm, K. L. (2000). Combining analyses from various parsers. In Wahlster, W., editor, *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 311–321. Springer-Verlag, Berlin etc.
- [100] Schilder, F. (2002). Robust discourse parsing via discourse markers, topicality and position. *Natural Language Engineering*, 8(2):235–255.
- [101] Schlangen, D. und Skantze, G. (2011). A general, abstract model of incremental dialogue processing. *Dialogue & Discourse*, 2(1):83–111.
- [102] Schoelles, M. und Hamburger, H. (1997). The NLP role in animated conversation for CALL. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 127–134, Washington, DC.
- [103] Schwinn, J. und Bub, T. (1997). Die Rolle der Architektur: Software-Engineering in großen Projekten am Beispiel Verbmobil. *KI*, (4):45–51.
- [104] Searle, J. R. (1979). A taxonomy of speech acts. In *Expression and Meaning*. Cambridge University Press.
- [105] Shieber, S. M. (1986). *An Introduction to Unification-Based Approaches to Grammar*. CSLI Lecture Notes. Stanford, CA.
- [106] Skut, W. und Brants, T. (1998). Chunk tagger – statistical recognition of noun phrases. In *ESSLLI-98 Workshop on Automated Acquisition of Syntax and Parsing*, Saarbrücken.
- [107] Skut, W., Krenn, B., Brants, T., und Uszkoreit, H. (1997). An annotation scheme for free word order languages. In Jacobs, P., editor, *Proc. 5th Conference on Applied Natural Language Processing (ANLP-1997)*, Washington D.C. Morgan Kaufmann Publishers.
- [108] Steinbiss, V., Ney, H., Haeb-Umbach, R., Tran, B.-H., Essen, U., Kneser, R., Oerder, M., Meier, H.-G., Aubert, X., Dugast, C., und Geller, D. (1993). The Philips research system for large-vocabulary continuous-speech recognition. In *Proc. 3rd Europ. Conf. on Speech Communication and Technology, Eurospeech '93*, pages 2125–2128, Berlin.
- [109] Sundheim, B. (1995). MUC-6 named entity task definition, version 2.1. In *Proceedings 6th Message Understanding Conference (MUC-6)*, Columbia, Maryland.
- [110] Taboada, M., Brooke, J., Tofiloski, M., Voll, K., und Stede, M. (2011). Lexicon-based methods for sentiment analysis. *Computational Linguistics*, 37(2):267–307.
- [111] Thuramair, G. (1990). Parsing for grammar and style checking. In *Proceedings 13th International Conference on Computational Linguistics, Coling '90*, pages 365–370, Helsinki, Finnland.
- [112] Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59:433–560.

- [113] van Halteren, H., Zavrel, J., und Daelemans, W. (1998). Improving data driven wordclass tagging by system combination. In *Proc. of the 36th Conf. of the ACL*, pages 491–497, Montreal.
- [114] Vandeventer, A. (2000). Diagnostic d’erreurs grammaticales par relachement de contraintes dans le cadre de l’ELAO. In *Proc. TALN-2000*, pages 357–366.
- [115] VanLehn, K., editor (1991). *Architectures for Intelligence*. Lawrence Erlbaum, Hillsdale, NJ.
- [116] Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, IT-13:260–269.
- [117] von Hahn, W. (1992). Von der Verknüpfung zur Integration: Kontrollstrategie oder kognitive Architektur. In Görz, G., editor, *Konvens 92: 1. Konferenz „Verarbeitung natürlicher Sprache“*, pages 1–10. Springer, Berlin, Heidelberg.
- [118] von Hahn, W., Höppner, W., Jameson, A., und Wahlster, W. (1980). The anatomy of the natural language dialogue system HAM-RPM. In Bolc, L., editor, *Natural Language Based Computer Systems*, pages 119–253. Akademie-Verlag, Berlin.
- [119] Vosse, T. (1992). Detecting and correcting morpho-syntactic errors in real texts. In *Proceedings of the 3rd Conference on Applied Natural Language Processing*, pages 111–118, Trento.
- [120] Wahlster, W., editor (2000). *Verbmobil: Foundations of Speech-to-Speech Translation*. Springer-Verlag, Berlin etc.
- [121] Wahlster, W., Bub, T., und Waibel, A. (1997). Verbmobil: The combination of deep and shallow processing for spontaneous speech translation. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, pages 71–74, München.
- [122] Wahlster, W., Reithinger, N., und Blocher, A. (2001). Smartkom: Towards multimodal dialogues with anthropomorphic interface agents. In *Proceedings of the Human Computer Interaction Status Conference*, Saarbrücken.
- [123] Waibel, A. (1996). Interactive translation of speech. *IEEE Computer*, 29(7).
- [124] Waltz, D. L. (1978). An english language question answering system for a large relational data base. *Communications of the ACM*, 21(7):526.
- [125] Wang, W., Harper, M. P., und Stolcke, A. (2003). The robustness of an almost-parsing language model given errorful training data. In *Proc. IEEE Int. Conference on Acoustic, Speech and Signal Processing*, Hongkong.
- [126] Weber, H., Amtrup, J. W., und Spilker, J. (1997). Innovative Systemarchitekturen zur inkrementellen interaktiven Verarbeitung. *KI*, (4):26–30.
- [127] Weizenbaum, J. (1963). ELIZA – a computer program for the study of natural language communication. *Communications of the ACM*, 9(1):36–45.
- [128] Winograd, T. (1972). *Understanding Natural Language*. Academic Press, New York.
- [129] Wittgenstein, L. (1993). *Philosophical Investigations*. Blackwell, Oxford.
- [130] Young, S. und Blothooft, G., editors (1997). *Corpus-based methods in language and speech processing*. Kluwer Academic, Dordrecht.
- [131] Zweig, G. G. (1998). *Speech recognition with dynamic Bayesian Networks*. PhD thesis, University of California, Berkeley.

14 Multiagentensysteme

Franziska Klügl

Multiagentensysteme bilden das grundlegende Konzept, intelligente Entitäten (Software, Hardware, Menschen) zu einem kohärenten Gesamtsystem zu vernetzen. Zur individuellen Intelligenz kommt die soziale. Die Entitäten interagieren, kommunizieren, teilen Information, kooperieren, koordinieren ihre Aktionen. Aus dem Blickwinkel der Künstlichen Intelligenz liegen die Ursprünge der Multiagentensysteme in der Verteilten Künstlichen Intelligenz, die sich ab etwa 1980 mit der Idee konstitutionalisiert hat, Koordination und Interaktion von intelligenten Problemlösern auf einer abstrakteren Ebene jenseits von technischen, maschinennahen Problemen der Parallelität zu behandeln [90]. Während in den ersten 20 Jahren die Unterscheidung in „Verteilte Problemlöser“ und „Multiagentensystem“ noch wichtig war (siehe [34] für eine Diskussion), bezeichnet man mittlerweile – nach dem Hype der 90iger Jahre – jede Form von System mit mehreren, interagierenden „Agenten“ als Multiagentensystem. Heute erhebt auch nicht mehr nur die Künstliche Intelligenz den Anspruch, Beiträge zur Multiagentensystem-Forschung zu liefern, sondern auch Verteilte Systeme als technische Grundlage, Wirtschaftsinformatik mit ihrer Beziehung zu den Wirtschafts- und Sozialwissenschaften als Ideengeber für Metaphern, Interaktionsprotokolle oder Analyse oder das Software Engineering.

So identifizierten [92] eine Revolution im Software Engineering indem sie vier Charakteristika von modernen Systemen beschrieben: Situiertheit, Offenheit, lokale Kontrolle und lokale Interaktion, also genau die Aspekte, die Multiagentensysteme charakterisieren.

Beispiele für Multiagentensysteme findet man in allen Anwendungsgebieten, bei denen diese Charakteristika wichtig sind:

- Systeme, bei denen Software- oder Hardware-Entitäten in einer Umwelt existieren und mit dieser und anderen Entitäten interagieren;
- bei denen die einzelnen Entitäten ihre eigenen Ziele verfolgen und neue Entitäten von außen dazukommen können;
- eine Verteiltheit von Kontrollen und Daten inhärent gegeben ist oder es sinnvoll ist eine solche Verteilung einzuführen, weil die Komplexität nicht mehr durch eine zentrale Steuerung behandelt werden kann;
- bei denen parallele, asynchrone Prozesse aktiv sind.

Mit solchen Charakteristika werden eine Vielzahl moderner Anwendungsgebiete beschrieben, von Ambient Intelligence (Umgebungsintelligenz) zu Verkehrsmanagement, von modernen Fabriken zu Werkzeugen für Soziale Netzwerke.

Es gibt mittlerweile eine Reihe von sehr guten Kompendien und Monographien, die sich für eine weiterführende Einführung eignen. [89]¹ versammelte einführende Kapitel zu allen wichtigen Themen.

¹ Eine zweite Edition erscheint 2013.

tigen Teilgebieten. [90] ist ein hervorragendes, breites Lehrbuch. [80] führt sehr fundiert in den aktuellen Stand der Verteilten Entscheidungsfindung ein. Eine konzentrierte, formalere Einführung bietet [87]. Darüber hinaus gibt es sehr viele Bücher und Artikel, die sich mit speziellen Teilgebieten der Multiagentensystem-Forschung befassen.

Dieses Kapitel kann nur einen oberflächlichen Einblick zu Multiagentensystemen liefern. Dabei wollen wir wie folgt vorgehen: Zunächst soll der Begriff des „Agenten“ erklärt werden, welche Eigenschaften damit verbunden sind und mit welchen Architekturen für einen Agenten diese Eigenschaften erreicht werden können. Danach werden Agenten zu Multiagentensystemen zusammengefügt. Im Abschnitt 14.2 wird näher betrachtet, wie Agenten interagieren können und wie diese Interaktion strukturiert werden kann. Dabei gehen wir zunächst von Agenten aus, die als kooperierende Problemlöser konzipiert sind. Im darauf folgenden Abschnitt werden verschiedene Interaktionsprotokolle dargestellt, mit denen egoistische Agenten gemeinsam Entscheidungen treffen können, um so auch bei nicht kooperativen Agenten zu einer sinnvollen Problemlösung zu kommen. In Abschnitt 14.4 werden zunächst Methoden vorgestellt, mit denen Multiagentensysteme entwickelt werden können, danach werfen wir einen kurzen Blick auf Werkzeuge und Anwendungen. Das Kapitel endet mit einer kurzen Diskussion von spannenden und aktuellen Themen, die nicht näher behandelt werden können.

14.1 Vom Agenten zum Multiagentensystem

14.1.1 Begriff und Charakteristika

Ausgangspunkt für jede Behandlung von Multiagentensystemen ist der Begriff „Agent“. In den Anfangsjahren herrschte noch terminologische Verwirrung. Jede Publikation begann mit einer entsprechenden Definition des dabei verwendeten Agentenbegriffs. Der Höhepunkt war [41], die verschiedene Definitionen miteinander verglichen haben. Auch in der Künstlichen Intelligenz selbst wurde mit [74] eine Agentensicht etabliert. Sie geben in ihrem bekannten Lehrbuch zur Künstlichen Intelligenz – das vom Agentenkonzept ausgehend organisiert ist – eine Definition des Begriffs „Agent“ als „An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.“ (p. 34). Diese sehr einfache Definition wird erweitert zu rationaler Agent etc. Eine andere wichtige alternative Sichtweise des Agentenbegriffs stammt aus der Principal-Agent Theorie der Wirtschafts- und Sozialwissenschaften [35]. Dabei wird „Agent“ als jemand gesehen, der im Auftrag seines Prinzipals handelt. Ein Mensch delegiert Aufgaben an „seinen“ Agenten. Diese Interpretation spielt heute noch bei Interface Agenten oder ähnlichem eine Rolle. Mittlerweile hat sich die Begriffsbildung praktisch konsolidiert: Die Definition, die beide Sichtweisen in sich vereinigt und sich durchgesetzt hat, ist folgende:

„An agent is a computer system that is situated in some environment and that is capable of independent (autonomous) action in this environment on behalf of its user or owner (figuring out what needs to be done to satisfy design objectives, rather than constantly being told)“ [90, p. 15]

Die wichtigsten Elemente dieser Definition sind

- **Situiertheit** in einer Umwelt beinhaltet, dass der Agent in einer Umwelt existiert und von dieser klar abgrenzbar ist. Das „In-einer-Umwelt-Sein“ hat als Konsequenz, dass ein Agent an einer bestimmten Position lokalisiert ist, von der aus er (lokal zugängliche) Informationen erhält und auch nur innerhalb eines bestimmten Aktionsradius Elemente seiner Umwelt manipulieren kann.
- **Unabhängigkeit/Autonomie** definiert M. Wooldridge klar als eigenständige Zielerfüllung. Das ist mehr als die (energetische) Unabhängigkeit eines einfachen Roboters, aber deutlich einfacher als die vollständige Autonomie, wie sie bei [74] beschrieben wird. Dort muss ein autonomer Agent sein Verhalten vollständig selbst bestimmen können, z.B. durch Lernen. [52] geben in ihrem einführenden Kapitel eine kurze Diskussion von verschiedenen Formen der Autonomie: von absoluter Autonomie, die man auch als Autismus bezeichnen könnte, zu Ausführungsautonomie, die nur fordert, dass der Agent seine Aktionen ohne Hilfe durchführen kann.

Diese grundlegende Definition wird erweitert zum „intelligenten Agenten“, der in der Lage ist flexibel zu handeln. Flexibilität bedeutet dabei, dass der Agent sein Verhalten an eine dynamische Umwelt anpassen und dabei immer seine Ziele im Auge behalten kann. Die reine Anpassungsfähigkeit wird auch als Reaktivität bezeichnet, das eigenständige Zielerfüllen als Proaktivität. Interaktionsfähigkeit mit anderen Agenten ist eine weitere zentrale Komponente eines intelligenten Agenten. Das bedeutet nicht unbedingt eine Beschränkung auf nachrichtenbasierten Austausch, Interaktion kann auch über die Umwelt geschehen: ein Agent verändert die Umwelt absichtlich oder unabsichtlich, andere Agenten nehmen diese Änderung wahr und adaptieren daraufhin ihr Verhalten.

Autonomie, Reaktivität, Zielgerichtetetheit und Interaktionsfähigkeiten sind die zentralen Eigenschaften, die einen intelligenten Agenten von einfacheren Einheiten unterscheiden. Ein grundsätzliches Problem ist dabei das Zusammenspiel zwischen Reaktivität und Zielgerichtetetheit: Ein Agent, der sein Verhalten permanent und sein aktuell verfolgtes Ziel opportunistisch ändert, wird es womöglich nicht schaffen, dieses Ziel zu erreichen. Ein Agent, der stur auf sein Ziel hinarbeitet und seine Pläne und Aktivitäten nicht den dynamischen Umweltbedingungen anpasst, wird ebenso scheitern. Im nächsten Abschnitt sollen die wichtigsten Agentenarchitekturen vorgestellt werden, die insbesondere eine Lösung für das Dilemma zwischen Reaktivität und Zielgerichtetetheit anbieten.

14.1.2 Wichtige Agentenarchitekturen

Das Grundprinzip aller Agentenarchitekturen ist gleich und wird in Abbildung 14.1 dargestellt. Der Agent bekommt Eingaben aus seiner Umwelt – über die Wahrnehmung oder Nachrichten – und benutzt diese Eingaben und den Inhalt seines internen Speichers, um die nächste Aktion zu bestimmen. Diese Aktion bestimmt dann, wie der Agent auf seine Umwelt einwirken will. Es liegt dann an der Umwelt, welche Effekte diese Aktion tatsächlich hat. Dies wird in [38] schön formalisiert.

Die Frage ist, wie der Agent im konkreten Fall seine nächste Aktion bestimmt. Eine Agentenarchitektur ist ein Muster, das beschreibt, wie interne Komponenten eines Agenten zusammenspielen, um aus Sensoreingaben im Endeffekt Verhalten zu generieren (für eine ausführlichere

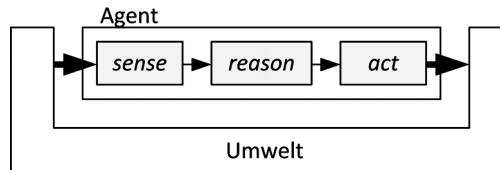


Abbildung 14.1: Der Wahrnehmen-Reasoning-Aktion-Zyklus eines Agenten.

Definition siehe [59]). Im Laufe der Jahre wurden sehr viele verschiedene Architekturen für Agenten vorgeschlagen. Die allgemein bekanntesten sind die Agentenarchitekturen – oder besser Architekturkategorien –, die in [74] vorgeschlagen wurden:

- Einfache Reflexagenten verknüpfen die Wahrnehmung des aktuellen Umweltzustandes direkt mit Aktionen, die in diesem Zustand ausgeführt werden sollen.
- Ein modellbasierter Reflexagent besitzt zudem einen Speicher, in dem Informationen über den Zustand der Umwelt gespeichert werden können. So muss der Agent nicht mehr in jedem Aktualisierungszyklus die vollständige Information in der Umwelt finden, um die geeignete Aktion auszuwählen.
- Ein zielbasierter Agent erweitert die Architektur um explizites Wissen über Ziele, für deren Erreichen Pläne erzeugt werden können. Situationen und Aktionen sind nicht mehr direkt verknüpft, sondern flexibel in einem oder mehreren Plänen kombiniert. Dies erlaubt komplexere Probleme zu lösen.
- Eine vierte Ebene an Komplexität stellen die nützlichkeitsbasierten Agenten dar. Ein Agent besitzt mehrere Ziel gleichzeitig, die er bewertet und verfolgt. Es gibt interessante Kombinationen von Architekturen: Ein Agent kann die Nützlichkeit eines jeden Zustands lernen. Auf der Basis dieses Wissens kann er im deterministischen Fall direkt bestimmen, welches die Aktion ist, die aus dem aktuellen Zustand zu dem nächsten mit der höchsten Nützlichkeit führt und dieses Wissen in Regeln organisiert. So wird aus einem nützlichkeitsbasierten Agenten ein einfacher Reflexagent mit oder ohne internen Zustand abhängig davon, wie viel Zugriff auf Zustandsinformation die Sensoren des Agenten bieten.

Diese allgemeinen Architekturkategorien kann man auch bei Multiagentensystemen finden. Reflexagenten wurden vor allem mit dem Ziel, eine schnelle Reaktion zu produzieren, verwendet; ein interner Zustand ist jedoch in einer nur lokal zugänglichen Umwelt unerlässlich, z.B. um Sensoreingaben zu puffern und zum Gesamtbild des aktuellen Umweltzustands zusammen zu fügen. Seit den 90igern wurden einige Architekturen vorgeschlagen. Regeln werden zum Beispiel aus Plänen kompiliert, indem alle Eventualitäten in einer baumartigen Struktur integriert wurden, z.B. in [76]. Andere Architekturen organisieren Aktivitäten, Aufgaben oder nur Aktionen in Graphstrukturen (z.B. [17, 31, 59, 62]). Diese sollen dann in der aktuellen Situation in Konkurrenz miteinander die nächst passende Aktion auszuwählen. Mittlerweile haben sich vor allem zwei Architekturformen herauskristallisiert: BDI-Architekturen und geschichtete Architekturen.

BDI (Belief Desire Intention) Architektur

BDI Architekturen gehören zu den wichtigsten Agentenarchitekturen. Die Idee stammt aus der Alltagspsychologie. [90, Seite 65ff] nennt es auch „praktisches Reasoning“. Die Ideen stammen

ursprünglich von M. E. Bratman, der sich als Philosoph mit dem menschlichen Denken beschäftigt hat: Um über Aktionen zu entscheiden, muss man sich zunächst bewusst machen, was man erreichen will und erst danach, wie man das Gewünschte erreichen kann. Das bedeutet, der Gesamtprozess ist bei Agenten in zwei Teile unterteilt: Zielfindung und Plangenerierung: Ein Agent generiert eine Menge von Zielen („Desire“ oder „Option“), die er verfolgen könnte, dann filtert er diese Ziele und legt sich auf das Verfolgen einiger weniger Ziele fest. Diese werden zu Intentionen. Ein Studierender beispielsweise kann für die Zeit nach dem Studium verschiedene Ziele haben: a) viel Geld verdienen, b) promovieren, oder c) auf Bali eine Bar aufmachen. Die Gesamtmenge an Zielen muss nicht konsistent sein. Jedoch, wenn sich der Agent auf eine Teilmenge festlegt, dann muss diese Teilmenge erreichbar sein, bzw. der Agent muss glauben, dass alle Intentionen in dieser Menge für ihn erreichbar sind. Das dritte Element der BDI-Architektur ist das interne Weltmodell des Agenten, die „Beliefs“². Damit ist das beschrieben, von dem der Agent denkt, dass es wahr ist: seine Annahmen, die Information in seinem internen Weltmodell, das er zu vorherigen Zeitpunkten aus seinen Wahrnehmungen oder Information von anderen Agenten aufgebaut hat. Die Welt kann sich aber geändert haben, Information kann fehlerhaft sein. Demzufolge müssen seine Beliefs nicht mit der tatsächlichen Situation übereinstimmen.

Auf der Basis dieser Überlegungen, wurde nicht nur eine BDI-Logik entwickelt [71], sondern die wohl einflussreichste Agentenarchitektur: Das „Procedural Reasoning System“ (PRS, [54]). Abbildung 14.2 zeigt den schematischen Aufbau von PRS.

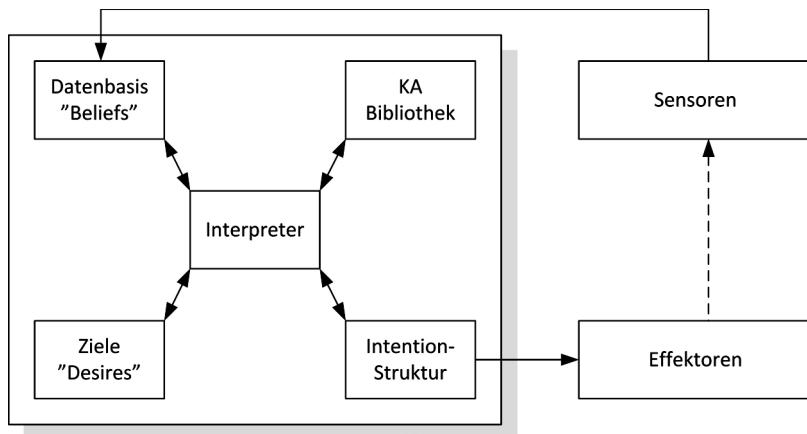


Abbildung 14.2: Schematischer Aufbau der PRS Architektur (nach [54]).

Intentionen werden dabei in Form sogenannter „Knowledge Areas (KA)“ repräsentiert. Das sind Planschablonen, mit denen in einer Graphstruktur Verhalten beschrieben wird. KAs können hierarchisch aufgebaut sein. Ausgewählte Intentionen werden auf einem Stack organisiert und abgearbeitet. Änderungen dieses Stacks werden durch neue Information oder neue Ziele getriggert. Der Stack kann auch über Meta-KAs manipuliert werden. PRS hat diverse Nachfolger-Architekturen, die zur Zeit bekannteste ist AgentSpeak(L) [70], auch wegen der Entwicklungsplattform JASON [15].

² „Beliefs“ sinnvoll in das Deutsche zu übersetzen, ist nicht ganz einfach. Die direkte Übersetzung wäre „Glaube“ oder „Überzeugungen“, was eine nicht ganz angemessene religiöse Konnotation hat.

Schichtenarchitekturen

Während bei den BDI-Architekturen das Balance-Halten zwischen Reaktivität und Zielgerichtetheit eine Frage des Reasoning-Mechanismus' und der Repräsentation des Verhaltens in den Planschablonen ist, wird in geschichteten Architekturen reaktives und zielorientiertes Verhalten explizit getrennt. Abbildung 14.3 zeigt die beiden grundsätzlichen Alternativen: horizontale und vertikale Schichtung. Als Vorbilder für diese etwas vereinfachten Darstellungen dienen die TouringMachines [39] für die horizontale Schichtung und InterRAP [61] für die vertikale Schichtung. Es gibt eine weitere Variante einer vertikalen Schichtung, die sogenannte Ein-Pass vertikale Schichtung. Im Gegensatz zu der hier dargestellten Zwei-Pass Schichtung, werden die Daten immer zur nächsten Schicht weitergereicht. In der Zwei-Pass Architektur kann die reaktive Schicht alleine bereits das Verhalten bestimmen.

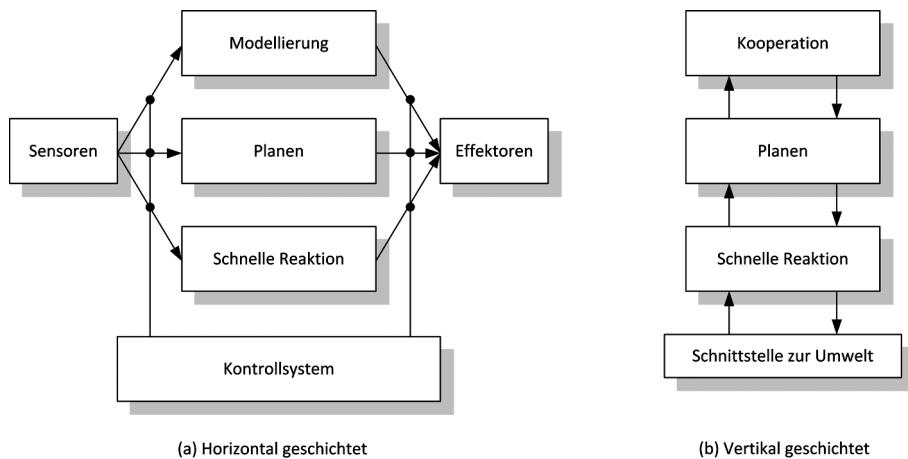


Abbildung 14.3: Schematischer Aufbau einer horizontal (a) und vertikal geschichteter (b) Agentenarchitektur.

Die verschiedenen Schichten haben klar getrennte Verantwortlichkeiten. Die unterste Schicht ist für die schnelle Reaktion in Notfällen verantwortlich, die Planer-Schicht für zielgerichtetes Verfahren. Eine dritte Schicht beschäftigt sich mit anderen Agenten. Der Unterschied liegt in der Zusammenschaltung der verschiedenen Schichten:

Bei der horizontal geschichteten Architektur werden alle Schichten gleichzeitig mit Sensorinformation versorgt und produzieren Vorschläge für die nächste Aktion. Welche von den Aktionen dann letztendlich zur Ausführung kommt, bestimmt eine Zensorkomponente. Diese mächtige, zentrale Komponente kann auch Sensorinformation für einzelne Schichten manipulieren, um z.B. zu verhindern, dass der Agent bei der Annäherung an ein interessantes, näher zu untersuchendes Objekt, kein Verhalten zur Kollisionsvermeidung aktiviert. Der kritische Punkt dabei ist klar diese Zensorkomponente, die sehr viel über den Aufbau und das Wissen der einzelnen Schichten kennen muss, um sinnvoll einzugreifen. Das Beispiel für eine horizontal geschichtete Architektur findet sich bei [39]. Auch die bekannte Subsumptionsarchitektur [17] kann als eine geschichtete Architektur gelten, allerdings ist die Funktionalität der Zensorkomponente implizit in die Verknüpfungen der Schichten eingebaut.

Prominenter sind vertikal geschichtete Architekturen. Ihr Prototyp war InterRAP [61]. Die unterste Schicht sorgt für eine schnelle Reaktion in Notfallsituationen. Erzeugt diese Schicht keine Aktion, delegiert sie die Aktionsgenerierung an die nächst-höhere Schicht, die aus dem aktuell verfolgten Plan die nächste Aktion liefert, bzw. für das aktuelle Ziel einen Plan generiert. Auf der obersten Schicht werden für die Arbeit in einem Team gemeinsame Ziele und die aktuelle Kooperationssituation verwaltet. Daraus werden die lokalen Ziele generiert, falls die Planungsebene kein aktuelles Ziel verfügbar hat. In InterRAP besitzt jede Schicht ihre eigene Wissensrepräsentation, z.B. Situation-Aktion-Regeln für schnelle Reaktionen, oder Skelettpläne für die Planungskomponente. Geschichtete Architekturen sind mittlerweile allgegenwärtig, z.B. auch in Simulationsanwendungen, in denen so taktisches und strategisches Verhalten getrennt wird. Ein Beispiel in der Verkehrssimulation findet man in [3].

14.1.3 Multiagentensystem

Es gibt im Bereich der agenten-basierten Systeme viele, die aus einem einzelnen Agenten bestehen. Dieser Agent ist oft ein persönlicher Assistent, mit dem ein Nutzer interagiert und an ihn Aufgaben delegiert, z.B. als Mail-Dämon. Der Übergang zu Multiagentensystemen ist dabei fließend: Ein Agent, der für seinen Benutzer bei einem Online-Auktionshaus in Konkurrenz zu vielen anderen Agenten Dinge ersteigert, macht dies als persönlicher Assistent in einem Multiagentensystem. Auch ein Mail-Agent kann sich mit anderen Mail-Agenten über aktuelle Spam-Signaturen austauschen. Ein Multiagentensystem ist ein System aus interagierenden Agenten. Diese Definition ist allerdings etwas zu kurz gedacht, da sie verschiedene Aspekte impliziert, aber nicht klar darstellt.

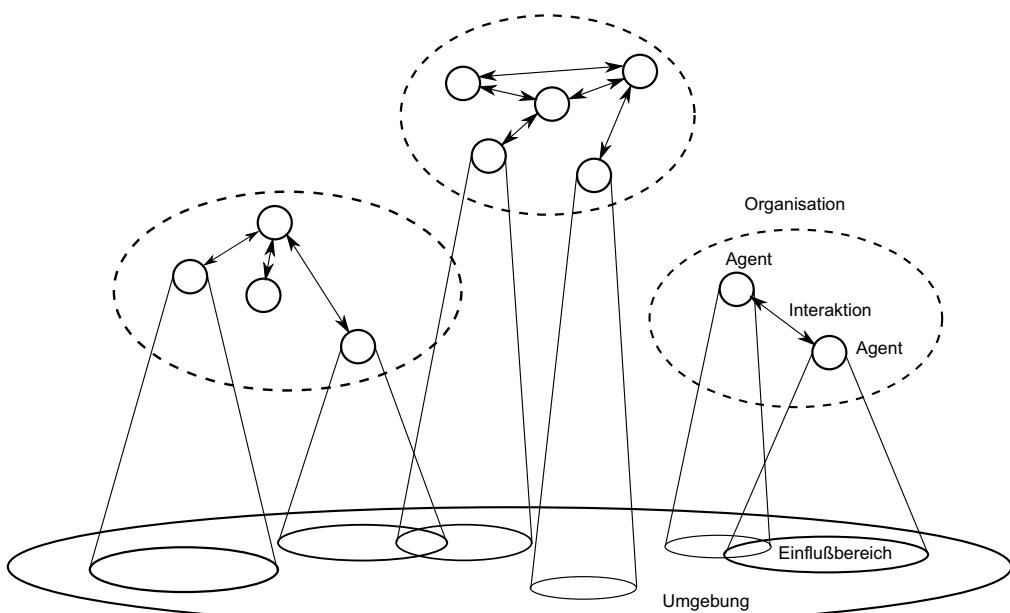


Abbildung 14.4: Multiagentensystem und Kontext (nach [90]).

Abbildung 14.4 zeigt eine abstrakte Darstellung eines Multiagentensystems in seinem Kontext: Jeder Agent kann in der allen Agenten gemeinsamen Umwelt nur ein beschränktes Areal wahrnehmen und hat ebenso nur einen beschränkten Einflussbereich. Wahrnehmung und Aktion sind also lokal. Die Agenten stehen miteinander in Beziehung. Sie interagieren, d.h. sie sind in einer dynamischen Relation zueinander, die durch eine Menge von aufeinander bezogenen Aktionen entsteht. Die Beziehungen der Agenten untereinander sind üblicherweise weder zufällig noch interagiert jeder Agent mit jedem anderen, sondern wer mit wem in Beziehung steht wird entweder durch die jeweiligen Aufgaben oder Positionen bestimmt und wird vom Entwickler des Systems festgelegt. Das Arrangement der Beziehungen zwischen Agenten, die zusammen ein System bilden, nennt man auch Organisation. Die meisten Autoren verlangen zudem als definierendes Element einer „Organisation“, dass die Organisation als Einheit ein Ziel hat, also das mehr oder weniger vordefinierte Muster der Agenteninteraktionen einem gemeinsamen Zweck dient [19]. Dies soll im Folgenden näher erläutert werden.

14.2 Interaktion, Kommunikation, Organisation

Interaktion zwischen Agenten bedeutet, dass die Aktionen eines Agenten das zukünftige Verhalten der anderen Agenten beeinflussen [36]. Es gibt direkte Interaktionen zwischen Agenten – z.B. durch direkte, nachrichtenbasierte Kommunikation – und indirekte Interaktion, bei der ein Agent die gemeinsame Umwelt verändert. Die Grundidee ist, dass die Manipulation oder Nachricht des einen Agenten das Verhalten der anderen Agenten beeinflusst.

Interaktionsfähigkeit ist eine der definierenden Eigenschaften des Agent-Seins. Zunächst soll nun nachrichtenbasierte Kommunikation darstellt werden. Nachrichten müssen in einen sinnvollen Zusammenhang gebracht werden. Das Senden einer Nachricht generiert Erwartungen, welche Nachrichten der Sender als Antwort empfangen könnte. Sinnvolle Konversationen werden mit Protokollen spezifiziert. Die nächste Frage ist dann, wer wann an welchen Protokollen teilnimmt und warum. Dies führt zu Organisation und Organisationsstrukturen. Zuletzt muss noch die Frage beantwortet werden, wie die Agenten letztendlich gut zusammenarbeiten können.

14.2.1 Kommunikation und Koordinationsinfrastruktur

Direkter Nachrichtenaustausch zwischen zwei oder mehreren Agenten ist die häufigste Form der Kommunikation. Weitere Formen sind die Multicast-Nachricht – es gibt mehrere Empfänger, deren Anzahl ist beschränkt, die Empfänger sind bekannt – oder die Broadcast-Nachricht, die wie im Radio an „alle“ geht, die zuhören. Das bedeutet, der Sender weiß nicht, wer und wie viele Agenten die Nachricht empfangen. Eine Alternative zum Nachrichtenaustausch ist die Verwendung eines Blackboards (siehe Seite 536).

Nachrichtenbasierte Interaktion

Kommunikation ist eine besondere Form der Interaktion, die sowohl das Empfangen von Nachrichten als Wahrnehmung und das Senden von Nachrichten als eine Aktion beinhaltet.

Die Basis für Kommunikation in Multiagentensystemen bildet die *Sprechakttheorie*. Die Basisidee dabei ist, dass Kommunikation eine Form von Aktion darstellt. Begründet wurde die

Sprechakttheorie von den Philosophen John Austin [2] und John Searle [77]. Ein Sprechakt entspricht einer Äußerung, die die Welt verändert. Ein oft verwendetes Beispiel ist die Aussage des Standesbeamten „Hiermit erkläre ich Sie zu Mann und Frau“ am Ende einer Trauung als einen Sprechakt, der Fakten schafft. Laut Sprechakttheorie besitzt jeder Sprechakt (etwas vereinfacht) drei Elemente: Die lokutive Komponente, die die Aussprache betrifft, die illokutive Komponente, die die Absicht des Sprechers darstellt und die perlokutive Komponente, die den Effekt auf den Empfänger beschreibt. Für die Entwicklung von Agentenkommunikationssprachen war insbesondere die Repräsentation der illokutiven Komponente wichtig. Sie wird explizit repräsentiert in Form von **Performativ** (Inhalt): **Request** („Hol' Kaffee“) oder **Inform** („Kaffee ist aus“). Das Performativ stellt explizit die Absicht des Senders dar.

Eine Agentenkommunikationssprache beschreibt die atomaren Bestandteile der Agentenkommunikation: die Nachrichten. Der sinnvolle Zusammenhang zwischen den einzelnen Nachrichten wird über Protokolle und ihre Spezifikation erreicht. Alle Nachrichtensprachen für Agentenkommunikation basieren auf der Idee, dass alle Information, die zum Verstehen des Inhalts einer Nachricht notwendig sind, mit der Nachricht verschickt werden soll. Eine solche Nachricht sieht typischerweise wie folgt aus:

```
(REQUEST
  :sender Professor
  :receiver Student
  :ontology Universität-Ontologie
  :language Deutsch
  :content "Kommen Sie pünktlich in die Vorlesung!"
)
```

REQUEST bestimmt die Intention des Senders der Nachricht, das Performativ. Im Beispiel ist es eine Anfrage oder Forderung (abhängig von der „Macht“ des Senders). Der eigentliche Inhalt steht im Feld **:content**. Um diesen Inhalt verstehen zu können, benötigt man zwei weitere Attribute: **:language**, das die Sprache angibt, in der der Inhalt geschrieben ist, und **:ontology**, mit der die Bedeutung der verwendeten Wörter spezifiziert wird. **:sender** und **:receiver** sind Informationen zum Sender und Empfänger der Nachricht. Es gibt weitere Felder, mit denen z.B. der Zusammenhang zwischen Nachrichten hergestellt werden kann, die im Beispiel nicht dargestellt sind.

KQML („Knowledge Query and Manipulation Language“) – entstanden aus der Knowledge Sharing Initiative [66] – war die erste Agentenkommunikationssprache [40], die eine weite Anwendung gefunden hat. Ihr Hauptproblem war, dass es weder eine Einigung auf einen minimalen Menge von Nachrichtenprimitive (Performative) gab, noch die Performative formal klar definiert waren. Je nach Anwendungsfall wurden neue Primitive hinzugefügt, so auch auch Performative für Infrastrukturaufgaben. Welchen Effekt ein Performativ haben sollte, konnte jedoch von Agentensystem zu Agentensystem unterschiedlich sein. Das Ziel der Unterstützung von Interoperabilität wurde dadurch nicht erreicht.

Aus diesem Grund wurde ACL („Agent Communication Language“) eingeführt und von FIPA³ standardisiert (siehe dazu www.fipa.org). Einen ausführlichen Vergleich zwischen KQML

³ FIPA steht etwas irreführend für „Foundation of Intelligent Physical Agents“ und zeichnet sich, 1996 gegründet, verantwortlich für Standardisierung bei Multiagentensystemen aus. Ziel ist, tatsächlich interoperable Multiagentensysteme zu schaffen.

und FIPA-ACL findet man in [57]. Die Anzahl der verfügbaren Nachrichtentypen ist beschränkt, es beinhaltet keine Infrastruktur-Performative, dafür aber Performative, mit denen Verpflichtungen („Commitments“) behandelt werden können. Somit können Verhandlungsprotokolle klarer realisiert werden. Ein weiterer Vorteil ist die formal definierte Semantik, die Interpretationsspielräume bei der Verwendung der Nachrichten verringert. Die Syntax von KQML wurde absichtlich wiederverwendet, um weiterhin die umfangreich vorhandene KQML-basierte Middleware benutzen zu können.

Indirekte Interaktion, Blackboard Systeme

Nachrichtenbasierte Interaktion ist meist direkt, d.h. von einem Agenten zum anderen. Viele Protokolle arbeiten mit „Multicast“ oder „Broadcast“ Nachrichten, um zum Beispiel mehreren Agenten gleichzeitig die Aufforderung zum Einreichen von Geboten zukommen zu lassen. Bei ersterem („Multicast“) ist die Anzahl der Empfänger begrenzt und die Empfänger sind dem Sender bekannt. Beim zweiten weiß der Sender nicht, wer und ob überhaupt jemand seine Nachricht empfängt. Eine andere Form der Interaktion geschieht indirekt über eine absichtliche oder unabsichtliche Manipulation der gemeinsamen Umwelt. Ein Agent ändert einen Aspekt der Umwelt, z.B. legt er eine Spur. Ein anderer Agent nimmt diese geänderte Umwelt wahr und passt sein Verhalten an, folgt der Spur. Diese Form der indirekten Interaktion wird auch Stigmergie genannt. Sie hat in so fern Ähnlichkeit mit Broadcast Nachrichten, als die tatsächlichen Empfänger unbekannt sind. Interaktionen mittels Stigmergie sind dazu noch zeitlich entkoppelt. Der Sender weiß weder, wer die Nachricht erhält, noch, wann die geänderte Umwelt wahrgenommen wird. Für ein robustes System muss die in der Umwelt abgelegte Nachricht wieder gelöscht werden. Ein prominentes Beispiel für stigmersive Interaktion ist die Massenrekrutierung bei Ameisen durch Pheromonspuren, die im Bereich der Ant Colony Optimization [30] in Multiagentensystemen zur Lösung von Routingproblemen nachgebildet werden.

Bei vielen emergenten Phänomenen (siehe [48]) liefert eine stigmersive Interaktion einen wichtigen Beitrag zur Entstehung. Kurz charakterisiert, geschieht Emergenz, wenn das Ganze mehr ist als die Summe seiner Teile. Durch Interaktionen zwischen Einheiten auf einer unteren, disaggregierten Ebene des Gesamtsystems werden nicht-lineare Rückkopplungen erzeugt, die auf einer höheren Ebene etwas „Neues“ erzeugen. Das Neue ist dadurch charakterisiert, dass es nicht mit den Mitteln beschreibbar ist, mit denen man das Verhalten der Entitäten auf der unteren Ebene beschreibt. Klassische Beispiele sind Neuronen im Gehirn, die ein Bewusstsein entstehen lassen, oder Stau, der sich als Ganzes in die entgegengesetzte Richtung zur Bewegungsrichtung seiner Bestandteile bewegt. Emergente Phänomene sind schwer zu analysieren, da sie erst bei der Simulation oder Ausführung der Verhaltensweisen auf der unteren Ebene entstehen. Multiagentensysteme eignen sich in besonderer Weise, Emergenz zu untersuchen. Wichtig ist andererseits, bei der Entwicklung von Multiagentensystemen unerwünschte emergente Phänomene auszuschließen.

Eine etablierte Form der stigmersiven Interaktion findet man bei Blackboardsystemen [21]. Diese wurden als Hilfsmittel beim gemeinsamen Problemlösen durch heterogene Agenten erfunden und basieren auf der Tafel-Metapher: Eine Gruppe von „Experten“ steht zusammen vor einer Tafel, ein Agent erweitert oder adaptiert einen Teil der gemeinsamen Lösung, die anderen nehmen diese Änderung direkt auf der Tafel wahr und adaptieren ihre Schlussfolgerungen und ändern ihrerseits. Die Tafel entspricht dabei dem gemeinsamen Speicherbereich der Agenten. Im Gegensatz zur Protokollen zur Aufgabenteilung unterstützen Blackboardsysteme das gemeinsame Bearbeiten von Ergebnissen, das sogenannte Result-Sharing. In den letzten Jahren wurde die

Implementierung von Blackboards auf der Basis von Tuplespaces vorgeschlagen [73]. Damit wurde Koordinationsinfrastruktur aus dem Bereich der Verteilten Systeme auf Multiagentensysteme angepasst.

14.2.2 Interaktionsprotokolle

Um eine sinnvolle Interaktion zu erreichen, reicht die Definition einer Nachrichtensprache nicht aus. Ein Agent und seine Interaktionspartner müssen einem gemeinsamen Muster von Nachrichten folgen, um zu einer sinnvollen Konversation zu kommen. Mit dem Schicken einer Nachricht erzeugt ein Agent Erwartungen, welche Antworten auf seine Nachricht möglich sind. Solche sinnvollen Nachrichtenfolgen werden in Protokollen spezifiziert. Nicht die Nachrichtentypen, die ein Agent verstehen kann machen seine Schnittstelle aus, sondern die Protokolle, an denen sich der Agent beteiligen kann. Auch Protokolle wurden von FIPA standardisiert. Zur Beschreibung von Interaktionsprotokollen wurden verschiedene Sprachen vorgeschlagen. [36, Kapitel 6] gibt einen schönen Überblick. Beispiele für Sprachen zur Modellierung von Protokollen basieren auf endlichen Automaten, wie z.B. COOL [4] oder auf Petri-Netzen, wie [23]. Meist wird jedoch AgentUML [63], eine Erweiterung der UML Sequenzdiagramme, verwendet. Auch FIPA benutzt diese Sprache zur Spezifizierung von standardisierten Protokollen. Ein leicht vereinfachtes Beispiel findet man in Abbildung 14.5. Die wichtigsten Erweiterungen betreffen zum einen die Behandlung von flexiblen Reaktionen auf eine Nachricht (wie im Bild dargestellt), zum anderen mögliche Verzweigungen der Lebenslinien. Zusätzliche Informationen, wie Performativ statt Methodennamen oder Rollen statt konkreten Objekten, betonen das höhere Abstraktionsniveau. Mit der Veröffentlichung von UML 2 wurde die weitere Entwicklung von AgentUML eingestellt [5].

Es gibt verschiedenste Protokolle zu unterschiedlichen Zwecken, von der einfachen Request-Agree Interaktion, bis zu komplexen, iterativen Verhandlungen. Ein wichtiger Schritt beim Entwickeln eines Multiagentensystems ist die Definition der verwendeten Protokolle (siehe dazu auch Abschnitt 14.4.1). Das Contract Net Protocol [82] ist das bekannteste Protokoll. Es dient zur Verteilung von Aufgaben auf Agenten und bildet einen Ausschreibungsmechanismus nach. Ein „Manager“ schickt eine Aufforderung zur Einreichung von Vorschlägen an eine Menge von möglichen Bearbeitern. Diese berechnen, wie schnell, mit welcher Qualität sie die Aufgabe erfüllen können, bzw. wollen. Auf dieser Basis schicken die möglichen Auftragnehmer Gebote an den Manager. Dieser evaluiert die Gebote und wählt das beste aus. Das Contract Net Protocol ist flexibel, robust und funktioniert, auch wenn mögliche Auftragnehmer ausfallen. Es kann hierarchisch angewendet werden. Natürlich gibt es auch Schwachpunkte zusätzlich zur zentralen Funktion des Managers: das Protokoll geht von einer vorgegebenen Unterteilung der Gesamtaufgabe aus. Wenn diese Unterteilung nicht zu den während der Verhandlung verfügbaren Agenten passt, dann ist das Protokoll hilflos. Dies gilt ebenso, wenn aus verschiedenen Gründen die besten Agenten nicht auf die Ausschreibung reagieren. Abbildung 14.5 zeigt eine Darstellung des Contract Net Protocols in einer etwas vereinfachten Version von AgentUML.

14.2.3 Organisation

Ein zentraler Aspekt – insbesondere wenn das Multiagentensystem als solches gezielt für einen bestimmten Zweck entwickelt wird – ist, wer mit wem zu welchem Zweck interagiert. Um die Entwicklung solcher kooperativer Multiagentensysteme zu verbessern, kann man eine explizite

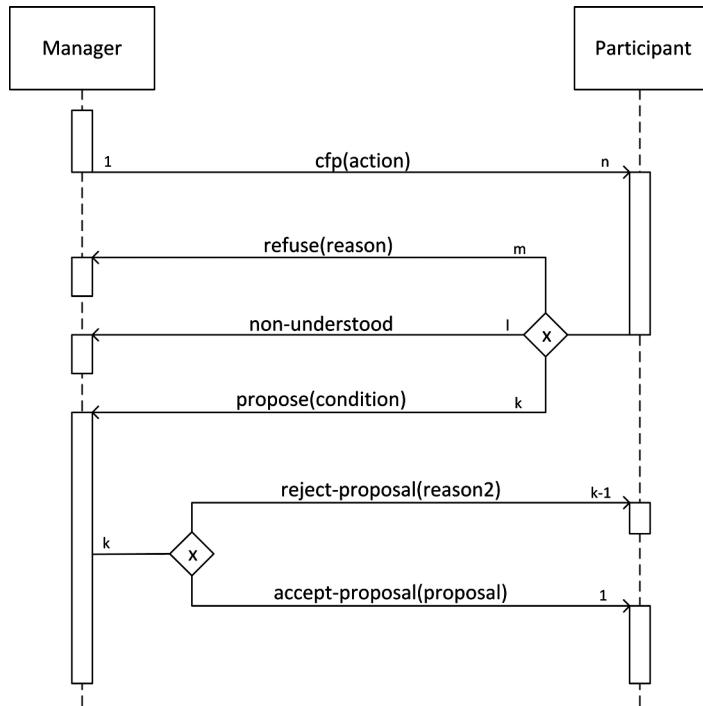


Abbildung 14.5: Spezifikation des Contract Net Protocols.

Sicht auf die Organisation von Multiagentensystemen verwenden. Ausgehend von der Idee, dass eine Organisation eine gemeinsames Ziel hat, kann ein Entwickler versuchen, die Menge aller Agenten in der Organisation zu strukturieren, Gruppen zu bilden, diesen Gruppen Teilziele zuzuordnen usw. Ein solches explizites Vorgehen ist typisch für eine Reihe von Methodologien im Bereich des Agentenorientierten Software Engineerings. [60] und [49] geben einen Überblick über die Organisationskonzepte, die auf verschiedenen Ebenen eines Multiagentensystems benutzt werden können. Die wichtigsten sind dabei folgende:

- Die *Rolle* als eine abstrakte Darstellung der Aktivitäten bzw. Services, die ein Agent im Rahmen einer Organisation bereitstellen muss. Die Rolle fokussiert also darauf, „was“ zu tun ist, ein Agent dagegen kümmert sich dann um das „wie“, wenn er die Rolle akzeptiert hat. Eine Rolle kann von mehreren Agenten gleichzeitig gespielt werden. Oft werden mit Rollen Verantwortlichkeiten und Pflichten assoziiert, ebenso wie Rechte und Ziele. [18] geben einen aktuellen Überblick über den Rollenbegriff in Multiagentensystemen, sowie über verschiedene Mechanismen, wie Rollen einem Agenten zugeordnet werden können.
- Abhängigkeiten und Protokolle verbinden Rollen in einer klar definierten und systematischen Art und Weise. Abhängigkeiten können dabei verschiedenste Formen annehmen – z.B. eine Informationsbeziehung, bei der ein Agent zu einem gewissen Zeitpunkt Informationen von einem anderen benötigt oder eine Autoritätsbeziehung, bei der ein Agent Anweisungen vom anderen akzeptiert. Protokolle realisieren die Abhängigkeiten oder lösen gegebenenfalls Konflikte auf.

- Rollen werden in *Gruppen* zusammengestellt. Gruppen sorgen für die Strukturierung der Interaktion, da meist ein Agent nur mit anderen Agenten aus der gleichen Gruppe interagieren darf. Interaktionen zwischen Gruppen können über Agenten geschehen, die Rollen in mehreren Gruppen spielen. Einer Gruppe kann dabei auch ein spezielles gemeinsames Ziel zugeordnet werden.

Eine Organisationsstruktur bestehend aus der Spezifikation von Gruppenstrukturen und Rollen, kann so als Muster oder Architektur für ein Multiagentensystem gesehen werden. Dies ist vergleichbar zu Agentenarchitekturen, die die Prozesse in einem Agenten steuern. Es wurden viele Spezifikationssprachen für Organisationen vorgeschlagen, die eine Entwicklung von organisationsbasierten Multiagentensystemen unterstützen sollen. Die bekanntesten sind das AGR-Modell [37], das eine recht pure Sicht auf Gruppen und Rollen bietet und sich auf einfache Konzepte beschränkt, und MOISE+ [51], das eine ausgefeilte Sprache bietet, unter anderem mit Rollenvererbung, verschiedenen Beziehungstypen oder hierarchischen Gruppenstrukturen. In einer zusätzlichen Sicht wird auch eine Zieldekomposition und Zuteilung zu Rollen unterstützt. Eine weitere Spezifikationssprache ist OperA [29]. Man findet auch verschiedene Ansätze zur Modellierung von Organisationen in Kombination mit Methoden zum Agentenorientierten Software Engineering. [85] zeigen, wie man eine dynamischere normative Sicht mit Agenten und Organisationen in einem flexibleren Modellierungsrahmen vereinigt. Informationen, wie mit Multiagentensysteme mit Organisationskonzepten programmiert werden können, findet man z.B. in [50].

14.2.4 Koordinierte Aktivitäten

Als letzter Schritt bei der Behandlung von Kommunikation und Koordination sollen nun Ansätze beschrieben werden, wie Agenten, nachdem sie wissen, mit wem sie zusammenarbeiten sollen, ihre Aktivitäten koordinieren können, um diese Zusammenarbeit zu realisieren (für eine Übersicht siehe [32]). Im Allgemeinen geht man davon aus, dass jeder Agent sein Teilziel kennt. Die Verteilung kann inhärent sein, z.B. auf der Basis der Rolle in einer Organisation, aufgrund von räumlicher Verteilung oder sie kann mittels z.B. eines Contract Net Protocols erfolgt sein. Auf der Basis dieser Aufgabenverteilung erzeugt jeder Agent einen Plan seiner zukünftigen Aktivitäten. Wie dies gemacht wird, ist abhängig von der verwendeten Agentenarchitektur. Der zentrale Aspekt ist die Koordination der Aktivitäten, um Abhängigkeiten zwischen den Plänen aufzulösen. Die Koordination kann vor dem Planen, während des Planens oder nach dem Planen geschehen. Bei der Koordination vor dem Planen versucht man, durch Organisationsstrukturen oder durch zusätzliche Constraints die Handlungsmöglichkeiten der Agenten so weit einzuschränken, dass Interferenzen gar nicht auftauchen, auch wenn jeder Agent unabhängig vom anderen plant. Solche Constraints werden auch als Soziale Gesetze („Social Laws“) oder Normen bezeichnet. Ein klassisches Beispiel ist die Regel, dass Fahrzeuge am rechten Rand der Straße fahren sollen. Dieses Rechtsfahrgesetz schränkt die Freiheit der Verkehrsteilnehmer ein; sie dürfen nicht mehr überall fahren. Es sorgt aber, sofern sich jeder dran hält, für eine konfliktfreie Wegeplanung und koordiniertes Fahren. Ein wichtiger Aspekt bei Normen insbesondere bei offenen Systemen (siehe nächster Abschnitt) ist die Durchsetzung dieser normativen Regeln. Einen generellen Rahmen und Formalisierung von „Coordination by Design“ – also der Koordination vor dem eigentlichen Planen findet man in [83].

Die beiden anderen Möglichkeiten, d.h. die Koordination nach und während des Planens, erfordern, dass Agenten in der Lage sind, so flexible Pläne zu entwickeln, dass sie für die Koor-

dination angepasst werden können (im Folgenden nach [32]). Bei der Koordination nach dem Planen erzeugt jeder Agent unabhängig von den anderen einen partiell geordneten Plan. Danach wird dieser Plan entweder an eine zentrale Instanz geschickt, die die Pläne der einzelnen Agenten aufeinander abstimmt, oder die Agenten erledigen diese Abstimmung durch massive Kommunikation, in dem sie sich z.B. schrittweise darauf einigen, wer wann welche Aktion ausführen darf. Die dritte Form des Multiagentenplanens ist, wenn das Planen und die Koordination miteinander verzahnt sind, wie z.B. im Partial Global Planing (PGP) [33] (oder seiner verallgemeinerten Version, dem Generalized Partial Global Planing (GPGP) [27]). Die Grundidee dabei ist, dass die Agenten gleichzeitig abstrahierte Pläne koordinieren, während sie diese abstrakten Pläne auf lokaler Ebene ausführen.

14.3 Von der Kooperation zum Wettbewerb

Bisher haben wir Systeme betrachtet, bei denen ein oder eine Gruppe von Entwicklern ein Gesamtsystem aus interagierenden Agenten entwickelt. Dabei kann angenommen werden, dass die Agenten, die gemeinsam ein bestimmtes Ziel erreichen sollen, dies wenn möglich auch tun wollen. Die Agenten kooperieren, weil sie so gemacht wurden. Eine andere Form der Multiagentensysteme sind die offenen System [47]. Dabei kann jeder Agent von einem anderen Entwickler entworfen worden sein und damit ein Ziel verfolgen, das nicht kompatibel mit den Zielen der anderen Agenten ist. Der Agent kann im Wettbewerb mit den anderen Agenten stehen. Ein Beispiel sind Auktionsplattformen im Internet: Benutzer (Agenten) können jederzeit hinzukommen, um Dinge zu verkaufen oder auf Dinge zu bieten. Bieten zwei Agenten auf das gleiche Gut, stehen sie im Wettbewerb um dieses Gut. Keiner weiß vom anderen, wie viel derjenige letztendlich bereit ist zu bieten, aber beide wollen das Gut so günstig wie möglich ersteigern. Solche *offenen* Multiagentensysteme haben andere Charakteristika als Multiagentensysteme, die mit einem Systemziel als Gesamtheit entworfen wurden. Man kann keine Annahmen über Kooperativität treffen, sondern muss davon ausgehen, dass die Agenten egoistisch und rational sind. Natürlich können obige Routinen, Protokolle oder Organisationsstrukturen auch in offenen Systemen verwendet werden, aber grundsätzlich müssen die egoistischen Agenten einen Vorteil im kooperativen Verhalten erkennen. Auch bei offenen Systemen verfolgen die Entwickler des Gesamtsystems bzw. der Interaktionsinfrastruktur ein Ziel. Das kann eine „gerechte“ Verteilung von Ressourcen oder eine bestimmte Funktionalität sein, die nur zusammen erreicht werden kann. Der Entwurf und die Analyse von Protokollen oder Mechanismen, die die Interaktion der Agenten so regeln, dass dieses Ziel erreicht werden soll, geschieht im sogenannten „Mechanism Design“ (siehe [56, 72, 75, 80]). Dabei versucht man, einen Mechanismus so zu entwerfen, dass er garantiert zum Ziel, d.h. einer Einigung kommt, und dass er von keinem Agenten so manipuliert werden kann, dass das Ergebnis nicht mehr für alle das beste ist. Außerdem soll es für den einzelnen Agenten besser sein, teilzunehmen als außen vor zu bleiben, d.h. individuell rational sein. Weitere Anforderungen sind unter anderem Verteiltheit (es gibt keinen Agenten, der eine zentrale Stelle einnimmt und so zum Flaschenhals wird), Effizienz und Einfachheit. Die Analyse solcher Mechanismen geschieht auf der Grundlage der Spieltheorie. Eine ausreichend tiefe Einführung in die Spieltheorie ist im Rahmen dieses Kapitels nicht möglich. [80] geben eine gut verständliche, aber sehr präzise Darstellung der Grundlagen der Spieltheorie soweit sie für Multiagentensysteme relevant ist. Im Folgenden soll daher nur kurz die grundlegende Idee der rationalen Agenten erläutert werden.

14.3.1 Idee des rationalen Agenten

Rationalität als Eigenschaft eines Agenten bedeutet zunächst (siehe oben), dass der Agent nichts macht, was seinen Zielen entgegen wirkt. Bei intelligenten Softwareagenten bedeutet dies, dass wenn ein Agent die Wahl hat, 10,0 Euro mit einer Lüge und 9,99 Euro für eine wahre Aussage zu verdienen (und sein Ziel ist möglichst viel Geld einzusammeln), dann wird dieser Agent lügen. Die Wahl der richtigen Aktion ist meist schwieriger, da das Ergebnis nicht nur von der Aktion eines Agenten abhängt, sondern auch von den Aktionen der anderen Agenten, die gleichzeitig ihre Aktion wählen und ausführen. Die für ihn beste Aktion bestimmt ein Agent, indem er alle Ergebnisse betrachtet, die bei allen jeweils möglichen Aktionskombinationen aller Agenten erzeugt werden. Diese möglichen Ergebnisse werden für zwei Agenten üblicherweise in einer Payoff-Matrix repräsentiert. Tabelle 14.1 ist ein Beispiel für eine Payoff-Matrix im Gefangenendilemma – dem bekanntesten Szenario der Spieltheorie.

Tabelle 14.1: Beispiel für eine Payoff-Matrix. Sie stellt vollständig die Resultate für beide Agenten für jede Aktionskombination dar. Die Werte entsprechen einem Gefangenendilemma.

		AgentB	
		C	D
		AgentA	
C	AgentA	2	1
	D	4	3
		1	3

Die Tabelle ist so zu lesen: Entscheidet sich der Agent A (der „Zeilenspieler“) für die Aktion „C“ (Kooperation) und der Agent B (der „Spaltenspieler“) für die Aktion „D“ (Verraten), erhält Agent A einen Punkt, während der Agent B 4 Punkte bekommt. Diese Payoff-Matrix ist ein Beispiel für das wohl bekannteste spieltheoretische Szenario: dem Gefangenendilemma. Es gibt zwei Aktionen – eine „Verraten“ (engl. defect, Aktion D) und „kooperieren“ (engl. cooperate, C). Das besondere an diesem Szenario ist der Wert des Vertrauens: Wenn beide kooperieren, ist das individuelle Ergebnis besser, als wenn beide nicht kooperieren (verraten). Die Versuchung zu verraten ist hoch, weil damit der höchste Gewinn erzielt werden kann. Ähnliche Situationen findet man häufig im realen Leben: Alle investieren in ein gemeinsames Projekt und kooperieren. Die Versuchung existiert, die Projektergebnisse auszunutzen ohne vorher in den gemeinsamen Topf investiert zu haben. Ein Agent der „C“ spielt, zahlt, ein Agent der „D“ spielt, nutzt das gemeinsame Gut ohne zu zahlen.

Die Spieltheorie zeigt, wie man solche Szenarien analysieren kann. Wichtige Mittel sind dabei: das Konzept der dominanten Strategie/Aktion, das Nash-Gleichgewicht und die Idee von Pareto-Optimalität. Alle drei – soweit vorhanden – sind Anhaltspunkte für die Stabilität des Spiels: Die dominante Strategie oder Aktion ist diejenige, die das beste Ergebnis für den Agenten liefert egal, welche Strategie der andere Agent auswählt. Im obigen Szenario ist dies die Aktion „D“. Wenn der Agent „C“ wählen würde, wäre das schlechteste mögliche Ergebnis für den Agenten gleich 1, wenn der „D“ wählt ist es 2. Das Nash-Gleichgewicht dagegen ist die Situation, in der die Strategie des einen Agenten die beste Antwort auf die des anderen Agenten ist und vice versa. Im Szenario ist das Nash-Gleichgewicht die Situation, in der beide Agenten „D“ wählen. Die beste Reaktion auf ein „D“ ist ein „D“ (ebenso ist „D“ die beste Reaktion auf ein „C“). Der pareto-efiziente Zustand dagegen, ist wenn beide „C“ wählen: Durch ein Abweichen von „C“ kann ein Agent nur dann etwas gewinnen, wenn ein anderer Agent verliert. Solche

Analysen ermöglichen es, die Entscheidungen von Agenten vorher zu sagen. Dies geschieht aber unter der Annahme, dass die Agenten vollständig rational sind und vollständiges Wissen über alle möglichen Ergebnisse als Reaktion der Aktionen aller Agenten kennen. Das Ergebnis widerspricht jedoch manchmal einer menschlichen Intuition von „Fairness“. [26] zeigen, dass „Fairness“ formal dargestellt werden kann und bei bestimmten Scenarien sehr nützlich sein kann.

Spieltheorie findet einen intensiven Einsatz in der theoretischen Ökonomie und anderen Sozialwissenschaften. Im Bereich der Multiagentensysteme bildet sie, wie oben erwähnt, die Grundlage für den „Mechanism Design“. Dieser Forschungsbereich beschäftigt sich mit der Entwicklung von Interaktionsmechanismen, also mit den Regeln des Spiels, die bestimmte Eigenschaften besitzen. Die hier dargestellte, einfache Form von spieltheoretischen Szenarien reicht dazu nicht aus. In [80] findet man eine Einführung in komplexere Formen, die tatsächlich für die Interaktionsanalyse verwendet werden.

Im Folgenden sollen die wichtigsten Interaktionsprotokolle für verschiedene, wichtige Szenarien vorgestellt werden. Es ist interessant zu sehen, wie viele dieser Mechanismen durch ein Vorbild aus menschlichen Gesellschaften inspiriert sind. In den 90igern wurde deswegen eine neue Forschungsrichtung „Sozionik“ gegründet, die Multiagentensystem- und Soziologische Forschung zusammenbringen sollte. Trotz großer Hoffnungen (und Fördermittel) hatten die Ergebnisse bisher noch kaum Einfluss auf den Stand der Forschung. Der Grund ist wohl auch, dass das hohe Abstraktionsniveau, auf dem Agenteninteraktionen behandelt werden, nur schlecht zu der Komplexität der von der Soziologie behandelten Realwelt passt.

14.3.2 Voting

Das Abstimmen – englisch „Voting“ – bezeichnet eine Gruppe von Mechanismen, bei denen die Agenten über die Auswahl einer Lösung aus einer Menge von möglichen Optionen abstimmen. Die gewählte Option ist dann für alle Agenten bindend. Die Agenten fällen also eine Gruppenentscheidung. Wie in menschlichen Gesellschaften kann über gemeinsame Aktivitäten oder einen „Anführer“ abgestimmt werden. Jeder Agent muss eine im einfachsten Fall vollständige Präferenzrelation besitzen, die für jedes Paar an Optionen angibt, welche Option der Agent im Vergleich zur anderen bevorzugt. Bei manchen Mechanismen muss jeder Agent nur öffentlich machen, welches seine bevorzugte Option ist, bei anderen Mechanismen muss der Agent seine gesamte Präferenzrelation offen legen. Es gibt eine Reihe von Abstimmungsmechanismen, jede mit spezifischen Vor- und Nachteilen. Eine tiefergehende Behandlung findet man in [75]:

- Pluralitätsprotokoll: Jeder Agent teilt als Eingabe dem Mechanismus mit, welche die von ihm bevorzugte Option ist. Der Mechanismus selektiert die Option mit den meisten Stimmen. Dieser Mechanismus hat eine sehr klare Entsprechung bei Wahlen in menschlichen Gesellschaften. Wie im richtigen Leben kann auch bei Agenten die Wahl in einem Unentschieden resultieren, wenn die Anzahl der Stimmen für mehrere, meist-gewählte Optionen gleich sind; dann wird eine zusätzliche Regel benötigt, um das Unentschieden auflösen zu können. Varianten des Pluralitätsprotokolls sind das kumulative Pluralitätsprotokoll, bei dem jeder Agent eine festgelegte Anzahl von Stimmen verteilen darf oder das Pluralitätsprotokoll mit Elimination, bei dem die Agenten wiederholt wählen: Nach jedem Wahlgang wird die Option mit den wenigsten Stimmen aus dem Pool der möglichen Optionen entfernt. Dies wird wiederholt, bis nur noch eine Option übrig ist.

- Bordaprotooll: Jeder Agent teilt dem Mechanismus als Eingabe seine vollständige Präferenzrelation mit. Für jeden Agenten ordnet der Mechanismus jeder Option eine Zahl zu, die der Position im persönlichen Ranking des Agenten entspricht. Diese Zahlen werden für alle Agenten aufsummiert. Die Option mit der höchsten Summe ist das Ergebnis des Mechanismus.
- Binärprotokoll: Die Agenten stimmen paarweise über die Optionen ab. Immer zwei Optionen werden miteinander verglichen. Die Option mit den meisten Stimmen wird mit der nächsten Option verglichen. Dies wird so lange wiederholt, bis nur noch eine Option übrig ist.

Folgendes Beispiel illustriert die verschiedenen Protokolle: Gegeben sind 3 Optionen $\{a, b, c\}$. Agent A besitzt folgende Präferenzrelation: $a > b > c$, Agent B $c > a > b$, ebenso wie Agent C, und Agent D $b > a > c$. Bei Verwendung eines Pluralitätsprotokolls teilt jeder Agent seine erste Wahl mit und die Option c gewinnt mit zwei Stimmen. Beim Borda-Protokoll wird für Agent A, der Option a eine 3 zugeordnet, 2 zu b und 1 zu c . Dies wird für alle Agenten entsprechend gemacht. Diese Zahlen werden summiert. Das Ergebnis ist, dass a den Borda-Score 9, b 7 und c 8 bekommt und somit a gewählt wird. Das Binärprotokoll vergleicht die Optionen paarweise: Nimmt man z.B. die Reihenfolge: $a-b-c$ kann folgendes passieren: In der ersten Runde stimmen z.B. die Agenten über a oder b ab: a ist der klare Gewinner mit 3:1 Stimmen. In der nächsten Runde $a-c$, gibt es ein Unentschieden, das durch zusätzliche Regeln aufgelöst werden muss. Man kann für jeden dieser Mechanismen ein Szenario konstruieren, bei dem Schwächen klar werden oder ein sogenanntes Voting Paradox auftritt. So kann sich beim Borda-Protokoll das Gesamtergebnis ändern, wenn irrelevante Optionen entfernt werden. Irrelevante Optionen sind solche, die kein Agent als bevorzugte Wahl hat. Das Binärprotokoll ist abhängig davon, in welcher Reihenfolge die Optionen miteinander verglichen werden. Auch hier kann man ein Szenario konstruieren, bei dem jede Reihenfolge zu einem anderen Gesamtergebnis führt. [80] schildert verschiedene Szenarien mit erstaunlichen Voting Paradoxien.

Alle diese Mechanismen können von Agenten manipuliert werden. In [80] findet sich ein schönes Beispiel, wie ein Pluralitätsprotokoll manipuliert werden kann: Ein Agent kennt die Präferenzrelationen aller anderen Agenten und weiß, dass seine bevorzugte Option keine Mehrheit finden wird. Die Manipulation besteht nun daraus, dass er nicht ehrlich seine bevorzugte Option angibt, sondern eine andere, und so den eigentlichen Gewinner der Abstimmung verhindert. Die Agenten haben einen Anreiz über mögliche Präferenzrelationen der anderen Agenten zu spekulieren, um die Abstimmung zu ihren Gunsten zu manipulieren. Derartiges Spekulieren und Manipulieren will man durch einen guten Mechanismus verhindern: Es gibt Abstimmungsverfahren – den Vickrey-Clarke-Groves Mechanismus [80] –, bei denen die dominante Strategie ist, ehrlich zu sein, d.h. immer die bevorzugte Option als solche zu benennen: Dabei gibt ein Agent nicht nur die Option an, sondern eine numerische Präferenz für jede Option. Der Mechanismus ermittelt dann nicht nur die Option mit der höchsten Bewertungssumme als den Gewinner der Abstimmung, sondern auch eine Art Steuer, die jeder Agent zahlen muss. Die individuelle Höhe der Steuer richtet sich danach, wie viel Einfluss der Agent auf das Ergebnis genommen hat. Die einzige sinnvolle Strategie für einen Agenten ist dabei, ehrlich zu sein. Würde er einen höheren Wert angeben, als die Option für ihn wirklich wert ist, würde er mehr Steuern zahlen als eigentlich notwendig. Würde er zu wenig angeben, um Steuern zu sparen, würde er riskieren, dass etwas anderes gewählt würde.

14.3.3 Auktionen

Ebenso wie Abstimmungsmechanismen orientieren sich auch Auktionen zunächst an Vorbildern aus der menschlichen Gesellschaft. Das Ziel ist bei Auktionen nicht, als Gruppe eine für alle bindende Option aus zu wählen, sondern, ein Gut dem Agenten zu geben, für den dieses am wichtigsten ist, bzw. der bereit ist am meisten dafür zu bezahlen. Im Endeffekt ist auch das Herzstück des oben beschriebenen Contract Net Protokolls eine Auktion um einen Auftrag. Die Grundaktionen sind dabei das Abgeben von Geboten, das Auswählen eines Gebots und das Festlegen/Zahlen eines Preises. Wie beim Abstimmen gibt es auch bei den Auktionen eine Vielzahl von Möglichkeiten. Diese unterscheiden sich auf einer technischen Ebene dadurch, ob die Gebote verdeckt oder offen abgegeben werden, ob alle Gebote gleichzeitig abgegeben werden oder iterativ, oder welcher Preis gezahlt werden muss. Für die Analyse einer Auktion ist noch dazu wichtig, wie das Auktionsgut bewertet wird, ob nur die persönliche Wertschätzung der Agenten relevant ist, oder ob die Auktion benutzt wird, um den Wert zu bestimmen. Ein Beispiel für ersteres findet man, z.B. bei der Auktion eines Werkzeugs oder einer Aufgabe; Beispiele für letzteres sind z.B. Kunstauktionen. Bekannte Formen von Auktionen sind folgende:

- Die Englische Auktion ist eine klassische Auktion, bei der ein Auktionator den Preis so lange erhöht, bis kein Bieter-Agent mehr ein höheres Gebot einreicht. Die dominante Strategie ist hierbei, das letzte Gebot um eine kleine Summe zu erhöhen. Das Protokoll ist nicht robust gegen eine verdeckte Koalition der Bieter. Der Auktionator kann den Mechanismus betrügen, indem er Preistreiber unter den Bieter versteckt.
- Einfache verdeckte Auktionen („sealed bid, first bid“) sind nicht iterativ. Jeder Bieter reicht ein verdecktes Gebot ein. Der Auktionator wählt das höchste Gebot aus, der Bieter zahlt den Preis seines Gebots. Der Bieter hat einen Anreiz, über die Gebote der anderen Agenten zu spekulieren, um den Preis zu drücken, den er zahlen wird. Dies ist umso interessanter, je höher sein Gebot als das der anderen sein würde. Wegen der verdeckten Gebote sind Koalitionen weniger attraktiv, da sie einfach gebrochen werden können, ohne dass der eigentlich Höchstbietende etwas dagegen machen kann.
- Bei einer Holländischen Auktion fängt der Auktionator mit einem hohen Gebot an und verringert den Preis schrittweise, bis der erste Agent den aktuellen Preis akzeptiert. In der Analyse ist diese Auktion äquivalent zur einfachen verdeckten Auktion.

Die wichtigste Form der Auktion in Multiagentensystemen ist allerdings keine der obigen drei, sondern die Vickrey-Auktion („sealed bid, second price“). Dabei geben alle Agenten ein verdecktes Gebot ab. Der Agent mit dem höchsten Gebot erhält das Gut, muss aber nur den Preis des zweithöchsten Gebots zahlen. Dadurch, dass die Höhe des zu zahlenden Preises vom gewinnenden Gut entkoppelt ist, ist die dominante Strategie für einen Agenten, seine wahre Wertschätzung des Gutes weiterzugeben. Bei der Vickrey-Auktion hat ein Agent keinerlei Motivation beim Gebot zu sparen: Ist sein Gebot niedriger als das, was er eigentlich zahlen würde, läuft er Gefahr, dass ein anderer Agent mehr bietet und das Gut erhält. Würde der Agent mehr bieten, als er eigentlich zahlen würde, könnte es sein, dass sein Gebot so knapp unterboten wird, dass der Agent mehr zahlt, als er eigentlich wollte.

Vickrey Auktionen findet man eigentlich in realen Gesellschaften nicht. Dies liegt vor allem daran, dass die Bieter dem Auktionator vertrauen müssen. Der gewinnende Bieter kennt die Höhe des zweithöchsten Gebots nicht. Der Auktionator könnte irgendeinen Preis knapp unter dem

höchstbietenden nennen. Interessant ist allerdings das Auktionsprotokoll von Ebay: Betrachtet man den Mechanismus im Gesamten mit automatischen Bietautoagenten, dann kann man ihn als Vickrey Auktion interpretieren, obwohl an sich eine klassische Englische Auktion verwendet wird: Der Bieter reicht den Wert, den er zu zahlen bereit ist, an den Bietautoagenten weiter und zahlt beim Gewinn der Auktion einen Preis, der minimal höher ist als der zweithöchste Preis. Der Bietautoagent nimmt an der eigentlichen Englischen Auktion teil, die stoppt, wenn kein anderer das letzte Gebot überbietet.

Wirklich kompliziert werden Auktionen, wenn der Wert der versteigerten Güter davon abhängt, was der Agent zuvor bereits ersteigert hat. In solchen Situationen macht es keinen Sinn, die Güter separat voneinander zu betrachten. Idealerweise schnürt der Auktionator ein Paket. Wenn das aus irgendwelchen Gründen nicht geht, dann sind die Agenten gezwungen, über ihre Gebote zu lügen, um zu einer effizienten Verteilung der Güter zu kommen. Ein wichtiger Ansatzpunkt ist dabei die Vorausschau: Z.B. werden zwei Güter G1 und G2 in dieser Reihenfolge versteigert. Macht es nun für einen Agent Sinn, dass er beide Güter hat, dann kann er seine Gebote durch Vorausschau modifizieren, um sicher zu stellen, dass er wirklich beide ersteigert: Bei der Berechnung des Gebots für G1 bezieht der Agent mit ein, welchen Vorteil er bei der Auktion um G2 dadurch haben wird, dass er dann bereits G1 hat. Dieser Vorteil von G2 wird auf das Gebot für G1 zurückgerechnet. Der Agent macht eine falsche Angabe bei G1, das Gesamtergebnis wird aber besser.

14.3.4 Verhandlungen

Ein weiteres Interaktionsprotokoll ist das „Bargaining“, das Handeln. Die Aufgabe ist hierbei, eine Ressource zu teilen. Beim strategischen Bargaining machen die Agenten abwechselnd oder gleichzeitig Vorschläge, die vom jeweils anderen Agenten akzeptiert oder abgelehnt werden. Kommt keine Einigung zustande, dann tritt die sogenannte „Fallback“-Lösung ein. Eine interessante Anwendung des Handelns ist die Verteilung von Aufgaben. [72] formalisieren dies als aufgaben-orientierte Domäne für Verhandlungen. Dabei besitzt jeder Agent eine initiale Zuordnung von Aufgaben, die er erfüllen muss. Über Verhandlungen mit anderen Agenten wird versucht, die Verteilung von Aufgaben zu verbessern, indem die einzelnen Agenten versuchen, Aufgaben abzugeben oder zu tauschen um die individuellen Kosten zu verringern. [72] schlagen ebenfalls ein Verhandlungsprotokoll vor, das monotone Zugeständnisprotokoll („monotonic concession protocol“): Die Menge möglicher Abmachungen („Deals“) ist dabei die Menge von Aufgabenverteilungen zwischen den Agenten, die für die Agenten jeweils individuell rational und pareto-effizient sind. Individuell rational bedeutet hier, dass die Kosten für die im Deal zugeordneten Aufgaben geringer als die ursprünglich zugeordneten Aufgaben sind. Pareto-effizient bedeutet, dass keine weitere Abgabe oder Tausch möglich ist, bei der ein Agent besser und ein anderer nicht schlechter gestellt wird. Das monotone Zugeständnisprotokoll geht nun wie folgt vor: Die Agenten beginnen damit, den für sie jeweils besten Deal aus der Menge der zulässigen vorzuschlagen. In der nächsten Runde ändert derjenige Agent seinen Vorschlag, der beim Rückfall auf die ursprüngliche Verteilung mehr verlieren würde. Im nächsten Vorschlag macht dieser Agent ein Zugeständnis für den anderen Agenten. Dies wird wiederholt, bis ein Agent den Vorschlag des anderen besser oder zumindest gleich gut wie den eigenen findet. Falls kein Zugeständnis möglich ist, dann wird die ursprüngliche Verteilung von Aufgaben realisiert. Eine kurze, aber gut verständliche Erläuterung findet man in [90].

14.3.5 Bildung von Koalitionen

Die Bildung von Koalitionen ist ein Problembereich, in dem sich egoistische Agenten zu einer Gruppe zusammen schließen, um ein Problem gemeinsam zu lösen. Eine Liste von Beispielen findet man in [80], z.B. eine Menge von Gemeinden, die gemeinsam einen Flughafen bauen könnten, der größer und effizienter ist, als wenn jede Gemeinde den eigenen Flughafen bauen würde. Wer beteiligt sich nun an dem gemeinsamen Flughafen, wer baut alleine bzw. wer baut mit wem? Eine vollständige Einteilung der Agenten in Gruppen nennt man auch Koalitionsstruktur. Die „Grand Coalition“ – die große Koalition – ist die Koalitionsstruktur, bei der alle Agenten in einer Koalition zusammenarbeiten.

Im Prinzip gibt es drei Phasen bei der Lösung des Koalitionsproblems (siehe auch [75]): 1) das Generieren der Gruppenstruktur 2) das tatsächliche Bilden eines Teams aus den Mitgliedern der Koalition, mit Verteilung der Aufgaben und deren Lösung und 3) das Verteilen des Gewinns, den man in der Gruppe gemacht hat. Das Problem ist dabei, dass der Gewinn, den ein Agent durch die Problemlösung in einer Gruppe machen wird, notwendig ist, um zu entscheiden, ob der Agent an der Koalition teilnimmt oder nicht. Es kann also sehr aufwändig sein, eine mögliche Koalitionsstruktur zu evaluieren. Für die Verteilung des Gewinns gibt es verschiedene Ansatzpunkte und Bedingungen. Insgesamt muss der Gewinn so verteilt werden, dass keine Subgruppe eine Motivation hat, aus der Koalition auszubrechen. Der Shapley-Wert⁴ bietet eine „faire“ Verteilung des Koalitionsgewinns: Jeder Agent erhält die Summe, die seinem Grenznutzen für die Gemeinschaft entspricht. Je mehr der Agent beiträgt, umso mehr erhält er. Wichtige Bedingungen sind, dass Agenten, die austauschbar sind und gleiches beitragen, auch gleich viel aus dem Koalitionsgewinn erhalten oder dass sogenannte „Dummy“ Agenten, deren Beitrag den Gesamtnutzen der Koalition nicht steigert, genau so viel erhalten, wie sie erhalten würden, wenn sie alleine arbeiten würden. Eine dritte Bedingung betrifft die Additivität der Bewertungsfunktion. Der Shapley-Wert für einen Agenten berechnet sich dann aus dem Gewinn der Koalition mit und ohne den Agenten und weitere Faktoren, um unterschiedliche Reihenfolgen des Hinzunehmens der Agenten in die Koalition abzudecken.

14.4 Entwicklung und Praxis

14.4.1 Agentenorientiertes Software Engineering

Eine zentrale Frage ist die, wie ein Multiagentensystem entwickelt werden kann. Dabei müssen im Endeffekt zwei Probleme gelöst werden: a) das Systemdesign und b) das Agentendesign. Bei ersterem muss das Gesamtsystem aus interagierenden Agenten so entwickelt werden, dass es bestimmte Anforderungen erfüllt. Das bedeutet, dass vom Multiagentensystem als Ganzes eine bestimmte Funktionalität in einer bestimmten Qualität bereitgestellt werden muss. Dabei müssen die Agenten entsprechend „organisiert“ werden. Das zweite Problem betrifft die untere Ebene, die der Agenten: Wie muss ein einzelner Agent entwickelt werden, welche Funktionalität und Eigenschaften muss ein Agent durch sein Verhalten bereitstellen, um das Ziel des Gesamtsystems zu erreichen. Methoden und Sprachen für eine systematische Lösung dieser Probleme zu entwickeln, ist das Ziel des Agentenorientierten Software Engineering [10]. [8] und [46] sind

⁴ Benannt nach Lloyd Shapley, der 2012 für diese Arbeiten den Nobel-Preis für Wirtschaftswissenschaften bekommen hat

Sammlungen mit Beschreibungen unterschiedlicher Methoden, die für die systematische Entwicklung von Multiagentensystemen vorgeschlagen wurden. Beide Kompendien enthalten auch Vergleiche zwischen den Methoden. Dies zeigt die Vielzahl vorhandener Ansätze. Bekannte, mehr oder weniger willkürlich ausgewählte Methodologien sind dabei im Folgenden ausgeführt. Weitere sind Prometheus [64] oder O-MaSE [28]. 2012 wurde bereits der 13te Internationale Workshop zum Agentenorientierten Software Engineering durchgeführt.

- GAIA [91] kann als die erste Methodologie zur Entwicklung von Multiagentensystemen betrachtet werden. Ausgehend vom Systemziel wird eine Organisation mit konzeptioneller Beschreibung von Rollen und Interaktionsprotokollen erstellt. Diese werden in weiteren Phasen konkretisiert. Agenten werden hinzugefügt, um diese Rollen ausfüllen zu können. Der Anwendungsbereich ist klar definiert als Systeme mit wenigen heterogenen und kooperierenden Agenten.
- TROPOS [16] legt einen besonderen Fokus auf das frühe Anforderungsmanagement als eine von fünf Phasen der Methodologie. Dort werden Akteure und ihre Abhängigkeiten untersucht und daraus funktionale und nicht-funktionale Anforderungen abgeleitet. Weitere Phasen sind ein verfeinertes Anforderungsmanagement, eine Designphase für die Gesamtsystemarchitektur und eine detaillierte Designphase für die Agenten. Eine fünfte und letzte Phase betrifft die Implementierung. Die wichtigsten Konzepte der dabei verwendeten Modellierungssprache sind unter anderem ein „Akteur“, der eine Entität mit strategischen Zielen darstellt, die Ziele selbst und die dabei auftretenden Abhängigkeiten.
- MAS-CommonKADS [53] hat seine Ursprünge im Wissensmanagement. Die Methodologie besteht aus fünf Phasen, von einer Konzeptionalisierungsphase als erstem Schritt zur Identifikation der funktionalen Anforderungen des Systems, über Analyse und Design zur Implementierung, Testen und zum tatsächlichen Einsatz. Verschiedene Aspekte des zu entwickelnden Multiagentensystems werden in einer Vielzahl von Modellen behandelt: Agentenmodell für die Eigenschaften und Aufbau der beteiligten Agenten; Taskmodell, das die Ziele und Methoden der Agenten strukturiert; das Expertisemodell für das notwendige Wissen; das Organisationsmodell für die Agenten-Gesellschaft; das Koordinationsmodell für die Interaktionen der Agenten untereinander; das Kommunikationsmodell für die Schnittstelle zum menschlichen Benutzer und zuletzt das Designmodell zur Konkretisierung des Netzwerkes, der Agenten und der verwendeten Plattform.
- MESSAGE [44] bieten einen Entwicklungsprozess auf der Basis des *Rational Unified Process*(RUP). Das Hauptaugenmerk liegt dabei auf der Erweiterung von UML durch Konzepte, die für ein Agentensystem relevant sind, wie Agent, Organisation, Rolle und Ziel. Es gibt zwei Typen von Aktivitäten: Aufgaben und Interaktionen/Interaktionsprotokolle; für jedes dieser Konzepte werden graphische Notationen in UML integriert. In einer Analysephase werden Ziele identifiziert, zerlegt und Rollen zugeordnet, Interaktionen werden beschrieben, die für das Erfüllen der Ziele notwendig sind. Abstrakte Modelle werden schrittweise verfeinert und in der Designphase zu Software-Elementen weiter verfeinert. INGENIAS [67] kann als Erweiterung von MESSAGE/UML gesehen werden. Es wird eine Umwelt-Sicht hinzugefügt und die Agentenmodelle stärker auf eine BDI-Architektur orientiert.
- Der PASSI [22] Prozess besteht aus fünf Elementen, die iterativ verfeinert werden: Systemanforderungsmodell, Agentengesellschaftsmodell, Agentenspezifikation, Implementierung und Verwendung. Bei jedem Schritt kommen entsprechende Beschreibungen in Standard UML zum Einsatz. Die Produktion von Programmcode wird unterstützt durch wiederverwendbare

Muster und Codegenerierung auf der Basis der standardisierten FIPA Multiagentensystem-Architektur.

- Adelfe [11] nimmt eine Sonderstellung im AOSE Bereich ein. Zum einen ist diese Methode deutlicher durch einer Bottom-up-Vorgehensweise geprägt als die anderen Ansätze, zum anderen zielt sie auf adaptive, lernfähige Agenten in offenen Multiagentensystemen ab. Basisprinzip ist das Konzept eines „Adaptiven Multiagentensystems“, das ein „Living Design“ ermöglicht: Agenten versuchen permanent kooperativ zu sein, was durch nicht-kooperative Situationen verhindert wird. Ein Beispiel ist eine Situation, in der ein Agent Informationen schickt, die der Empfänger nicht verstehen kann. Durch Adaptionen auf verschiedenen Ebenen – vom Parameter-Tuning zur Reorganisation –, lernen die Agenten solche Nicht-kooperativen Situationen zu vermeiden.

Aktuelle Entwicklungen gehen dahin, diese mehr oder weniger verschiedenen Ansätze miteinander zu kombinieren. Dazu werden die unterschiedlichen Methodologien in sogenannte Methodenfragmente zerlegt. Ein Fragment entspricht dabei einem Schritt in einem übergeordneten Prozess, mit dem z.B. eine bestimmte Sicht auf das Gesamtsystem spezifiziert wird oder mit dem Anforderungen auf die Agentenebene abgebildet werden. Diese Fragmente sollen mittels geeigneter Metamodelle verknüpft werden [9]. Sogar eine Selbstorganisation von Fragmenten ist angedacht [13]. Auf diese Weise können vollständige Methoden aus den Bausteinen zusammengesetzt werden, die für eine spezielle Anwendung am besten geeignet sind. Weitere, neuere Entwicklungen betreffen das modellbasierte Entwickeln von Multiagentensystemen [45]. Weitere Metamodelle sind FAML [12] oder [24]. Diese Metamodelle sollen als explizite Verallgemeinerung der verwendeten Abstraktionen die Grundlage für Schnittstellen zwischen Methodenfragmenten oder Softwaremodellen bilden.

14.4.2 Werkzeuge und Wettbewerbe

Seit 2002 gibt es eine eigene Serie von jährlichen Workshops, die sich ausschließlich mit Werkzeugen zur Programmierung von Agenten und Multiagentensystemen beschäftigt. Dies zeigt, dass es mittlerweile eine große Vielfalt von Werkzeugen und speziellen Programmiersprachen gibt. Es gibt viele Projekte, die die bekannten objekt-orientierte Programmiersprachen, wie z.B. JAVA oder C++ benutzen. Besonders interessant für Multiagentensystem-Projekte ist die funktionale Programmiersprache Erlang für verteilte Prozesse (www.erlang.org). Agent0 [79] gilt als erste Programmiersprache speziell für Agenten. Es gab eine Lisp-basierte Implementierung, von der heute keine funktionierende Version mehr gefunden werden kann. Das besondere an Agent0 ist, dass es eine logikbasierte Programmiersprache war, deren Programme auf sogenannten Commitment-Regeln beruhen. Das sind Regeln, mit denen der Agent sich selbst oder anderen das Ausführen einer bestimmten Aktion zu einem bestimmten Zeitpunkt verspricht. Eine ausgefeilte Programmiersprache für BDI-Agenten ist JASON [15], ein Interpreter für AgentSpeak(L) Programme. Einen Überblick über Programmiersprachen für Multiagentensysteme findet man unter anderem bei [14].

Es wurden mittlerweile auch viele Spezialwerkzeuge entwickelt. Einige der oben genannten Methodologien wurden mit Werkzeugen zur Spezifikation, Code Generierung oder für Simulation und Testen ausgestattet. Darüber hinaus gibt es viele Plattformen, die dem FIPA-Architekturmodell folgen und auch standardisierte Services für Agentenkommunikation bereitstellen. Die wohl bekannteste ist JADE (<http://jade.tilab.com/>, [7]). Das einfache

Agentenmodell wurde von L. Braubach und A. Pokahr [68] mit JADEX zu einer BDI Architektur erweitert.

Multiagentensimulation ist eine besondere Form der Anwendung, die mehr und mehr Beachtung findet. Laut der AgentLink III Roadmap [58] ist Simulation sogar das am besten etablierte Anwendungsgebiet von Multiagentensystemen. Ausgehend von Swarm (www.swarm.org) – das in den frühen 90igern für Simulationen von komplexen Systemen im Artificial Life Umfeld entwickelt wurde – haben sich eine Reihe von Simulationsplattformen etabliert. Die heute wichtigsten sind Repast (repast.sourceforge.net/) und NetLogo (ccl.northwestern.edu/netlogo/). Beim Einsatz im industriellen Umfeld haben Multiagentensysteme nach [69] den großen Vorteil, dass Agenten aus simulierten Umgebungen ohne großen Aufwand in die Realwelt überführt werden können. Dazu ist allerdings eine Simulationsplattform notwendig, die die Umwelt in ausreichendem Detaillierungsgrad repräsentieren kann, wie z.B. ihre AgentFly Plattform [81].

Information über Werkzeuge findet man insbesondere im Zusammenhang mit Wettbewerben, die dazu dienen die Leistungsfähigkeit von Werkzeugen, bzw. den damit erzeugten Artefakten zu vergleichen. Was 1997 mit dem RoboCup (www.robocup.org) und kleinen und großen fußballspielenden Robotern begann, hat in der Forschungslandschaft der Multiagentensysteme weite Kreise gezogen. Der RoboCup ist mittlerweile eine Sammlung von Wettbewerben, die nicht nur die Forschung im Bereich der komplexen, kooperierenden Roboter über Jahre angetrieben hat, sondern auch im Bereich des Mensch-Maschine Interaktion und Kooperation. Weitere Wettbewerbe folgten: Die „Trading Agent Competition“ (<http://www.sics.se/tac/>) für Agenten wurde 1999 initiiert und seit 2002 regelmäßig durchgeführt. Zu Beginn war es nur ein komplexes Auktionsszenario mit multiplen, von einander abhängigen Auktionen, um ein attraktives Reisepaket für einen Menge von Kunden zu schnüren. 2003 ist ein Supply Chain Scenario hinzugekommen. Mittlerweile gibt es weitere Szenarien mit Agenten, die möglichst gut handeln sollen. Der „Multiagent Programming Contest“ (multiagentcontest.org/) konzentriert sich seit 2005 auf interessante Herausforderungen zur Agentensteuerung und Koordination in abstrakten Benchmark-Szenarien. Ein wichtiger Aspekt ist dabei, dass nicht derjenige Teilnehmer mit der ausgefeiltesten Hardware gewinnt, sondern derjenige mit den besten Konzepten.

14.4.3 (Zu) kurzer Blick auf die Anwendungen

Schon vor zehn Jahren war es fast ein Ding der Unmöglichkeit, einen umfassenden Überblick über erfolgreiche Anwendungen von Multiagentensystemen zu geben. Einen recht kondensierten Überblick aus dieser Zeit findet man in [55].

Pěchouček und V. Mařík geben einen Überblick zur industriellen Anwendung von Multiagentensystemen und vier Fallstudien im Bereich der verteilten Kontrolle und Diagnose in technischen Systemen, der Produktionsplanung, der Luftverkehrskontrolle und Materialflusskontrolle. Passende Anwendungsgebiete für Multiagentensysteme sind nach [69]:

- Szenarien, in denen Daten und Wissen für die Problemlösung nicht an einem zentralen Ort vorhanden sind. Gründe dafür können in der geographischen Verteiltheit oder in Beschränkungen im Datenaustausch liegen, weil die Agenten in einer Konkurrenzsituation stehen.

- Szenarien, die besondere Robustheit und schnelle Reaktionen in einer verteilten Umgebung benötigen, z.B. in zeitkritischer Fertigung. In solchen Szenarien kann lokale Kontrolle oder lokales Umplanen sehr viel zur Robustheit des Gesamtsystems beitragen.
- Simulation und Modellierungszenarien, bei denen eine einfache Übertragbarkeit in die Realwelt gewünscht wird.
- Offene Systeme, die eine Integration und Interoperabilität zwischen Systemen erfordern, die nicht a priori bekannt sind, sondern erst während der Laufzeit hinzukommen.
- Komplexe Systeme generell, bei denen eine Aufteilung in Teilprobleme Sinn macht, die dann verteilt auf verschiedenen Agenten, mit Verhandlungen und gegebenenfalls parallel bearbeitet werden.
- Autonomie-orientierte Szenarien, bei denen ein Nutzer einen substantiellen Teil der Entscheidung an einen Agenten delegiert.

Diese Eigenschaften treffen auf viele Systeme in unterschiedlichsten Domänen zu. Sie reichen vom Taxi-Management [43] zu neuen Systemkonzepten zur lokalen Energieversorgung (z.B. [88]), von innovativen Ampelsteuerungen (siehe [6]) zum elektronischen Handeln (z.B. [25]), vom Supply Chain Management ([20]) zur Produktionssteuerung ([65] als Pionier), von virtuellen Gegnern oder Helfern in Computerspielen [84] zu Sicherheit in Flughäfen [78], von selbstorganisierenden Sensornetzen (für eine Übersicht siehe [86]) zur Diagnose in Telekommunikationsnetzen (siehe [1], z.B. [42]). Dies ist nur eine winzige, zufällige Auswahl von erfolgreichen Projekten, die Multiagentsystemtechnologie anwenden. Für jedes dieser Gebiete ließe sich ein Übersichtsartikel finden/schreiben, der den Umfang dieses Kapitels übertrifft.

AgentLink (www.agentlink.org), das Europäische Network of Excellence of Agent-based Computing, schaffte es über fast zehn Jahre, weitgehend alle Europäischen Forschungsinstitute, die sich mit Multiagentensystemen beschäftigen, zu vereinigen. Am Ende der Laufzeit, 2006, wurde eine Roadmap⁵ [58] publiziert, die Erreichtes dokumentierte und neue Trends identifizierte. Ein Schwerpunkt wurde dabei auch auf den industriellen Einsatz von agentenbasierter Technologie gelegt. Die Agentlink-Roadmap dokumentiert sehr einprägsam den Stand verschiedener agentenbasierter Technologien und Anwendungsgebiete. AgentLink sammelte zudem prominente Fallstudien für den industriellen Einsatz von Agententechnologie.

14.5 Aktuelle Trends

In diesem Kapitel sollte ein Überblick über Multiagentensysteme im Kontext der Künstlichen Intelligenz gegeben werden. Hierbei haben wir den Fokus hauptsächlich auf Interaktionen gelegt und weniger auf die Ideen, die möglicherweise zentraler für die Künstliche Intelligenz sind. Wir haben auf Themen aus dem Bereich der formalen Behandlung von Multiagentensysteme, z.B. Multiagentenlogiken, verzichtet. Mit dem Fortschritt bei der Entwicklung und Anwendung von Multiagentensystemen, wird die Frage nach der formalen Verifikation immer interessanter. Die Beweisbarkeit, dass ein Multiagentensystem wirklich die vorgegebenen Eigenschaften erfüllt, ist und bleibt eine wichtige Voraussetzung, derart komplexe Systeme in der Realität auch in sicherheitskritischen Anwendungen einsetzbar zu machen. Ebenso haben wir den interessanten Bereich des Multiagentenlernens ignoriert, in dem Agenten voneinander, miteinander oder

⁵ Die Roadmap ist immer noch von www.agentlink.org downloadbar.

in Konkurrenz zueinander lernen. Multi-Robot Systeme oder Multiagentensysteme aus Virtuelle Agenten haben in den letzten Jahren ihren Platz in der Mainstream Multiagentensystem-Forschung eingenommen, auch weil die technologischen Hürden durch kommerzielle Roboterplattformen, wie z.B. die Nao-Robots (<http://www.aldebaran-robotics.com>), ebenso wie effizientere und mächtigere Game Engines, wie z.B. die MASSIVE Plattform (<http://www.massivesoftware.com/>), niedriger geworden sind.

Eine interessante Forschungsrichtung, die wir ebenfalls kaum betrachtet haben, hat sich unter dem Schlagwort „Agreement Technology“ formiert. Dabei sollen Agenten entwickelt werden, die auf menschlichem Niveau miteinander verhandeln, zu einer Einigung kommen und entsprechende Abmachungen treffen können. Forscher aus den Bereichen Normative Multiagentensysteme, Multiagentenorganisationen, Argumentation und Verhandlung und Technologien, die Vertrauen und Reputation zwischen und von Agenten unterstützen, sind an einer gleichnamigen COST-Aktion beteiligt (<http://www.agreement-technologies.org/>). Ein Buch, das die Ergebnisse dieser COST Aktion zusammenfasst, erscheint Ende 2012.

Literaturverzeichnis

- [1] Albayrak, S., editor (1998). *Intelligent Agents For Telecommunications Application*. IOS Press.
- [2] Austin, J. (1962). *How to do things with words*. Cambridge.
- [3] Balmer, M., Nagel, K., und Raney, B. (2004). Large scale multi-agent simulations for transportation applications. *J. of Intelligent Transport Systems*, 8:205–223.
- [4] Barbuceanu, M. und Fox, M. S. (1995). COOL: A language for describing coordination in multi agent systems. In *Proc. of the 1st. Int. Conf. on Multiagent Systems*, pages 17–24. AAAI.
- [5] Bauer, B. und Odell, J. (2005). UML 2.0 and agents: how to build agent-based systems with the new UML standard. *Eng. Appl. Artif. Intell.*, 18(2):141–157.
- [6] Bazzan, A. L. C. (2009). Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *Autonomous Agents and Multi-Agent Systems*, 18(3):342–375.
- [7] Bellifemine, F. L., Caire, G., und Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE*. John Wiley.
- [8] Bergenti, F., Gleizes, M.-P., und Zambonelli, F., editors (2004). *Methodologies and Software Engineering For Agent Systems – The Agent-Oriented Software Engineering Handbook*. Kluwer Academic Publishing, New York.
- [9] Bernon, C., Cossentino, M., Gleizes, M.-P., Turci, P., und Zambonelli, F. (2005a). A study of some multi-agent meta-models. In Odell, J., Giorgini, P., und Müller, J. P., editors, *Agent-Oriented Software Engineering V, 5th International Workshop, AOSE 2004, New York, NY, USA, July 19, 2004, Revised Selected Papers*.
- [10] Bernon, C., Cossentino, M., und Pavon, J. (2005b). Agent oriented software engineering. *Knowledge Engineering Review*, 20:99–116.
- [11] Bernon, C., Gleizes, M.-P., Peyruqueou, S., und Picard, G. (2003). Adelfe: A methodology for adaptive multi-agent systems engineering. In Petta, P., Tolksdorf, R., und Zambonelli, F., editors, *Engineering Societies in the Agents World III*, volume 2577 of *Lecture Notes in Computer Science*, pages 70–81. Springer Berlin / Heidelberg.

- [12] Beydoun, G., Low, G., Henderson-Sellers, B., Mouratidis, H., Gomez-Sanz, J. J., Pavón, J., und Gonzalez-Perez, C. (2009). FAML: A generic metamodel for MAS development. *IEEE Transactions on Software Engineering*, 35(6):841–863.
- [13] Bonjean, N., Gleizes, M.-P., Maurel, C., und Migeon, F. (2012). Forward self-combined method fragments. In *Proc. of the 13th International Workshop on Agent-Oriented Software Engineering, Valencia, 2012*.
- [14] Bordini, R., Braubach, L., Dastani, M., Seghrouchni, A. E. F., Gomez-Sanz, J. J., Leite, J., G. O'Hare, G., Pokahr, A., und Ricci, A. (2006). A survey of programming languages and platforms for multi-agent systems. *Informatica*, 30:33–44.
- [15] Bordini, R., Hübner, J., und Wooldridge, M. (2005). *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley.
- [16] Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., und Mylopoulos, J. (2004). Tropos: An agent-oriented software development methodology. *Autonomous Systems and Multi-Agent Systems*, 8:203–236.
- [17] Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14–23.
- [18] Campbell, A. und Wu, A. (2011). Multiagent role allocation: issues, approaches and multiple perspectives. *Autonomous Agents and Multiagent Systems*, 22:317–355.
- [19] Carley, K. M. und Gasser, L. (1999). Computational organization theory. In Weiss, G., editor, *Multiagent systems*, pages 299–330. MIT Press, Cambridge, MA, USA.
- [20] Chaib-Draa, B. und Müller, J. P., editors (2006). *Multiagent based Supply Chain Management*. Springer.
- [21] Corkill, D. D. (1991). Blackboard systems. *AI Expert*, 6:40–47.
- [22] Cossentino, M. (2005). From requirements to code with the PASSI methodology. In Henderson-Sellers, B. und Giorgini, P., editors, *Agent-Oriented Methodologies*, pages 79–106. Idea Group.
- [23] Cost, R. S., Chen, Y., Finin, T. W., Labrou, Y., und Peng, Y. (2000). Using colored petri nets for conversation modeling. In *Issues in Agent Communication*, pages 178–192, London, UK, UK. Springer-Verlag.
- [24] da Silva, V. T., Choren, R., und de Lucena, C. J. P. (2008). Mas-ml: a multi-agent system modelling language. *Int. J. Agent-Oriented Softw. Eng.*, 2(4):382–421.
- [25] Dasgupta, P., Narasimhan, N., Moser, L. E., und Melliar-Smith, P. M. (1999). MAgNET: mobile agents for networked electronic trading. *Knowledge and Data Engineering, IEEE Transactions on*, 11(4):509 –525.
- [26] de Jong, S. und Tuyls, K. (2011). Human-inspired computational fairness. *Autonomous Agents and Multiagent Systems*, 22:102–126.
- [27] Decker, K. S. und Lesser, V. R. (1992). Generalizing the partial global planning algorithm. *Int. Journal of Intelligent and Cooperative Information Systems*, 1:319–346.
- [28] DeLoach, S. A. und Garcia-Ojeda, J. C. (2010). O-MaSE: a customizable approach to designing and building complex, adaptive multiagent systems. *Int. Journal of Agent-Oriented Software Engineering*, 4:244–280.
- [29] Dignum, V. (2004). *A Model for Organizational Interaction: Based on Agents, Founded in Logic*. PhD thesis, Utrecht University.
- [30] Dorigo, M. und Stützle, T. (2004). *Ant Colony Optimization*. MIT Press.
- [31] Drogoul, A. und Dubreuil, J. (1992). Eco-problem solving: Results of the n-puzzle. In Demanzeau, Y. und Werner, E., editors, *Decentralized AI III*. North Holland.

- [32] Durfee, E. H. (1999). Distributed problem solving and planning. In Weiss, G., editor, *Multiagent systems*, pages 121–164. MIT Press, Cambridge, MA, USA.
- [33] Durfee, E. H. und Lesser, V. R. (1991). Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21:1187–1183.
- [34] Durfee, E. H. und Rosenschein, J. S. (1994). Distributed problem solving and multi-agent systems: Comparisons and examples. In *AAAI Technical Report WS-94-02*, pages 52–62.
- [35] Eisenhardt, K. M. (1989). Agency theory: An assessment and review. *The Academy of Management Review*, 14:57–74.
- [36] Ferber, J. (1999). *Multi-Agent System: An Introduction to Distributed Artificial Intelligence*. Addison Wesley Longman.
- [37] Ferber, J., Gutknecht, O., und Michel, F. (2004). From agent to organizations: an organizational view of multi-agent systems. In Giorgini, P., Müller, J., und Odell, J., editors, *Agent-Oriented Software, AOSE IV, Melbourne 2003*, volume LNCS 2935, pages 214–230. Springer.
- [38] Ferber, J. und Müller, J.-P. (1996). Influences and reactions: a model of situated multiagent systems. In *Proc. of the ICMAS'96, International Conference on Multi-Agent Systems*. AAAI Press.
- [39] Ferguson, I. A. (1992). Towards an architecture for adaptive, rational, mobile agents. In Werner, E. und Demazeau, Y., editors, *Decentralized A.I. 3 – 3rd European Workshop on Modellings Autonomous Agents in a Multi-Agent World, MAAMAW'91*, pages 249–263. North Holland.
- [40] Finin, T., Fritzson, R., McKay, D., und McEntire, R. (1994). KQML as an agent communication language. In *Proc. of the 3rd Int. Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463. ACM Press.
- [41] Franklin, S. und Graesser, A. (1997). It is an agent, or just a program? a taxonomy for autonomous agents. In Jennings, N. und Wooldridge, M., editors, *Intelligent Agents*, volume III. Springer.
- [42] Garcia-Gomez, S., Gonzalez-Ordas, J., Garcia-Algarra, F. J., Toribio-Sardon, R., Sedano-Frade, A., und Buisan-Garcia, F. (2009). KOWLAN: A multi agent system for bayesian diagnosis in telecommunication networks. In *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT '09*, pages 195–198.
- [43] Glaschenko, A., Ivaschenko, A., Rzevski, G., und Skobelev, P. (2009). Multi-agent real time scheduling system for taxi companies. In Decker, K., Sichman, J., Sierra, C., und Castelfranchi, C., editors, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, May 2009*, pages 29–36.
- [44] Gómez-Sanz, J. J. und Pavón, J. (2002). Agent oriented software engineering with message. In Giorgini, P., Lespérance, Y., Wagner, G., und Yu, E. S. K., editors, *AOIS '02, Agent-Oriented Information Systems, Proceedings of the Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002 at CAiSE*02)*.
- [45] Hahn, C., Madrigal-Mora, C., und Fischer, K. (2009). A platform independent meta model for multi-agent systems. *Journal of Autonomous Agents and Multi-Agent Systems*, 18(2):239–266.
- [46] Henderson-Sellers, B. und Giorgini, P. (2005). *Agent-Oriented Methodologies*. IDEA Group, Hershey.
- [47] Hewitt, C. und Inman, J. (1991). DAI betwixt and between: from ‘intelligent agents’ to open systems science. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(6):1409–1419.

- [48] Holland, J. H. (1999). *Emergence: From Chaos to Order*. Helix Books.
- [49] Horling, B. und Lesser, V. (2004). A survey of multi-agent organizational paradigms. *Knowledge Engineering Review*, 19(4):281–316.
- [50] Hübner, J. F., Boissier, O., Kitio, R., und Ricci, A. (2010). Instrumenting multi-agent organizations with organisational artifacts and agents: “giving the organisational power back to the agents”. *Autonomous Agents and Multi-Agent Systems*, pages 369–400.
- [51] Hübner, J. F., Sichman, J. S., und Boissier, O. (2002). A model for the structural, functional, and deontic specification of organizations in multiagent systems. In *Proceedings of the 16th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence*, SBIA ’02, pages 118–128, London, UK, UK. Springer-Verlag.
- [52] Huhns, M. N. und Singh, M. P., editors (1999). *Readings in Agents*. Morgan Kaufmann.
- [53] Iglesias, C. A. und mercedes Garijo (2005). The agent-oriented methodology MAS-CommonKADS. In Henderson-Sellers, B. und Giorgini, P., editors, *Agent-Oriented Methodologies*, pages 46–78. Idea Group Publishers.
- [54] Ingrand, F. F., Georgeff, M. P., und Rao, A. S. (1992). An architecture for real-time reasoning and system control. *IEEE Expert*, 7(6):34–44.
- [55] Klügl, F. (2004). Applications of software-agents. *Künstliche Intelligenz*, 18:5–10.
- [56] Kraus, S. (2001). Automated negotiation and decision making in multiagent environments. In *Multiagent Systems and Applications, Selected Tutorial Papers*, pages 150–172. Springer-Verlag.
- [57] Labrou, Y. (2001). Standardizing agent communication. In Luck, M., Marik, V., Stepankova, O., und Trappel, R., editors, *Multiagent Systems and Applications, Selected Tutorial Papers*, LNAI 2086, pages 74–91, Berlin, Heidelberg. Springer.
- [58] Luck, M., McBurney, P., Shehory, O., und Willmott, S. (2005). *Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing)*. AgentLink.
- [59] Maes, P. (1992). Behavior-based artificial intelligence. In Meyer, J.-A. und Wilson, S. W., editors, *From Animals to Animats 2: 2nd Conference on the Simulation of Adaptive Behavior*, pages 2–10. MIT Press.
- [60] Mao, X. und Yu, E. (2005). Organizational and social concepts in agent oriented software engineering. In *Proceedings of the 5th international conference on Agent-Oriented Software Engineering*, AOSE’04, pages 1–15, Berlin, Heidelberg. Springer-Verlag.
- [61] Müller, J. P. (1996). *The Design of Intelligent Agents – A Layered Approach*, volume 1177 of *LNAI*. Springer.
- [62] Nilsson, N. (1994). Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 1:139–158.
- [63] Odell, J., Parunak, H. V. D., und Bauer, B. (2000). Extending UML for agents. In *Proc. of the Agent-Oriented Information Systems, Workshop at the 17th National Conference on Artificial Intelligence*, pages 3–17.
- [64] Padgham, L. und Winikoff, M. (2005). Prometheus: A practical agent-oriented methodology. In Henderson-Sellers, B. und Giorgini, P., editors, *Agent-Oriented Methodologies*, pages 107–135. Idea Group Inc.
- [65] Parunak, H. V. D. (1987). Manufacturing experience with the contract net. In Huhns, M. N., editor, *Distributed Artificial Intelligence*, pages 285–310.
- [66] Patil, R. S., Fikes, R. E., Patel-Schneider, P. F., MacKay, D., Finin, T., Gruber, T., und Neches, R. (1992). The DARPA knowledge sharing effort: Progress report. In *Proceedings of the 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 777–787. Reprinted in M. Huhns and M. Singh. *Readings in Agents*, 1999.

- [67] Pavón, J., Gómez-Sanz, J. J., und Fuentes, R. (2005). The INGENIAS methodology and tools. In Henderson-Sellers, B. und Giorgini, P., editors, *Agent-Oriented Methodologies*, pages 236–276. Idea Group Publishing.
- [68] Pokahr, A. und Braubach, L. (2009). From a research to an industrial-strength agent platform: Jadex v2. In Hansen, H. R., Karagiannis, D., und Fill, H.-G., editors, *Business Services: Konzepte, Technologien, Anwendungen – 9. Internationale Tagung Wirtschaftsinformatik (WI 2009)*, pages 769–778. Österreichische Computer Gesellschaft.
- [69] Pěchouček, M. und Mařík, V. (2008). Industrial deployment of multi-agent technologies: review and selected case studies. *Autonomous Agents and Multi-Agent Systems*, 17(3):397–431.
- [70] Rao, A. S. (1996). AgentSpeak(L): BDI agents speak out in a logical computable language. In *Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world : agents breaking away: agents breaking away*, MAAMAW ’96, pages 42–55. Springer-Verlag New York, Inc.
- [71] Rao, A. S. und Georgeff, M. P. (1991). Modeling rational agents within a bdi-architecture. In Allen, J. F., Fikes, R., und Sandewall, E., editors, *Proc. of the 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning (KR’91). Cambridge, MA, USA, April 22–25, 1991*, pages 473–484. Morgan Kaufmann.
- [72] Rosenschein, J. S. und Zlotkin, G. (1994). *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. MIT Press.
- [73] Rossi, D., Cabri, G., und Denti, E. (2000). Tuple-based technologies for coordination. In Omicini, A., Zambonelli, F., Klusch, M., und Tolksdorf, R., editors, *Coordination of Internet Agents: Models, Technologies and Applications*. Springer, Berlin.
- [74] Russell, S. und Norvig, P. (1995). *Artificial Intelligence – A Modern Approach*. Prentice Hall, 3rd edition.
- [75] Sandholm, T. (1999). Distributed rational decision making. In Weiss, G., editor, *Multiagent systems*, pages 201–258. MIT Press, Cambridge, MA, USA.
- [76] Schoppers, M. (1987). Universal plans for reactive robots in unpredictable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-87*, 1987.
- [77] Searle, J. (1962). *Speech Acts*. Cambridge.
- [78] Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., und Meyer, G. (2012). PROTECT: a deployed game theoretic system to protect the ports of the united states. In van der Hoek, W., Padgham, L., Conitzer, V., und Winikoff, M., editors, *Int. Conf. on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 2012*, pages 13–20. IFAAMAS.
- [79] Shoham, Y. (1993). Agent-oriented programming. *Artificial Intelligence*, 60:51–92.
- [80] Shoham, Y. und Leyton-Brown, K. (2009). *Multiagent Systems – Algorithmic, Game-Theoretic and Logical Foundations*. Cambridge University Press.
- [81] Sislak, D., Volf, P., Kopriva, S., und Pechoucek, M. (2012). *AgentFly: Scalable, High-Fidelity Framework for Simulation, Planning and Collision Avoidance of Multiple UAVs*. John Wiley&Sons.
- [82] Smith, R. G. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Comput.*, 29(12):1104–1113.
- [83] ter Mors, A., Yadati, C., Witteveen, C., und Zhang, Y. (2010). Coordination by design and the price of autonomy. *Autonomous Agents and Multiagent Systems*, 20:308–341.

- [84] Torres, D. (2008). On virtual environments and agents in next-generation computer games. *The Knowledge Engineering Review*, 23:389–397.
- [85] Vazquez-Salceda, J., Dignum, V., und Dignum, F. (2005). Organizing multiagent systems. *Autonomous Agents and Multiagent Systems*, 11:307–360.
- [86] Vinyals, M., Rodriguez-Aguilar, J. A., und Cerquides, J. (2005). A survey on sensor networks from a multiagent perspective. *The Computer Journal*, 54:455–470.
- [87] Vlassis, N. (2007). *Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence*. Synthesis Lectures in Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.
- [88] Vytelingum, P., Ramchurn, S. D., Voice, T. D., Rogers, A., und Jennings, N. R. (2010). Trading agents for the smart electricity grid. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems, Toronto, Canada*, pages 897–904.
- [89] Weiss, G., editor (1999). *Multiagent Systems – A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, USA.
- [90] Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. John Wiley, Chichester, 2nd edition.
- [91] Zambonelli, F., Jennings, N. R., und Wooldridge, M. (2005). Multiagent systems as computational organisations: the Gaia methodology. In Henderson-Sellers, B. und Giorgini, P., editors, *Agent-Oriented Methodologies*, pages 136–171. Idea Group Publishers.
- [92] Zambonelli, F. und Parunak, H. V. D. (2003). Signs of a revolution in computer science and software engineering. In *Proceedings of the 3rd international conference on Engineering societies in the agents world III*, pages 13–28, Berlin, Heidelberg. Springer-Verlag.

Teil III

Anwendungen

15 Verkörperte Kommunikation mit kognitiven virtuellen Agenten

Nadine Pfeiffer-Leßmann, Stefan Kopp und Ipke Wachsmuth

15.1 Einleitung

Miteinander zu kommunizieren ist eine der komplexesten und wichtigsten Fähigkeiten des Menschen, deren Erforschung in der Künstlichen Intelligenz immer schon eine zentrale Rolle gespielt hat. Abgeleitet von dem lateinischen *communicari* („teilen, mitteilen“) wird Kommunikation als Informationsaustausch zwischen Personen verstanden, oder meist etwas spezieller als wechselseitige, weitgehend beabsichtigte Informationsübertragung zwischen mindestens zwei Partnern. Die verbreitete, auf die Kommunikationstheorie von Shannon und Weaver zurückgehende Vorstellung, dass bei diesem Informationsaustausch ein „Sender“ Informationen verschlüsselt, die ein „Empfänger“ aufnimmt und entschlüsselt, hat dabei auch klassische Kommunikationsmodelle in der KI geprägt [77] und ist bis heute in der Mensch-Maschine-Interaktion stark verbreitet. Ein Vielzahl von empirischen Befunden in den kognitiven Disziplinen hat seitdem aber deutlich gemacht, dass dieses Modell für ein Verstehen und Modellieren der menschlichen Kommunikation – speziell der Kommunikation von Angesicht zu Angesicht – nicht ausreichend ist. Insbesondere die körperliche Einbettung kognitiver Agenten hat für ihre Kommunikation eine große Bedeutung, sowohl bezüglich der Signale, über die sie interagieren, als auch bezüglich der kognitiven Prozesse, die dabei zugrunde liegen. Beides wird durch den Begriff „verkörperte Kommunikation“ (embodied communication [75]) in den Mittelpunkt gestellt und im vorliegenden Kapitel für die Mensch-Maschine-Kommunikation und die Erforschung virtueller verkörperter Agenten thematisiert.

Eine Vielzahl jüngerer Arbeiten in der Künstlichen Intelligenz und Robotik greift die Fragestellung der verkörperten Kommunikation in technischem Zusammenhang auf. Neben dem Aspekt der technischen Machbarkeit sind diese Arbeiten mit der Erwartung verbunden, durch die Entwicklung und den Test konkreter Modelle neue Erkenntnisse über das Funktionieren menschlicher Kommunikation zu gewinnen. Wie funktioniert beispielsweise das zeitliche und inhaltliche Zusammenspiel von Sprechen und Gestikulieren? Welche Rolle spielen Emotionen in der Kommunikation? Wie wird das Abwechseln im Dialog gesteuert? Wie koordinieren sich Interaktionspartner miteinander? Der Ansatz, technische Kommunikationspartner zu entwickeln, verlangt immer auch eine Behandlung der Frage, welche internen Mechanismen die entsprechenden Fähigkeit des Agenten erbringen können. Speziell widmen sich die Arbeitsrichtungen der „embodied conversational agents“ [17] und der „virtual humans“ [32] dem Ziel, Schlüsselemente der Verkörperung in der Kommunikation zu modellieren und auf der anderen Seite anthropomorphe Assistenzsysteme mit zahlreichen Anwendungsperspektiven zu erarbeiten. Oberflächlich betrachtet streben solche Systeme eine multimodale Interaktion an, welche ver-

bale und non-verbale Signale der menschlichen Konversation umfasst; d.h. Sprache, Mimik, Gesten oder Körperhaltung werden von beiden Interaktionspartnern eingesetzt, um zum Dialog beizutragen. Intern liegen komplexe Verhaltensmodelle und Agentenarchitekturen zugrunde, um z.B. mit der großen Varietät von kommunikativen Verhaltensweisen, ihrer feinen zeitlichen Synchronisierung und der Dynamik der Wechselwirkung zwischen Kommunikationspartnern umzugehen.

Im Folgenden werden Grundlagen der verkörperten Kommunikation samt wichtiger Fachbegriffe eingeführt. Danach werden technische Lösungen für die Modellierung einzelner Aspekte natürlicher verkörperter Kommunikation behandelt. Anschließend wird anhand des virtuellen Agenten Max ein Beispiel eines verkörperten Kommunikationspartners vorgestellt und so ein Einblick in ein komplexes Gesamtsystem vermittelt. Zum Schluss werden weiterführende Forschungsfragen auf dem Weg hin zu natürlich und verkörpert kommunizierenden Agenten thematisiert.

15.2 Grundlagen

Der Begriff „verkörperte Kommunikation“ (embodied communication) [75] betont die zentrale Rolle der Körperlichkeit kognitiver Agenten in ihrer Kommunikation. Eine genaue Analyse der Verwendung von körperlichen Signalen macht deutlich, wie vielschichtig und nebenläufig menschliche Kommunikation ist: Während des Sprechens produzieren wir einen Strom von körperlichen Signalen. Gleichzeitig nehmen wir dieselben Signale des Adressaten wahr und achten darauf, ob er aufmerksam ist und an den entsprechenden Stellen wie erwartet reagiert (z.B. einen Blick erwidert, mit einem Nicken Verstehen signalisiert oder durch ein Heben der Hand das Rederecht anfordert). Darüber hinaus passen sich Kommunikationspartner in vielfältiger Hinsicht aneinander an. Sie tendieren z.B. dazu dieselben Wörter zu verwenden, ihre Sprechgeschwindigkeit oder Tonhöhe anzugeleichen oder ihre Körperhaltung zu imitieren [40]. Auch in ihren Gesten sind sie sensiv für die Gesten anderer [10].

Diese Beobachtungen haben Konsequenzen für die Forschung in der Künstlichen Intelligenz, der humanoiden Robotik und der Mensch-Maschine-Kommunikation. Sie verlangen ein vollständigeres und korrekteres Verständnis von Interaktion, das – über symbolische Kommunikation hinaus – systematisch den Einbezug physischer Ressourcen verlangt. Als Herausforderung für die Konstruktion verkörperter kommunizierender Agenten impliziert dies den Entwurf operationaler Modelle, die spezifizieren, wie mentale Prozesse und Verkörperung in der Kommunikation und innerhalb eines kognitiven Agenten zusammenwirken.

15.2.1 Begriffe der Kommunikation

Dialog, Sprechakte und andere Handlungen

McTear definiert einen Dialog als eine gemeinsame, kooperative Handlung zwischen zwei oder mehreren Beteiligten [51]. In der Sprachwissenschaft besteht eine lange Tradition darin, Sprechen als eine Form des Handelns zu sehen [5, 68]. Austin betonte, dass verschiedene „Kräfte“ in einer Äußerung wirken: Der *lokutionäre* Akt der reinen Äußerung von Wörtern, der *perlokutionäre* Akt des gewünschten Effekts auf den Adressaten oder die Welt und den *illokutionären* Akt, eine bestimmte Aktion zur Herstellung des perlokutionären Ziels. Basierend auf dieser

Definition prägt Searle den Begriff des „Sprechakts“ (speech act), welcher sowohl einen propositionalen Inhalt als auch eine mentale Absicht der Verwendung umfasst. Andere Forscher haben für verwandte Konzepte Begriffe wie *communicative act* [1], *conversational move* [15] oder *dialogue move* [20] geprägt. [58] definieren einen kommunikativen Akt als die kleinste Einheit der Kommunikation, welche aus einem Signal und einer Bedeutung besteht. Zur Bedeutung gehört dabei ein propositionaler Inhalt sowie ein *Performativ*, welches die Aktion repräsentiert, die der Akt ausführt. Die Modelle unterscheiden hierbei zwischen verschiedenen Aktionen. Poggi und Kollegen etwa nehmen allgemein drei unterschiedliche Typen von Performativen an: Präsentation von Informationen (*inform*), Fragen (*query*) und Aufforderungen (*request*). Andere Taxonomien unterteilen in weit feinere Klassen, z.B. das DAMSL-Annotationssystem [21] oder die DIT++-Taxonomie von kommunikativen Funktionen [13]. Letztere umfasst mehr als 100 verschiedenen Funktionen, grob unterteilt in *General Purpose Communicative Functions* wie Informationstransfer, Fragen oder Direktiven und *Dialog Control Functions*, die sich auf Aktionen zur Koordinierung der Kommunikation mit Aushandlung des Rederechts (*Turn-Taking*) oder der Etablierung von Informationen (*Grounding*) beziehen. Darüber hinaus erfüllen kommunikative Handlungen oft auch *sozio-emotionalen Funktionen*, die die Beeinflussung der Wahrnehmung und Einschätzung der eigenen Person durch andere betreffen [52].

Multimodalität

Laut Argyle laufen mehr als 65 Prozent des Austausches in einem Gespräch über nicht-sprachliche Kanäle wie Gesten, Körperhaltung, Mimik oder Sprachmelodie ab [3]. Allgemein lassen sich *verbale* von *non-verbalen* Modalitäten unterscheiden, die letzteren können als vokalisch und nicht vokalisch differenziert werden. Sprache ist gut dafür geeignet, symbolische Informationen zu übertragen. Der Ausdruck in Gesicht und Stimme kann z.B. emotionale Zustände übermitteln. Gesten eignen sich besonders für die Vermittlung von visuellen räumlichen Informationen wie die Form eines Objekts oder den direkten Verweis auf ein externes Objekt oder eine Position (wie in Abb. 15.1 dargestellt). Durch ikonische Gestik kann ein Vorstellungsbild in verkörperter Form extern repräsentiert und übermittelt werden [50]. Ein solches gestisches Zeichen trägt Bedeutung durch bildliche Ähnlichkeit zu einzelnen wichtigen Aspekten des vorgestellten Bezugsobjekt in sich. Zudem können non-verbale Signale wie Körperhaltung, Mimik, Gestik und Blicke Intentionen vermitteln und bei der Steuerung des Dialogs (z.B. beim Aushandeln des Rederechts) mitwirken. *Paraverbale* Anteile einer Äußerung umfassen die Art und Weise des Sprechens (Stimmlage, Intonation, Lautstärke sowie Sprechtempo, Sprechpausen und Sprachmelodie) und können das Gesagte – oft im Zusammenspiel mit einem Gesichtsausdruck – modifizieren.

Turn-Taking

Turn-Taking stellt einen interaktiven Basismechanismus dar, um die Ablaufplanung des Rederechts während einer Konversation zu koordinieren [23], und sichert den kooperativen Ablauf einer Interaktion unabhängig davon, ob die inhaltlichen Ziele der Kommunikationspartner kooperativ sind [29]. Die Konversationsanalyse geht von einem kontextfreien, regelbasierten Mechanismus aus [66], mit dem lokal an übergaberelevanten Stelle der Sprecherwechsel erfolgt. Dagegen hat Duncan [23] im Rahmen eines signalbasierten Ansatzes dokumentiert, dass eine Aushandlung der Sprecherrolle durch interaktive Signale geleitet wird. Spätere Dialogtheorien [18, 30] greifen beides auf. Goodwin [31] hält einen signalbasierten Ansatz allein für nicht ausreichend, um die Phänomene des Turn-Takings zu erklären. Er sieht eine Aushandlung des Turns als zeitgebundenen kooperativen Prozess zwischen Sprecher und Hörer. Nach



Abbildung 15.1: Multimodale Kommunikation mit Sprache und Gestik.

Clark's Dialogtheorie [18] stellt Turn-Taking dazu eine Ebene des Dialogmanagements dar, auf der dezidierte Aktionen wirken, z.B. Ergreifen des Turns (take-turn), Anfordern des Turns (request-turn), Aufgabe des Turns (release-turn), Halten des Turns (hold-turn), Zuweisen des Turns (assign-turn).

Initiative

Während die Mechanismen des Turn-Takings auf der interaktionalen Ebene operieren, lässt sich auf der inhaltlichen Ebene ebenfalls ein Abwechseln der Einflüsse von Interaktionspartnern ausmachen. Man spricht von „Initiative“ und unterscheidet mindestens zwei verschiedene Ebenen [19]: Auf linguistischer Ebene betrifft Initiative die Kontrolle im Dialogablauf, und Initiativwechsel treten oft zeitgleich mit der Übergabe oder Übernahme des Turns auf. Auf Problemlösungsebene bezieht sich Initiative auf das Vorantreiben des Dialogs zu einem Ziel. Je nach Rolle, Ziele und Kompetenz kann die Initiative nur bei einem Dialogpartner verbleiben oder dynamisch zwischen den Dialogpartnern wechseln. Ersteres (*system-initiative*) ist häufig der Fall in technischen Sprachdialogsystemen. Letzteres (*mixed-initiative*) ist eine Voraussetzung für gleichberechtigtes Problemlösen und Kooperieren, mit verschachtelten Beiträgen der Interaktanten und asynchronen Initiativwechseln [34].

Feedback und Engagement

Während das klassische Sender-Empfänger-Modell von klar getrennten Rollen zu jedem Zeitpunkt ausgeht, funktioniert natürlicher Dialog anders. Dort produzieren die Kommunikationsteilnehmer Feedback- und Engagement-Signale, auch wenn sie gerade nicht im Besitz des Turns sind und sprechen [41]. Man spricht daher von „Backchannel-Signalen“ [79], die eine wichtige Rolle im Prozess des Groundings (s.u.) spielen. Eine andere Rolle betrifft das Signalisieren von „Engagement“, durch das ein oder mehrere Teilnehmer während einer gemeinsamen Interaktion eine Verbindung eingehen und aufrechterhalten [69]. Engagement-Signale verdeutlichen, dass ein Agent in eine Interaktion stark involviert ist, steuern eine Kooperation und fördern die Effizienz und Reibungslosigkeit einer Interaktion. Engagement kann sowohl durch verbale als auch durch non-verbale Signale gefördert werden. Im Bereich der non-verbalen Signale kann insbesondere der Aufmerksamkeitsfokus, repräsentiert durch die Blickfokussierung der Interaktionspartner, als zentrales Engagement-Signal eingesetzt werden [4, 23, 39].

Interpersonale Koordination

In der verkörperten Kommunikation kommt der gegenseitigen Anpassung von Interaktionspartnern großes Gewicht zu, da es oft mit automatischen Prozessen der Kopplung zwischen Wahrnehmung und Motorik in Verbindung gebracht wird [40]. In der Tat ist Verhaltensangleichung – vor allem non-verbale – in der Interaktion ein umfassend belegtes Phänomen, dem eine positive soziale Funktion zugeschrieben wird [44]. Einige Dialogtheorien – allen voran der Alignment-Ansatz nach [57] – nehmen an, dass gegenseitige Anpassung universell und automatisch ist und auch auf linguistischen Ebenen wirkt (von phonologischer über lexikalische bis hin zu syntaktischer und semantischer Angleichung). Ihr wird dementsprechend eine kommunikative Funktion zugeschrieben. Dies ist im Gegensatz zu klassischen Dialogtheorien zu sehen, die eine zentrale Rolle des kooperativen Etablierens („Grounding“) von Inhalten betonen. Demnach werden spezifische explizite Mechanismen, von Feedback bis hin zur relevanten nächsten Äußerung, eingesetzt und bei der Verarbeitung sowie die Produktion von Äußerungen berücksichtigt [12, 18].

15.3 Technische Ansätze

Die Ermöglichung verkörperter Kommunikation zwischen Menschen und Maschinen ist ein Ziel für die Künstliche Intelligenz ebenso wie für die Mensch-Maschine-Interaktion. Im Folgenden werden speziell die technischen Ansätze besprochen, die zur Realisierung kognitiver virtueller Agenten mit multimodalen Kommunikationsfähigkeiten entwickelt werden.

15.3.1 Dialogmanagement

Traditionelle Ansätze der Dialogmodellierung lassen sich in *zustandsbasierte* (*state-based*) und *Frame-basierte* Ansätze einteilen [51]. Die *zustandsbasierten* Ansätze bauen auf explizit repräsentierten Dialogzuständen auf und definieren durch eine Dialoggrammatik, wie durch einen Dialogakt (dialogue act) ein Zustand in einen anderen Zustand überführt werden kann. Sie werden in Szenarien eingesetzt, in welchen mögliche Dialogverläufe im Vorhinein bestimmt werden können. *Frame-basierte* Ansätze machen lediglich Annahmen über die auszutauschenden Inhalte (nicht deren Ordnung) und zeichnen sich daher durch eine größere Flexibilität aus. Modelle, nach denen der Dialogverlauf dynamisch durch zielgerichtete Entscheidungsprozesse der Konversationspartner entsteht, werden oft als agentenbasierte Ansätze konzipiert. Diese Ansätze betrachten Dialog als kooperativen Prozess zwischen zwei oder mehreren rationalen Agenten, denen allgemeine mentale Einstellungen wie Beliefs, Ziele, Pläne, sowie spezielle Commitments und Obligationen zugeschrieben werden [73]. Mit planbasierten Techniken werden Äußerungen in Form von Sprechakten wie intentionale Akte behandelt, die formal als Operatoren mit Rollen, Vorbedingungen, Constraints und Effekten dargestellt werden. Dafür muss der sich fortlaufend entwickelnde Dialogzustand in Form dieser Wissensstrukturen repräsentiert und verwaltet werden. Viele Dialogsysteme bauen dazu zum Beispiel auf dem *Information State Approach* auf [74]. Hier wird davon ausgegangen, dass jeder Agent Zustände verwaltet, die Informationen über den bisherigen wie evtl. zukünftigen Diskurs auf verschiedenen Ebenen repräsentieren und in den Entscheidungsprozess einbeziehen. Der Zustand des Dialogs aus Sicht des Agenten ist hierdurch klar definiert. Jede Aktion eines Dialogteilnehmers wird als Dialogakt verstanden, für den diese Wissensstrukturen anhand von festgelegten Regeln kontextabhängig aktualisiert werden.

Ein weiterer wichtiger Aspekt des Dialogmanagements ist die Regulierung des Konversationsablaufs, insbesondere durch das Turn-Taking. Klassische Modelle gehen von einer begrenzten Anzahl von konversationalen Zuständen aus (z.B. *NotPresent*, *Present*, *UserTurn*, *HoldTurn*, *AgentTurn*), zusammen mit Übergängen zwischen diesen, die durch kontextabhängige interaktionale Funktionen (z.B. *give-turn*, *want-turn*, *take-turn*) ausgelöst werden können [17]. Diese Ansätze wurden durch Übergangszustände (*Gap*, *Overlap*) sowie durch interaktionale Funktionen (*yield-turn*, *hold-turn*) erweitert [49]. Neuere Arbeiten setzen einen nicht-deterministischen endlichen Automaten mit einer Kostenmatrix und entscheidungstheoretischen Prinzipien ein, um Turn-Taking Verhalten zu modellieren [62]. Nach [72] reicht es nicht aus, kontextsensitive Regeln aufzustellen, nach denen das Rederecht genommen oder abgegeben wird. Vielmehr müssen auch antizipative Mechanismen in der Agentenarchitektur verankert werden. Aufbauend auf Arbeiten von Goodwin und Duncan stellt Thorisson das Ymir-Turn-Taking-Modell (YTTM) auf, ein Schichtenmodell, welches den kompletten Wahrnehmen-Handeln-Prozess modelliert. Weitere Arbeiten erfassen die Regeln des Turn-Takings durch maschinelles Lernen [38] oder modellieren Turn-Taking in einer Einheit mit Feedback und den zugrunde liegenden Verstehens- und Antwortprozessen [41].

15.3.2 Multimodale Verhaltensverarbeitung

Die Verarbeitung der kommunikativen Signale des Gegenübers folgt zumeist einer Erkennungs-pipeline von der Verarbeitung der Sensordaten der verschiedenen Modalitäten über Vorverarbeitungsschritte, das Extrahieren von Merkmalen, Prozesse der Segmentierung und Mustererkennung, bis hin zur multimodalen Integration und Interpretation (siehe Abbildung 15.2). Dieser Ansatz wurde zum Beispiel für Sprache und Gesten umgesetzt, aber auch Blickverhalten, Mimik sowie die Körpераusrichtung können so verarbeitet werden. Sprache wird durch Mikrophone aufgenommen und weiter durch Spracherkennner verarbeitet. Für Gesten müssen die Position und Ausrichtung der Hände erkannt werden sowie die Winkel zwischen den einzelnen Fingern und damit die Postur der Hand. Die Erkennung wird z.B. mit Datenhandschuhen oder durch videobasierte Ansätze oder mit Infrarotkameras vorgenommen. Als Vorverarbeitungsschritte werden Filtermechanismen eingesetzt, zum einen um Rauschen zu reduzieren und zum anderen um ein Hand- bzw. Körpermodell auf die Daten zu projizieren, aus welchem dann Winkel und Positionsvektoren abgelesen werden können. Dazu gibt es kinematische Modelle, welche die Skelettstruktur der Gelenke und Gliedmaßen beschreiben, sowie dynamische Modelle, welche die Bewegung als Ergebnis von Kräften und Drehmomenten beschreiben. Der Vorteil der Körpermodelle besteht darin, dass fehlende Sensorinformationen ergänzt, Trajektorien vorhergesehen und Adaptationsprozesse eingesetzt werden können, um sich auf die Charakteristika des individuellen Benutzers einzustellen, sowie unmögliche Bewegungen und Posturen ausschließen zu können (Details siehe [70]).

Um aus den Sensordaten eine Geste identifizieren zu können, muss das Signal zunächst segmentiert werden, um die expressive Phase der Geste aus dem Datenstrom zu filtern. Zusammen mit den Daten des Spracherkenners können die Daten dann in einem n-dimensionalen Vektor aller zu einem bestimmten Zeitpunkt relevanten Sensordaten repräsentiert werden.

Sowohl Gesten- als auch Spracherkennung werden oft als Mustererkennungsprozesse aufgefasst. Dabei können Template-basierte wie auch merkmalsbasierte Ansätze eingesetzt werden [8]. Während bei Template-basierten Ansätzen die Klassifikationsklassen vorgegeben sind, wird bei den merkmalsbasierten Ansätzen von einer Menge maßgebender Merkmale ausge-

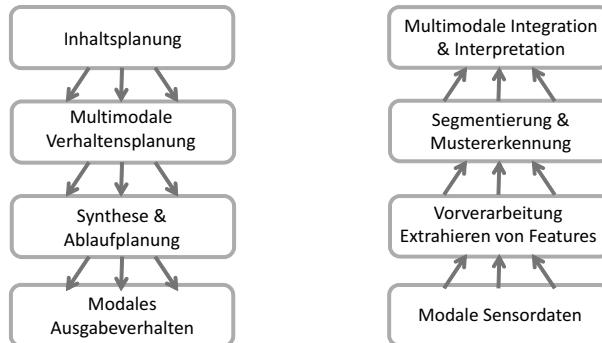


Abbildung 15.2: Allgemeiner Aufbau einer Pipeline für die Verarbeitung und Generierung kommunikativen Verhaltens.

gangen. [26] nutzt einen merkmalsbasierten Ansatz, der auf das Beschreibungssystem HamNoSys [60] aufsetzt. Dieses Framework wurde später für multimodale Eingaben erweitert und besonders für Echtzeitinteraktionen in der Virtuellen Realität angepasst [46].

Die Ansätze lassen sich anhand des Zeitpunkts, zu dem die Informationen der verschiedenen Modalitäten integriert werden, unterscheiden. Wenn die Daten sehr eng miteinander verbunden sind, können sie auf der Merkmalsebene früh zusammengeführt werden („early fusion“; beispielsweise Sprachsignale und Lippenbewegungen). Auf semantischer Ebene können Informationen später integriert werden („late fusion“; dabei werden die semantischen Informationen der einzelnen Modalitätskanäle zu einer übergeordneten Bedeutung zusammengeführt. Für die Bedeutungsspezifikation werden Attribut-Wert-Matrizen (AVMs) oder Frames genutzt [76]. Eine Alternative zum Frame-basierten Ansatz besteht darin, endliche Automaten zu verwenden [37]; dieser Ansatz wurde in [47] um zeitliche Aspekte erweitert, sodass multimodale Grammatiken mit erweiterten Transitionsnetzwerken (augmented transition networks) eingesetzt werden können.

15.3.3 Multimodale Verhaltensgenerierung

Analog zu der Pipeline der Verhaltensverarbeitung wird kommunikatives Verhalten in der Regel in mehreren Schritten generiert (siehe Abbildung 15.2). Die Struktur baut auf Arbeiten zur Sprach- bzw. Textgenerierung von [64] auf. Der erste Schritt besteht aus der Auswahl der zur übermittelnden Information (*Inhaltsplanung*) sowie der Anknüpfung zum vorherigen Diskurs (*Diskursplanung*). Dabei wird ein baumartiger Inhaltsplan erstellt, in welchem die Diskuseinheiten (zumeist Propositionen) die Blattknoten darstellen. Die Position der Blattknoten im Baum wird oft durch rhetorische Relationen (wie z.B. Reihung, Elaboration oder Kontrastierung) zwischen den einzelnen Diskuseinheiten festgelegt. Eine besondere Herausforderung ist die Planung von multimodalen Äußerungen mit aufeinander abgestimmten Teilen. Ähnlich wie bei der multimodalen Integration wird für die Verteilung des Inhalts und die Koordination der unterschiedlichen Modalitäten oft eine Grammatik eingesetzt. Dabei wird von einer Sprechaktbasierten Repräsentation ausgegangen, welche auch Diskursfunktionen und emotionale Funktionen beinhaltet kann.

Der nächste Schritt betrifft die Wahl geeigneter Verhaltensformen (z.B. Wörter, Sätze oder Gesten) zur Realisierung multimodaler Äußerungen (*Mikroplanung* und *Realisierung* nach [64]). Dazu müssen in der Sprachgenerierung die Aufgaben der Lexikalisierung, Aggregation und der Generierung von referierenden Ausdrücken gelöst werden. Die Planung non-verbaler Aktionen geschieht meist Lexikon-basiert, wobei kontext-sensitiv aus vorgegebenen Schablonen ausgewählt wird [16]. Weiterführende Ansätze ermöglichen die Erzeugung neuer Aktionen durch Auswahl und Komposition verschiedener Anteile. So lassen sich z.B. ikonische Gesten aus Bestandteilen wie Handform, Trajektorie oder Händigkeit assemblieren, die wiederum aus Repräsentationen von räumlichen Informationen sowie zusätzlichen Kontextinformation (Sprechakttyp, Diskursstatus, vorherige Geste) abgeleitet werden können [9]. Für die Verhaltensumsetzung werden zumeist datenbasierte Modelle wie *Unit Selection* für Text-zu-Sprache-Systeme oder *Motion-Capturing* für non-verbales Verhalten eingesetzt [71]. Flexiblere Systeme setzen auch hier modellbasierte Techniken ein, um größere Teile des Verhaltens autonom und bedarfsgerecht zu generieren [43].

15.3.4 Emotionen

Emotionen übernehmen in der Kommunikation eine wichtige Signalfunktion. Sie geben Aufschluss über den inneren Stimmungszustand eines Agenten, können aber auch gezielt in der sozialen Interaktion eingesetzt werden um z.B. Empathie auszudrücken. Zudem beeinflussen die Emotionen auch die kognitiven Prozesse. So sind sie z.B. wichtig für Bewertung von Wahrnehmungen und Handlungserfolg. In Kognitionstheorien gelten sie sogar als eine Grundvoraussetzung für organisiertes Handeln und die Fähigkeit, zwischen verschiedenen Handlungsoptionen zu entscheiden [22]. Man charakterisiert Emotionen im allgemeinen durch eine positive oder negative Wertigkeit (Valenz) sowie ihre Stärke. Computational Ansätze der Emotionsmodellierung lassen sich in zwei Grundrichtungen einteilen. In kommunikationsgetriebenen Ansätzen werden etwa die Gesichtsausdrücke eines animierten Agenten direkt nach der beabsichtigten Wirkung auf den Benutzer ausgewählt, z.B. [58]. Simulationsbasierte Ansätze bauen auf *Appraisal*-Theorien wie OCC [54] auf. Die OCC-Theorie sieht Emotionen als durch die bewertende Reaktion (*valenced reaction*) auf Ereignisse und Objekte im Licht von Zielen, Standards und Attituden entstehend, nimmt aber an, dass Emotionen vollständig kognitiv und kategorial sind. Andere Arbeiten setzen eine Simulation der Emotionsdynamik in kontinuierlichen Emotionsräumen ein, die durch Impulse von außen und innen beeinflusst wird [7]. Aktuelle Bestandsaufnahmen zu computationalen Ansätzen der Emotionsmodellierung finden sich bei [63] und – speziell das europäische Forschungsnetzwerk HUMAINE (Human-Machine Interaction Network on Emotion) betreffend – bei [67].

15.3.5 Kognitive Architektur

Um ein kohärentes Gesamtverhalten zu erzielen, ist eine Architektur erforderlich, die Wahrnehmungs- und Verhaltensprozesse mit kognitiven Prozessen vereinigt. Zu diesen zählen Aufmerksamkeit, Erinnerungen, Urteile, Vorstellungen, Antizipation, Planen, Entscheiden, Problemlösen und das Mitteilen von Ideen [80]. Für die verkörperte Kommunikation kommen Sprachverständnis und -generierung, Multimodalität, Dialogplanung sowie Prozesse der sozialen Kognition hinzu. Zu den bekanntesten kognitiven Architekturen zählen die auf Produktionssystemen basierenden wie z.B. SOAR [65]. Eine solche Architektur beruht auf Annahmen über die repräsentationalen Charakteristika und die Struktur von Gedächtnissen und den Prozessen, die

auf diesen Gedächtnissen operieren. In Analogie zu Wahrnehmungsvorgängen verläuft der Auswertungsprozess parallel, während Entscheidungs- und Problemlösevorgänge seriell erfolgen. Der Datenspeicher wird als Pendant des menschlichen Arbeitsgedächtnisses gesehen, während der Produktionsspeicher als Form eines prozeduralen Langzeitgedächtnisses dient. Produktionen selbst bestehen aus Regeln der Form, „wenn Bedingung, dann Aktion“. Hybridsysteme wie Act-R [2] vereinen symbolisches Wissen mit subsymbolischen Prozessen, die auf dem Wissen operieren.

Des Weiteren werden oft Belief-Desire-Intention (BDI) Architekturen eingesetzt. Diese haben ihren Ursprung im Bereich der mentalen Modelle [11]. [28] und [24] betonen, dass ein System, welches in dynamischen, unsicheren und unvorhersehbaren Welten agieren soll, eine Repräsentation von *Beliefs*, *Desires* und *Intentions* enthalten muss. Die *Beliefs* repräsentieren deklarativ die Annahmen des Agenten. Die *Desires* bestehen aus den diversen übergeordneten Zielen des Agenten. Wird ein Ziel mit einer konkreten Handlungsabsicht verknüpft, d.h. mit einem Plan zur Erreichung des Ziels, so spricht man von einer *Intention*. Für den BDI-Ansatz existieren logische Semantiken [61, 78] sowie eine Vielzahl von Implementierungen wie PRS [27], JAM [36], Jack [35] und Jadex [59] mit einem breiten Spektrum an erfolgreichen Anwendungen in verschiedenen Domänen, u.a. auch der konversationellen Agenten [73].

15.4 Verkörperte Kommunikation mit einem virtuellen Agent am Beispiel Max

Nachdem verschiedene Methoden zur Realisierung eines verkörperten kommunizierenden Agenten im Überblick vorgestellt wurden, soll nun eine konkrete Realisierung am Beispiel des virtuellen humanoiden Agenten „Max“ verdeutlicht werden. Max wird seit 1999 an der Universität Bielefeld entwickelt und in diversen Interaktionsszenarien eingesetzt. Im Folgenden wird die Realisierung einer Kooperation zwischen Agent und menschlichem Partner im Rahmen einer gemeinsamen Montageaufgabe beschrieben, die auch vielfältige Prozesse der verkörperten Kommunikation einschließt.

15.4.1 Szenario

Der hier beschriebene Agent Max [42] kooperiert mit einem menschlichen Interaktionspartner, um den Zusammenbau eines Flugzeugmodells aus Teilen eines Konstruktionsbaukastens zu lösen. Dabei verhandeln die beiden Dialogpartner über einzelne Konstruktionsschritte und übernehmen Rollen als Instrukteur und Konstrukteur. Beide Rollen können sowohl vom Menschen als auch von Max eingenommen werden. Mensch und Max stehen sich in einer Virtual-Reality-Installation an einem virtuellen Tisch mit verschiedenen virtuellen Bausteinen gegenüber, wie in Abb. 15.3 gezeigt. Der Mensch sieht Max und die gesamte Szene mit einer Stereobrille dreidimensional und hört die synthetische Stimme von Max räumlich aus versteckten Lautsprechern. Max „sieht“ den Menschen, dessen Blickrichtung, Hand- und Armbewegungen mit Hilfe eines Infrarot-Trackingsystems und kabelloser Datenhandschuhe, und er „hört“ über Funkmikrofon dessen Sprache, die er mit einem Spracherkennung verarbeitet. Sowohl Mensch als auch Max können durch natürlichsprachliche Instruktionen und Gesten den Zusammenbau einzelner Modellbauteile veranlassen, der in physikgerechter Simulation ausgeführt wird.

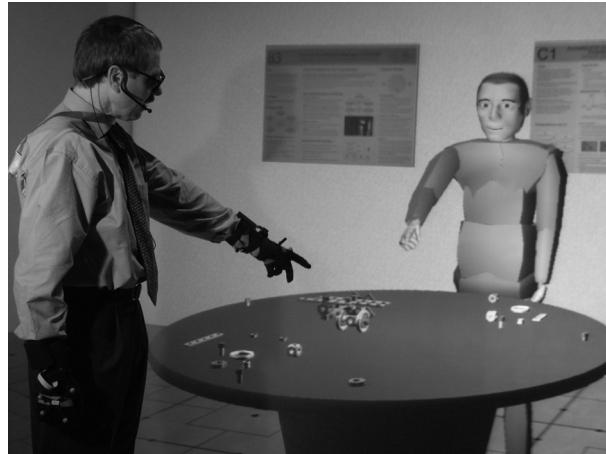


Abbildung 15.3: Face-to-Face-Dialogsituation im Kooperationsszenario mit Max

Abbildung 15.4 zeigt den Ablauf einer Beispielsinteraktion, in welcher Max zusammen mit seinem Partner einen Propeller aus Modellbauteilen baut. Die Bilder illustrieren das gezeigte Verhalten sowie die Bauteile in verschiedenen Momentaufnahmen der Interaktion (nach [48, S. 298]).

15.4.2 Kognitive Architektur: Beispiel

Die Grundlage für die Realisierung des künstlichen Kommunikationspartners Max ist die in Bielefeld entwickelte CASEC-Architektur (Cognitive Architecture for Situated Embodied Communicators). Diese verbindet symbolverarbeitende und verhaltensbasierte Ansätze in einer hybriden Systemarchitektur. Für die Modellierung des mentalen Zustands des Agenten baut CASEC auf einer BDI-Implementierung [36] auf, welche die Grundlage für die Modellierung des Zusammenspiels von Zielen, Wissen und Ereignissen bietet. Um reaktive Verarbeitungsprozesse zu berücksichtigen, wurden weitere Ebenen einer kognitiven Architektur konzipiert und implementiert: An Stelle der starren Menge der Beliefs tritt ein dynamisches Arbeitsgedächtnis mit Aktivierungsfunktionen, welches in die BDI-Struktur eingebettet ist und auf Modelle der Psychologie zurückgreift [6, 53]. Subsymbolische Aktivierungsfunktionen werden auf die Ziel- und Intentionsmodellierung übertragen und ihre Einflüsse auf den kognitiven Entscheidungsprozess berücksichtigt. CASEC baut auf den folgenden Prinzipien auf:

- **Nebenläufige** Realisierung von Wahrnehmungs-, Schlussfolgerungs- und Handlungskomponente (*Perceive, Reason, Act*) durch **modulare** Struktur
- Ausführung reaktiver und deliberativer Verhaltensweisen mittels **Behaviors** mit individuellen Prioritätswerten, **kontextsensitive Zielsteuerung** durch den Einsatz sowohl von **zielbasierten** als auch von **reaktiven Plänen**
- Ständiger Fluss von Informationen und Aktivierungen innerhalb einer zentralen Verarbeitungsschleife durch ein **dynamisches Arbeitsgedächtnis**
- Modellierung **expliziter mentaler Zustände** durch ein **erweitertes BDI-Modul** (Obligationen, gemeinsame Pläne, *Constraint*-basierte Repräsentation)



Abbildung 15.4: Beispiel-Interaktion aus dem kooperativen Montage-Szenario

Um die Prinzipien umzusetzen, besitzt die Architektur folgenden strukturellen Aufbau (Abbildung 15.5). Die Sektoren *Perceive* und *Act* repräsentieren die Physis des Agenten, durch die er mit der Umwelt wechselwirkt. Sie bietet multimodale Ausdrucksmöglichkeiten (Sprache, Gestik, Mimik) und verankert seine sensorische Wahrnehmung und ausführende Aktorik in der Umwelt. Der direkte Informationsfluss zwischen den beiden Sektoren ermöglicht schnelles, reaktives Verhalten. Ein *Reason*-Sektor ermöglicht deliberatives, planbasiertes Verhalten durch ein enges Zusammenspiel des klassischen *Perceive-Reason-Act*-Zyklus. Die Wahrnehmungs-, Schlussfolgerungs- und Handlungskomponenten sind nebenläufig realisiert, so dass die deliberative und reaktive Verarbeitung parallel erfolgen. Ein *Mediator* im *Act*-Sektor übernimmt die Aufgabe, reaktive und deliberative Verhaltenstränge zusammenzuführen und zieht in Betracht, welche Modalitäten gerade frei bzw. im Einsatz sind. Im Konfliktfall besteht die Entscheidungsgrundlage aus Prioritätswerten, die die Dringlichkeit und Angemessenheit eines Verhaltens in einer vorliegenden Situation repräsentieren. Reaktive Verhaltensweisen realisieren bspw. unmittelbare Systemreaktionen mit hoher Priorität, aber auch untergeordnete *Secondary Behaviors* wie Lidschlag und Atmen.

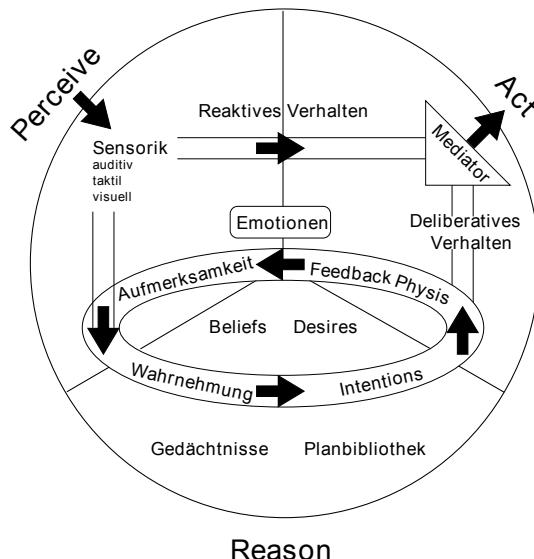


Abbildung 15.5: Struktur der CASEC-Architektur.

Die interne kognitive Verarbeitung des Agenten geschieht in einer „deliberativen Schleife“. Wahrnehmung besteht damit nicht aus der reinen Erfassung sensorischer Daten, sondern aus einer aktiven, situationsabhängigen Filterung und Verarbeitung perzipierter Sensoreindrücke. Kognitive Prozesse werden so nicht als abgelöster interner Vorgang betrachtet, sondern an die Physis gekoppelt, mit einer stärkeren Betonung der prozeduralen Komponente.

Der Kern des deliberativen Moduls folgt dem BDI-Ansatz. Als verhaltensauslösender Antrieb dienen explizit repräsentierte Ziele (*Desires*), die sowohl intern als auch von außen aufgeworfen werden können. Durch eine *Constraint*-basierte Repräsentation können abstrakte Ziele, Teilentscheidungen sowie eingegangene *Commitments* dargestellt werden. Die Intentionsbildung in der kognitiven Schleife wird aufgrund der vorliegenden Beliefs, der aktuellen Ziele sowie alternativer Handlungsmöglichkeiten bestimmt. Für die erforderliche Modellierung des Commitment-Verhaltens im Rahmen einer Kooperation ist die BDI-Struktur so erweitert, dass Obligationen und gemeinsame Pläne unterstützt werden.

Handlungsoptionen liegen in Form von Plänen vor, die durch Vorbedingungen, Kontextbedingungen, erreichbare Konsequenzen und eine Prioritätsfunktion beschrieben werden. Die Planbibliothek besteht aus simplen Plankonstrukten, die einfache Aktionen direkt in entsprechende *Behaviors* umsetzen können. Bei Bedarf werden darüber hinaus eigenständige Planer angestoßen, die einen komplexeren Plan ausarbeiten. Planselektion findet dabei einerseits auf der Ebene der kognitiven Intentionsbildung statt, andererseits prioritätsbasiert durch den Mediator auf der Ebene der direkten Aktionsausführung. Sowohl die aktiv ausgeführten Intentionen als auch die aktuell anliegenden und möglicherweise konkurrierenden Behaviors werden bei den zurückfließenden Feedbackinformationen berücksichtigt. Die so entstehende Schleife verdeutlicht eine der Kernideen der Architektur, nämlich dass ein ständiger Strom von Informationen zwischen den Sektoren umläuft, der sowohl aktuelle Sensor- und Aktorinformationen als auch interne Zustände einbezieht.

15.4.3 Interaktionssteuerung

Während die reaktiven Verhaltensweisen für die Generierung von Engagementsignalen und Feedbackmechanismen Einsatz finden, spielt die deliberative Komponente eine zentrale Rolle für die Interaktionssteuerung. Der gemeinsame Montageprozess zwischen Max und Mensch vereint Kommunizieren und Handeln zu einer zielgerichteten Interaktion, die auf Basis der Sprechakttheorie bzw. der Theorie kommunikativer Akte als Abfolge intentionaler Akte verstanden werden kann. Das Interaktionsmanagement bei Max geschieht daher planbasiert; kommunikative wie manipulative Akte werden als Aktion-Plan-Operatoren dargestellt und in der Handlungsplanung der BDI-Komponente verarbeitet.

Genauer verfügt der Agent über prozedurales Wissen in Form aufgabenorientierter Pläne, welche die als nächstes auszuführenden Konstruktionsschritte repräsentieren und hierarchisch geschachtelt werden können. So besitzt er beispielsweise einen Skelettplan, der den generellen Ablauf der kooperativen Montage eines bestimmten Aggregats vorgibt und die Koordination der daraus abgeleiteten Handlungsoptionen übernimmt. Zunächst initiiert dieser Plan einen Montageplaner, der die auszuführenden Konstruktionsschritte aus dem Langzeit-Montagewissen des Agenten ableitet, welche dann als Unterziele sukzessive aufgeworfen und situationsabhängig in der Interaktion abgearbeitet werden.

Die augenblickliche Situation sowie den Verlauf der Interaktion mit dem menschlichen Partner erfasst Max deklarativ in Form von Beliefs. Aus Sicht des Agenten stellt sich dieser Verlauf als Folge von Situationen dar, in denen jeweils eine bestimmte (kommunikative oder manipulative) Aktion zu neuen Objektkonstellationen in der Szene sowie zu veränderten mentalen Zuständen der Interaktionspartner führt. Der Agent muss diese Entwicklungen erfassen, um beispielsweise Fragen wie „Welche Schraube hast du gemeint?“ verstehen und beantworten zu können. Ein Gedächtnis erfasst dazu den Dialogverlauf in den folgenden Aspekten:

- die *Ziele*, die in der Interaktion verfolgt werden; oftmals gibt es mehrere Ziele, die sich überlagern und in bestimmten Beziehungen zueinander stehen
- den propositionalen *Inhalt*, der im Diskurs ausgetauscht wird
- die *Obligationen* (Verpflichtungen), die für die Kommunikationspartner im Verlauf des Dialogs entstehen, z.B. auf eine Frage zu antworten
- die Übernahme der *Initiative*, d.h. die Kontrollausübung auf den Dialog durch das Aufwerfen neuer Ziele
- den Besitz der Sprecherrolle (*Turn*)

Diese Aspekte und ihr Einfluss auf das Interaktionsverhalten des Agenten werden in der kognitiven Architektur wie folgt modelliert: Spezielle Pläne übernehmen die Aushandlung des Rederechts, Ziele werden mit dem jeweiligen Initiator versehen und fließen in die Intentionsbildung ein, und der Inhalt eines Dialogaktes oder einer Montageaktion führt zu veränderten Beliefs. Obligationen stellen eine besondere Unterart von Zielen dar, die als *Social Attitudes* zusätzlich in den BDI-Ansatz eingeführt wurden [73]. Sie betreffen Verpflichtungen, die ein kooperativer Dialogpartner entsprechend gewisser Normen eingeht, die aber mit seinen eigentlichen Zielen in Widerspruch stehen können.

Um dem Verlauf des Montageprozesses folgen zu können und die dabei neu entstandenen Aggregate auch sprachlich referenzierbar zu machen, verfügt der Agent über detailliertes Langzeitwissen über die Modellbauteile sowie die Zieldomäne (z.B. Flugzeug-Bauen). Zum einen

benötigt Max dieses Wissen für seinen Montageplaner, der die Konstruktionsschritte bestimmt, die für den Zusammenbau des vom Menschen nachgefragten Montageziels auszuführen sind. Zum anderen nimmt Max fortwährend das Montagegeschehen in der Szene wahr und aktualisiert schritthalbend eine Beschreibung, die Informationen über die Objekte und deren eingegangene Verbindungen enthält. Das Konzeptwissen über Aggregate und die Rollen einzelner Komponenten ist über assoziative Verknüpfungen an die Deliberation angebunden. Wenn der Agent prüft, welches Objekt sich für den Bau eines Aggregats eignet, wird relevantes Langzeitwissen aktiv und so für weitere Inferenzen verfügbar.

Um die Handlungsplanung dynamisch an die Konstruktion des entstehenden Aggregats anpassen zu können, werden Pläne möglichst allgemein formuliert. Wenn Max einen Plan für den Bau eines Propellers entwirft, so gibt dieser den allgemeinen Typ der dafür zu verwendenden Bauteile an, legt sich aber nicht auf spezielle Typen oder gar konkrete Objektinstanzen fest. Dadurch ist der Agent in der Lage, sich auf unterspezifizierte Absichten seines Konstruktionspartners und deren Konkretisierung in der laufenden Interaktion einzustellen. Max verwendet eine Constraint-basierte Repräsentation, die es erlaubt, möglichst viele Optionen offen zu lassen, und die gleichzeitig die geltenden und ausgehandelten Einschränkungen explizit macht. Die Constraints gelten dabei innerhalb des gesamten Handlungskontextes, werden an Unterpläne vererbt und können auch versprachlicht werden. In Abbildung 15.8, die den Ablauf einer multimodalen Äußerungsplanung zeigt, sind Constraints bei der Satzplanung dargestellt.

15.4.4 Sprach- und Gestenverarbeitung

Im Rahmen der multimodalen Verhaltensverarbeitung perzipiert Max seinen menschlichen Partner in der realen Welt (Bewegung, Blickrichtung, Gestik) über Sensor-Marker, Datenhandschuhe, einen Eye-Tracker und durch ein Mikrofon, das Daten an seine Sprachverarbeitungskomponente liefert. Die virtuelle Umwelt perzipiert er über zwei visuelle Sensoren, die jeweils einen Blickwinkel auf die Szene definieren und sich mit den Augen des Agenten bewegen.

Spracheingaben des menschlichen Interaktionspartners analysiert Max in mehreren Verarbeitungsschritten, beginnend mit dem Registrieren, dass etwas gesagt wurde, und der Verarbeitung des Signals durch einen Spracherkennender [25]. Die Daten des Spracherkennenders werden dann weiter verarbeitet, um die syntaktischen Kategorien Subjekt, Prädikat und Objekt zuzuordnen (*Part-of-Speech-Tagging*). Es folgen die semantische Analyse und die Auflösung von Referenzen. Parallel dazu wird die Äußerung nach Hinweisen auf die Intention des Sprechers und das Performativ (*inform, query, request, propose*) des Sprechers ausgewertet. Die gewonnenen Daten füllen die Felder einer Frame-artigen Wissensstruktur, die in Form eines Dialogakts sämtliche für die deliberative Verarbeitung relevanten Aspekte beinhaltet: illokutionärer Akt, perlokutionärer Akt, propositionaler Inhalt, Turn-Taking-Funktion, Bezug zu vorherigen Äußerungen sowie den Adressaten.

Für die Referenzauflösung wird ein *Constraint Satisfaction*-Verfahren eingesetzt [55]. Die Referenzauflösung wie auch die Intentionserkennung erfordern den Einbezug von Kontextwissen über den mentalen Zustand des Agenten, den Partner, die aktuelle Szene, sowie über die Diskurshistorie. Aus diesem Grund ordnet Max mit Hilfe spezieller Pläne jede Äußerung des Menschen in einen relevanten Diskurskontext ein. Dabei wird überprüft, ob sich die Äußerung direkt auf aktuelle Vorhaben oder Ziele des Agenten bezieht oder auf vorherige Dialogbeiträge, die in der Diskurshistorie gespeichert sind. Dies geschieht durch die Berechnung einer Korrelation

zwischen dem Inhalt des Dialogaktes und der aktuellen Intention des Agenten. Wenn der Agent einen passenden Kontext findet, so wirft er als Unterziel die Obligation auf, die Äußerung in diesem Kontext zu bearbeiten. Durch diese Assoziation eines Dialogaktes mit einem Ziel-Kontext ist Max in der Lage, mit mehreren Handlungssträngen gleichzeitig umzugehen.

Um multimodale Eingaben zu verarbeiten, durchsuchen spezielle Detektoren fortwährend die Sensordaten aus Bewegungstracking und Datenhandschuhen nach Mustern, die auf eine Geste hindeuten (aufbauend auf [45]). Entsprechend der Merkmalsstruktur einzelner Gesten sind die Detektoren als hierarchische Netze organisiert. Erkannte Gesten werden dann im Kontext der zeitnahen verbalen Äußerung sowie der aktuellen virtuellen Szene ausgewertet und multimodal integriert. Diese Integration geschieht nach temporalen und semantischen Aspekten und wird von erweiterten Zustandsübergangautomaten durchgeführt.

15.4.5 Turn-Taking

Als zentrale Form des Dialogmanagements besitzt Max Turn-Taking-Kompetenzen. Max erkennt und interpretiert Turn-Taking-Signale des menschlichen Interlokutors. Dazu wird das Sprachsignal an einen *is-speaking*-Detektor geleitet. Dieser reagiert erst ab Äußerungen einer gewissen Länge, sodass Feedbacksignale wie „*hmmm*“ und „*okay*“ nicht als Sprechakt im Sinne eines Turns gewertet werden. Zusätzlich zur Sprachverarbeitung werden Detektoren für Gestenerkennung und multimodale Integration eingesetzt. Die Detektoren werden in Abhängigkeit vom konversationalen Zustand aktiviert und deaktiviert, beziehen den aktuellen konversationalen Zustand in ihre Berechnungen mit ein und operieren kontextsensitiv. Die erkannten Gesten, Blickrichtungen und sprachlichen Signale werden auf konversationale Funktionen abgebildet. Innerhalb der kognitiven Komponente spielen die konkreten Ausprägungen der Signale keine Rolle, die Signale können aber die Dringlichkeit ihrer Funktion aufzeigen.

Es sind drei Zeitpunkte im Verlaufe der Interaktion vorgesehen, zu denen der Agent entscheiden kann, auf eine Unterbrechung einzugehen. Abbildung 15.6 zeigt die relevanten Zustände beim Aushandeln des Turns in zeitlicher Abfolge sowie die aktiven Pläne des Agenten. Des Weiteren werden der Wechsel des konversationalen Zustands und der konversationalen Rollen sowie die Sprechhandlungen der Agenten angezeigt. Zunächst wird ein wahrgenommenes Turn-Taking-Signal von einem Plan verarbeitet, der Verhaltensweisen (z.B. Blicke und Gesten) anstößt, die dem Gegenüber vermitteln, dass der Agent den Interlokutor wahrgenommen hat. Um dem regelbasierten Charakter der Turn-Taking-Mechanismen gerecht zu werden, werden für Max *Conclude*-Pläne eingesetzt, welche auf die Signale *Taking-Turn*, *Wanting-Turn*, *Giving-Turn* oder *Yielding-Turn* „lauern“. Sie sind jeweils durch Vorbedingungen so eingeschränkt, dass ein kontext-sensitives Verhalten in Abhängigkeit vom konversationalen Zustand und der konversationalen Funktion des wahrgenommenen Signals in Form einer Obligation generiert wird. Der erste Entscheidungszeitpunkt ist unabhängig vom Inhalt der Äußerung des Gegenübers. Die Entscheidung, sich zu diesem frühen Zeitpunkt unterbrechen zu lassen, ist damit allein abhängig von der Dominanzbeziehung zwischen den Konversationspartnern und den Dringlichkeiten, mit welchen die Interlokutoren jeweils ihr Rederecht verfolgen. So sieht man in der Beispieldiskussion (Abbildung 15.6), wie der Interlokutor Max mit einer Zwischenfrage unterbricht. Dies führt zur Detektion der konversationalen Funktion *TakingTurn* und der Verarbeitung durch den entsprechenden *Conclude*-Plan. Max entscheidet sich seinen Turn zu unterbrechen und macht dies durch eine *GivingTurn*-Handlung deutlich („*Ja?*“). Nachdem der Sprechakt des Interlokutors interpretiert wurde, besteht erneut die Möglichkeit, dass Max sich unterbrechen lässt.

cRole	Speaker	Listener	Speaker	
cState	MyTurn	Overlap	Gap	MyTurn

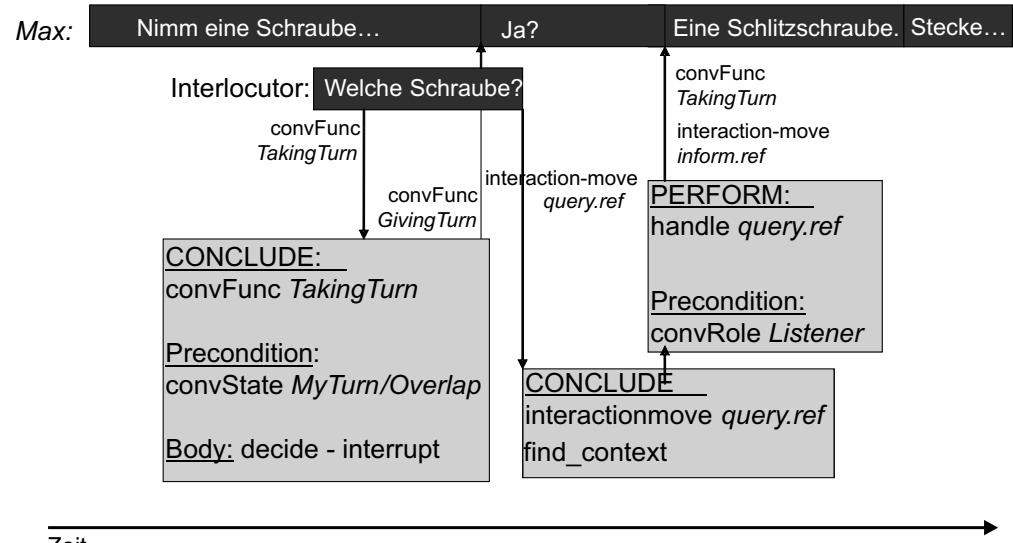


Abbildung 15.6: Turn-Verarbeitung: Zeitlicher Ablauf

Zu diesem zweiten Entscheidungspunkt beruht die Entscheidung allein auf dem bis dahin erkannten Performativ des Dialogakts; handelt es sich um eine Frage (*query*) oder eine direkte Anweisung (*request*), so sollte sich der Agent direkt seinem Konversationspartner zuwenden. Im Falle der Performative *inform* und *propose* fährt Max zunächst mit seiner Äußerung fort. In Abhängigkeit der Relevanz des registrierten Sprechakts lässt er sich aber noch zu einem dritten Zeitpunkt unterbrechen, nachdem die Äußerung in den intentionalen Kontext des Agenten eingeordnet wurde. Besteht ein offensichtlicher Konflikt mit bestehenden Plänen und Intentionen, so kann der Agent direkt darauf einzugehen. Die letzte Möglichkeit des Agenten, auf eine Unterbrechung einzugehen, besteht nach Beendigung seines Turns. In der Beispielinteraktion (Abbildung 15.6) wurde der Inhalt der Zwischenfrage erkannt und einem passenden Kontext zugeordnet. Max geht auf die Frage ein, indem er seinen Interlokalutor darüber aufklärt, dass eine Schlitzschaube gemeint war. Danach führt Max seinen ursprünglichen Turn fort.

Möchte der Agent selbst eine Äußerung machen, so muss er sich an konversationale Regeln halten und die Rolle des Sprechers mit seinem Interaktionspartner aushandeln. Während der Interaktion modelliert Max intern explizit seine Sichtweise des konversationalen Zustands (*MyTurn*, *OthersTurn*, *Gap* oder *Overlap*) sowie der Rollenverteilung der Interlokatoren (*uninvolved*, *speaker* oder *listener*). Die Rolle des Konversationspartners wird aufgrund des overten Verhaltens beurteilt. Für sich selbst setzt Max nur dann die Rolle „Sprecher“, wenn er eine längere Äußerung vornehmen möchte; so kann er auch kurzes konversationales Feedback geben, ohne sich in der Rolle des Sprechers zu sehen. Um die Rolle des Sprechers anzunehmen, muss Max zuvor den konversationalen Zustand *MyTurn* herbeiführen. Besteht der konversationale Zustand aus *OthersTurn*, *Gap* oder *Overlap*, so existieren verschiedene Handlungspläne, wel-

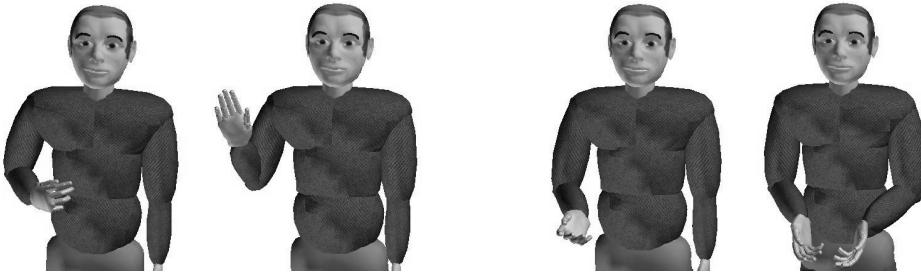


Abbildung 15.7: Gesten für Wanting-Turn, Taking-Turn, Giving-Turn (Stufe 1), Giving-Turn (Stufe 2).

che durch Vor- bzw. Kontextbedingungen kontext-sensitiv modelliert sind und beispielsweise die Produktion von Gesten anstoßen, deren Ausdrucksstärke iterativ gesteigert wird (siehe Abbildung 15.7), bis der Agent den Turn bekommen hat oder die Pläne als gescheitert gelten. Möchte Max gezielt sein Gegenüber unterbrechen, so findet ein gesonderter Plan Einsatz, welcher unabhängig von der Rollenverteilung und dem konversationalen Zustand ausgeführt wird.

15.4.6 Multimodale Verhaltensgenerierung

Wirft die Deliberation des Agenten eine Handlungs- bzw. Kommunikationsintention auf, übernehmen spezielle Generator-Pläne die Ableitung von geeigneten Aktionssequenzen sowie deren Ausdifferenzierung in entweder multimodale Äußerungen oder manipulative Aktionen. Der Ablauf der Äußerungsplanung ist in Abbildung 15.8 dargestellt und startet von einem kommunikativen Ziel, das sich direkt aus der aktuellen Intention des Agenten ergibt (Inhaltsplanung). Die folgenden Schritte werden von einem speziellen Planer vorgenommen, der sich an systemischen Grammatiken [33] orientiert. Der Planer wählt den passenden Dialogaktyp aus und expandiert einzelne Konzepte schrittweise gemäß ihrer semantischen Rolle. Er operiert auf einer Repräsentation, die sich aus einem Template-Knoten für die Aktion (das Performativ) sowie weiteren Knoten (*MessageNodes*) für die involvierten Konzepte zusammensetzt, aus denen die Konstituenten des Satzes (z.B. Subjekt, Prädikat, Objekt, Attribut) generiert werden können.

In der Satzplanung werden die *MessageNodes* mit den abstrakten Konzepten gefüllt, die sie repräsentieren sollen und die durch Merkmalswissen in Form von Attribut-Wert-Listen angereichert sind. Diese Knoten enthalten Realisierungsanweisungen, die einzelne Generierungsentscheidungen repräsentieren und die syntaktische Struktur der Äußerung Schritt für Schritt einschränken. Solche Anweisungen können sich beispielsweise auf das Einfügen von Konstituenten (+Verb, +Subjekt), deren Reihenfolge (*Subjekt > Verb*) oder die Realisierung von grammatischen Beugungen beziehen. Dadurch können Subjekt-Verb-Kongruenz oder Attribut-Nomen-Kongruenz realisiert werden, auch kann der Typ eines Objektes die Auswahl des lexikalischen Eintrags des mit dem Objekt assoziierten Verbs beeinflussen. Während der Satzplanung baut der Agent multimodale, referierende Ausdrücke aus sprachlichen und gestischen Anteilen auf – insbesondere deiktischen Gesten, mit denen direkt auf Objekte in der Umwelt verwiesen werden kann, sowie ikonischen Gesten, die räumlich-visuelle Informationen darstellen können. Max kann Gesten explizit durch die Formeigenschaften der bedeutungstragenden Phase (wie z.B. Handform oder Bewegung) oder indirekt durch die Angabe einer kommunikativen

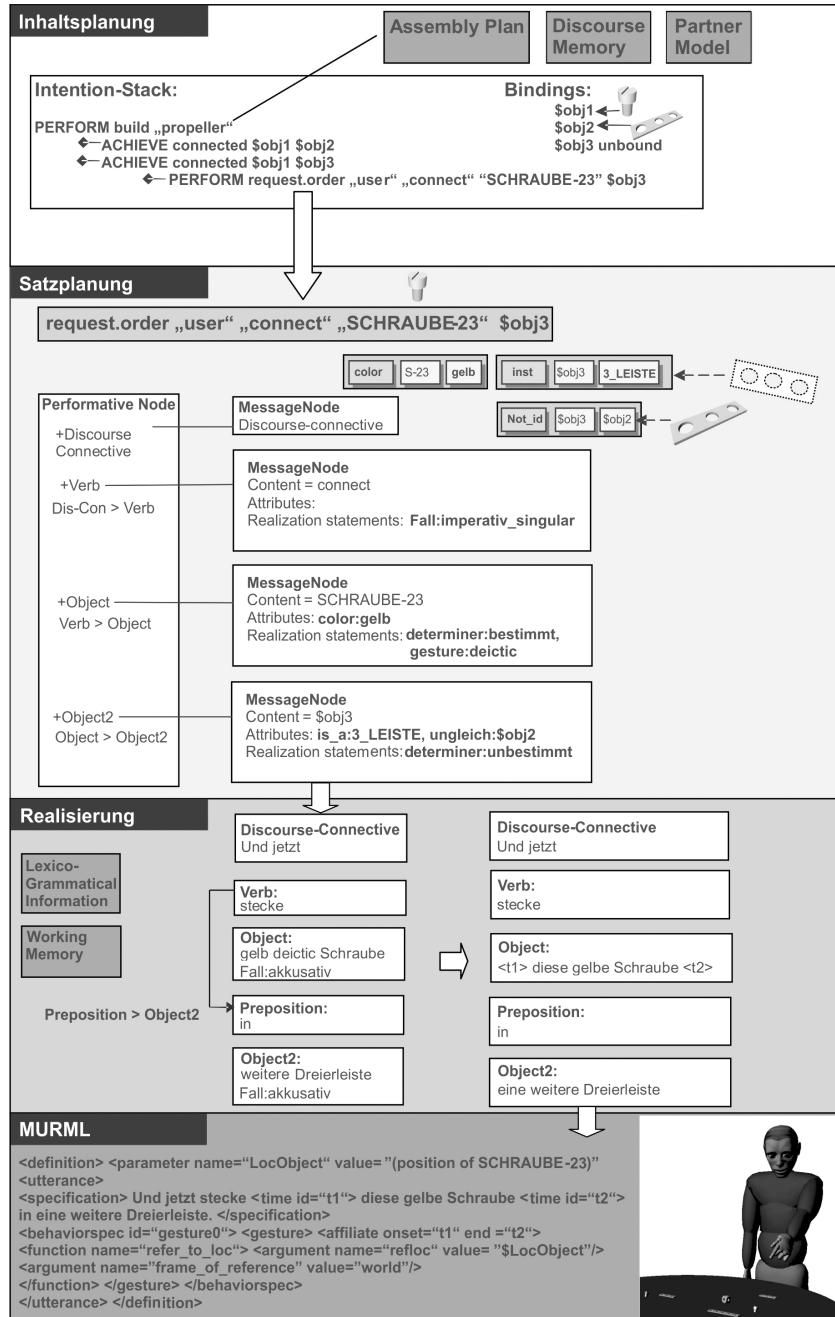


Abbildung 15.8: Beispielablauf einer multimodalen Äußerungsplanung.

Funktionen spezifizieren [43]. In Abbildung 15.8 wird im MessageNode der definit bestimmten Schraube (SCHRAUBE-23) eine deiktische Geste vorgesehen. Als letzten Unterschritt der Satzplanung leitet Max die Funktion der Äußerung im Diskurs sowie das Diskurssegment, zu dem sie beiträgt, aus seiner aktuellen Intentionen ab. Ist die Äußerung Teil einer bereits vorgeplanten längeren Sequenz, so fügt er passende Konnektoren ein, wie zum Beispiel „*als erstes*“ zu Beginn einer Sequenz, „*und dann*“, „*als nächstes*“ für Zwischenschritte und „*als letztes*“ für den Abschluss der Sequenz.

In der Realisierungsphase wird für jeden Knoten ein sprachlicher Ausdruck aus dem Lexikon ausgewählt und entsprechend den jeweiligen grammatischen Informationen angepasst. Für multimodale Äußerungen wird dabei sichergestellt, dass die Gesten synchron zu den zugehörigen verbalen Elementen geplant werden. So geplante Äußerungen werden in MURML, einer Auszeichnungssprache auf XML-Basis, formuliert und dann in synthetische Sprache und simultane Gesten- bzw. Gesichtsanimationen umgesetzt [43]. Um sprachbegleitend zu gestikulieren, erzeugt Max seine Hand- und Armbewegungen jedes Mal neu und speziell an den Kontext angepasst. Die Gesteplanung bearbeitet dazu zunächst Fragen des Timings, der Auswahl der Gliedmaße und der Einpassung in den räumlichen Kontext (z.B. die Zielposition bei Zeigegesten). Die Motorplanung erstellt dann in Realzeit direkt ausführbare Animationen.

Komplexe Äußerungen entstehen schrittweise aus mehreren sogenannten *Chunks*, einzelne Pakete aus jeweils einer kurzen Intonationsphrase und begleitenden non-verbalen Aktionen. Solche Pakete werden nacheinander kontinuierlich verbunden und mit Phasen wechselseitiger Ablösung produziert. Diese Phasen sind notwendig, um die Synchronität z.B. zwischen Sprache und Geste sicherzustellen, für die sich beide Modalitäten an die jeweils andere anpassen müssen. Innerhalb eines Pakets adaptiert sich die Geste in ihrem Bewegungsablauf an die Anfangszeit und Dauer der zugehörigen Wörter, kann also erst geplant werden, wenn Zeitinformationen aus der Sprachsynthese vorhanden sind. Ist der Körper in der Simulation nicht in der Lage, rechtzeitig zu einer Gestenausführung zu gelangen, wird die Aussprache der zugehörigen Intonationsphrase hinausgezögert. Die kognitive Komponente wird dabei durch ständiges Feedback über den Planungs- und Ausführungsstand einzelner Pakete und ihrer sprachlichen wie gestischen Anteile informiert. Die Flexibilität seines Sprachsynthesystems erlaubt es Max, Betonung an jeder gewünschten Stelle sinnvoll zu erzeugen und mit dem Gesteplanungsprozess abzustimmen. Auf diese Weise kann Max im richtigen und natürlich wirkenden Miteinander sprechen und gestikulieren.

15.4.7 Physis, Emotionen, Bewegungsgenerierung

Um natürliche Ausdrucksmöglichkeiten samt multimodalen Ausdrücken, sprachlichen Äußerungen, emotionalen Reaktionen und situationsangepasstem Blickverhalten zu beherrschen, verfügt Max über eine anthropomorphe Erscheinung. Die äußerliche Physis von Max besteht aus einer mehrteiligen Hülle, die sowohl seinen Körper als auch sein Gesicht umfasst. Der Körper wird durch ein bewegliches kinematisches Skelett gesteuert, das insgesamt 68 Segmente und 103 Freiheitsgrade in 57 Gelenken enthält, davon allein 25 Freiheitsgrade (16 Gelenke) in den Fingern jeder Hand.

Das Gesicht von Max besteht aus einer verformbaren „Haut“ mit simulierten Muskeleffekten. Zur Darstellung der Sprechmimik wird der zu sprechende Satz als Abfolge von Lauten durch eine Liste aus Phonemen modelliert; für deutsche Sprache werden 39 Phoneme einbezogen, die

auf 11 sog. „Viseme“ (visuelle Phoneme) abgebildet werden. Viseme beschreiben die Gesichtsstellung (Mund, Lippen etc.) bei der Artikulation eines oder mehrerer Phoneme und werden ineinander überführt. Mit dem Beginn eines jeden Lautes beginnt zeitgleich die Animation des Visems, so dass der Mund synchron zum Sprechen bewegt wird.

Für das Engagementverhalten zeigt Max zusätzlich ein leicht verständliches Feedback seines inneren Zustandes in Form expressiver Gesichtsausdrücke; z.B. kann Max verständnislos oder nachdenklich schauen. Zudem können mimische Elemente auch das von Max Gesagte unterstreichen (freundliches Gesicht bei dem Angebot von Hilfe). Beide Arten der Gesichtsbewegung lassen sich kombinieren. Für die Berechnung des emotionalen Zustands des Agenten wird im Gegensatz zu den kommunikationsgetriebenen Ansätzen eine kontinuierliche Simulation der Emotionsdynamik [7] eingesetzt. Sowohl Impulse von außen (durch seine Perzeption) als auch von innen (ein Ziel wird erreicht oder verfehlt) wirken auf den momentanen emotionalen Zustand von Max. Da sich aufeinander folgende Impulse aufsummieren, können auch starke Emotionen entstehen; bei Ausbleiben von Impulsen strebt das System zurück in den neutralen Zustand.

Eine weitere Form des Engagement wird durch die Modellierung interaktiven Blickverhaltens umgesetzt, welches den aktuellen Aufmerksamkeitsfokus des Agenten verdeutlicht und bis hin zur Herstellung gemeinsamer Aufmerksamkeit reicht [56].

15.5 Zusammenfassung und Ausblick

Neben einem Überblick über grundlegende Fragestellungen und aktuelle technische Ansätze wurde am Beispiel des künstlichen Agenten „Max“ ein konkretes System beleuchtet, das „verkörperte Kommunikation“ mit Computersimulationen in virtueller Realität realisiert.

An diesem Beispiel wurde gezeigt, wie virtuelle Agenten mit synthetischer Stimme und einem animierten Körper sprechen, gestikulieren und Gesichtsausdrücke zeigen können. Analog können sie Sprache, Gestik und Blickrichtung des Menschen als Eingaben verarbeiten. Doch die Entwicklung verkörperter Agenten, die tatsächlich mit menschlichen Partnern interagieren sollen, verdeutlicht, dass verkörperte Kommunikation mehr als multimodale Kommunikation ist. Sie verlangt die enge Verbindung von Wahrnehmen und Handeln in der Interaktion auf zwei Ebenen. Auf der Kommunikationsebene müssen verbale und non-verbale Signale im Sinne einer kohärenten multimedialen Konversation zusammenwirken. Entsprechende Ansätze zur Verhaltensverarbeitung, Verhaltensgenerierung und Dialogmodellierung wurden vorgestellt. Gleichzeitig müssen diese auf der Architekturebene durch eine kognitiv plausible Verbindung von reaktivem und deliberativem Verhalten umgesetzt werden. Am Beispiel Max wurde konkret gezeigt, wie dazu kognitive Architektur, Emotionsmodellierung, Sensorik und Physis zusammenspielen.

Die Erforschung natürlicher verkörperter Kommunikation birgt noch viele Herausforderungen für die Entwicklung intelligenter interaktiver Systeme. Die wohl größte besteht in der Umsetzung der vielschichtigen Dynamik menschlicher verkörperter Kommunikation. Ein wichtiger Baustein dafür ist die Fähigkeit der inkrementellen und parallelen Verarbeitung, sowohl in der Wahrnehmung und Interpretation der Signale des menschlichen Partners als auch der Generierung und Anpassung des Agentenverhaltens. Jüngere Arbeiten verfolgen dazu Ansätze der kontinuierlichen Verhaltensadaptation (z.B. in der Sprach- und Gestengenerierung [14]), mit der

künstliche Kommunikationspartner bereits während einer eigenen Äußerung auf das Feedback des Adressaten reagieren können. Mit der verkörperten Kommunikation geraten dabei kognitive Mechanismen und Architekturprinzipien in den Blick, die auch für die sprachbasierten Systeme der Zukunft prägend sein können.

Literaturverzeichnis

- [1] Allwood, J. (1976). Linguistic communication in action and co-operation: A study in pragmatics. In *Gothenberg Monographs in Linguistics*, volume 2, pages 637–663. Department of Linguistics, University of Gothenberg.
- [2] Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., und Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, 111(4):1036–1060.
- [3] Argyle, M. (1988). *Bodily Communication*. Methuen & Co., New York, 2nd edition.
- [4] Argyle, M. und Cook, M. (1976). *Gaze and mutual gaze*. Cambridge University Press, Cambridge, UK.
- [5] Austin, J. L. (1962). *How to do things with words*. Oxford University Press, Oxford, UK.
- [6] Baddeley, A. (2007). *Working Memory, Thought, and Action*. Oxford University Press, New York.
- [7] Becker, C., Kopp, S., und Wachsmuth, I. (2004). Simulating the emotion dynamics of a multimodal conversational agent. In *Affective Dialogue Systems*, LNAI 3068, pages 154–165, Berlin. Springer.
- [8] Benoit, C., Martin, J.-C., Pelachaud, C., Schomaker, L., und Suhm, B. (2000). Audio-visual and multimodal speech-based systems. In Gibbon, D., Mertins, I., und Moore, R., editors, *Handbook of multimodal and spoken dialogue systems: Resources, terminology and product evaluation*, pages 102–203. Dordrecht, The Netherlands: Kluwer.
- [9] Bergmann, K. und Kopp, S. (2009). Increasing expressiveness for virtual agents – autonomous generation of speech and gesture for spatial description tasks. In Decker, K., Sichman, J., Sierra, G., und Castelfranchi, editors, *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, pages 361–368. Ann Arbor, MI: IFAAMAS.
- [10] Bergmann, K. und Kopp, S. (2012). Gestural alignment in natural dialog. In *Proc. of the Int. Conference of the Cognitive Science Society (CogSci 2012)*.
- [11] Bratman, M. (1987). *Intention, Plans, and Practical Reason*. Havard University Press.
- [12] Brennan, S. E. und Clark, H. H. (1996). Conceptual pacts and lexical choice in conversation. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 22:1482–1493.
- [13] Bunt, H. (2011). Multifunctionality in dialogue. *Computer, Speech and Language*, 25:222–246.
- [14] Buschmeier, H., Bergmann, K., und Kopp, S. (2010). Adaptive Expressiveness – Virtual Conversational Agents That Can Align to Their Interaction Partner. In van der Hoek, W., Kaminka, G. A., Luck, M., und Sen, S., editors, *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 91–98. International Foundation for Autonomous Agents and Multiagent Systems.
- [15] Carletta, J., Isard, A., Isard, S., Kowtko, J. C., Doherty-Sneddon, G., und Anderson., A. H. (1997). The reliability of a dialogue structure coding scheme. *Computational Linguistics*,

- 23(1):13–31.
- [16] Cassell, J., Stone, M., und Yan, H. (2000a). Coordination and context-dependence in the generation of embodied conversation. In *Proceedings of the International Natural Language Generation Conference*, pages 171–178, Mitzpe Ramon, Israel.
 - [17] Cassell, J., Sullivan, J., Prevost, S., und Churchill, E., editors (2000b). *Embodied Conversational Agents*. The MIT Press, Cambridge (MA).
 - [18] Clark, H. H. (1996). *Using Language*. Cambridge University Press, Cambridge, UK.
 - [19] Cohen, R., Allaby, C., Cumbara, C., Fitzgerald, M., Ho, K., Hui, B., Latulipe, C., Lu, F., Moussa, N., Pooley, D., Qian, A., und Siddiqi, S. (1998). What is initiative? *User Modeling and User-Adapted Interaction*, 8:171–214.
 - [20] Cooper, R. und Larsson, S. (1999). Dialogue moves and information states. In *Proceedings of the Third International Workshop on Computational Semantics*, pages 398–400.
 - [21] Core, M. G. und Allen, J. F. (1997). Working Notes: AAAI Fall Symposium on Communicative Action in Humans and Machines.
 - [22] Damasio, A. (1994). *Descartes' Error: Emotion, Reason, and the Human Brain*. Putnam, New York.
 - [23] Duncan, S. (1972). Some signals and rules for taking speaking turns in conversations. *Journal of Personality and Social Psychology*, 23(2):283–292.
 - [24] Elio, R. (2002). Belief-desire-intention agency in a general cognitive architecture. *Cognitive Science Quarterly*, 2:321–340.
 - [25] Fink, G., Schillo, C., Kummert, F., und Sagerer, G. (1998). Incremental speech recognition for multimodal interfaces. In *Proceedings 24th Annual Conference of the IEEE Industrial Electronics Society*, pages 2012–2017.
 - [26] Fröhlich, M. und Wachsmuth, I. (1998). Gesture recognition of the upper limbs – from signal to symbol. In Sowa, T. und Wachsmuth, I., editors, *Gesture and Sign Language in Human-Computer Interaction*, pages 173–184. Springer.
 - [27] Georgeff, M. und Lansky, A. (1987). Reactive reasoning and planning. In *Proceedings of the Sixth National Conference of Artificial Intelligence*, pages 677–682, Menlo Park, California, USA. AAAI, AAAI Press/MIT Press.
 - [28] Georgeff, M., Pell, B., Pollack, M., Tambe, M., und Wooldridge, M. (1999). The Belief-Desire-Intention Model of Agency. In Müller, J., Singh, M. P., und Rao, A. S., editors, *Proceedings of the 5th International Workshop on Intelligent Agents V: Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555, pages 1–10, Heidelberg, Germany. Springer-Verlag.
 - [29] Goffman, E. (1983). The interaction order. *American Sociological Review*, 48:1–17.
 - [30] Goodwin, C. (1981a). Achieving Mutual Orientation at Turn Beginning. *Conversational Organization: Interaction between speakers and hearers*, pages 55–89.
 - [31] Goodwin, C. (1981b). *Conversational organization: Interaction between speakers and hearers*. Academic Press New York.
 - [32] Gratch, J., Rickel, J., André, E., Cassell, J., Petajan, E., und Badler, N. (2002). Creating interactive virtual humans: Some Assembly Required. *IEEE Intelligent Systems*, pages 54–63.
 - [33] Halliday, M. (1985). *An Introduction to Functional Grammar*. Arnold.
 - [34] Horvitz, E. (2007). Reflections on Challenges and Promises of Mixed-initiative Interaction. *AI Magazine*, pages 19–22.

- [35] Howden, N., Rönnquist, R., Hodgson, A., und Lucas, A. (2001). JACK Intelligent Agents – Summary of an Agent Infrastructure. In *Proceedings of the 5th ACM International Conference on Autonomous Agents*.
- [36] Huber, M. J. (1999). JAM: A BDI-theoretic Mobile Agent Architecture. In *Proceedings of the International Conference on Autonomous Agents*, pages 236–243, Seattle, WA.
- [37] Johnston, M. und Bangalore, S. (2001). Finite-state methods for multimodal parsing and integration. In *Proc. of the ESSLLI Summer School on Logic, Language, and Information, Helsinki, Finland*, pages 1–6.
- [38] Jónsdóttir, G. R., Thorisson, K. R., und Nivel, E. (2008). Learning smooth, human-like turn-taking in realtime dialogue. In *Proceedings of the 8th international conference on Intelligent Virtual Agents*, IVA '08, pages 162–175, Berlin, Heidelberg. Springer-Verlag.
- [39] Kendon, A. (1967). Some functions of gaze direction in social interaction. *Acta Psychologica*, 26:1–47.
- [40] Kopp, S. (2010). Social resonance and embodied coordination in face-to-face conversation with artificial interlocutors. *Speech Communication*, 52(6):587–597.
- [41] Kopp, S., Allwood, J., Ahlsen, E., Grammer, K., und Stocksmeier, T. (2008). Modeling embodied feedback in a virtual human. In Wachsmuth, I. und Knoblich, G., editors, *Modeling Communication With Robots And Virtual Humans*, pages 18–37. Springer-Verlag, Berlin.
- [42] Kopp, S., Jung, B., Lessmann, N., und Wachsmuth, I. (2003). Max – A Multimodal Assistant in Virtual Reality Construction. *KI-Künstliche Intelligenz*, 4/03:11–17.
- [43] Kopp, S. und Wachsmuth, I. (2004). Synthesizing multimodal utterances for conversational agents. *Computer Animation and Virtual Worlds*, 15(1):39–52.
- [44] Lakin, J. L., Jefferis, V. E., Cheng, C. M., und Chartrand, T. L. (2003). The chameleon effect as social glue: evidence for the evolutionary significance of nonconscious mimicry. *Journal of Nonverbal Behavior*, 27(3):145–162.
- [45] Latoschik, M. (2001a). *Multimodale Interaktion in Virtueller Realität am Beispiel der virtuellen Konstruktion*, volume 251 of *DISKI*. Akademische Verlagsgesellschaft Aka GmbH, Berlin.
- [46] Latoschik, M. E. (2001b). A gesture processing framework for multimodal interaction in virtual reality. In *Proc. of 1st International Conference on Computer Graphics, Virtual Reality and Visualization in Africa (Afrigraph 2001)*, pages 95–100.
- [47] Latoschik, M. E. (2002). Designing transition networks for multimodal VR interactions using a markup language. In *Proc. of the 4th IEEE Int. Conf. on Multimodal Interfaces (ICMI 2002)*, pages 411–416. IEEE Computer Society.
- [48] Lessmann, N., Kopp, S., und Wachsmuth, I. (2006). Situated interaction with a virtual human – perception, action, and cognition. In Rickheit, G. und Wachsmuth, I., editors, *Situated Communication*, pages 287–323. Mouton de Gruyter, Berlin.
- [49] Lessmann, N., Kranstedt, A., und Wachsmuth, I. (2004). Towards a cognitively motivated processing of turn-taking signals for the embodied conversational agent Max. In *AAMAS 2004 Workshop on Embodied Conversational Agents: Balanced Perception and Action*, pages 57–64.
- [50] McNeill, D. (1992). *Hand and Mind: What Gestures Reveal about Thought*. University of Chicago Press, Chicago.
- [51] McTear, M. (2002). Spoken dialogue technology: Enabling the conversational user interface. *ACM Computing Surveys*, 34(1):90–169.

- [52] Mehrabian, A. und Ferris, S. R. (1967). Inference of attitudes from non-verbal communication in two channels. *Journal of Consulting Psychology*, 31:248–252.
- [53] Oberauer, K. (2003). The Multiple Faces of Working Memory: Storage, Processing, Supervision, and Coordination. *Intelligence*, 31:167–193.
- [54] Ortony, A., Clore, G. L., und Collins, A. (1988). *The Cognitive Structure of Emotions*. Cambridge University Press.
- [55] Pfeiffer, T. und Latoschik, M. E. (2004). Resolving object references in multimodal dialogues for immersive virtual environments. In *Proceedings of the IEEE Virtual Reality 2004*, Illinois: Chicago.
- [56] Pfeiffer-Lessmann, N. und Wachsmuth, I. (2008). Toward alignment with a virtual human – Achieving joint attention. In Dengel, A., Berns, K., und Breuel, T., editors, *KI 2008: Advances in Artificial Intelligence*, Berlin. Springer-Verlag.
- [57] Pickering, M. J. und Garrod, S. (2004). Toward a mechanistic psychology of dialogue. *Behavioral and Brain Science*, 27:169–226.
- [58] Poggi, I. und Pelachaud, C. (2000). Performative facial expression in animated faces. In [17].
- [59] Pokahr, A., Braubach, L., und Lamersdorf, W. (2005). Jadex: A BDI Reasoning Engine. In Bordini, R., Dastani, M., Dix, J., und Seghrouchni, A. E. F., editors, *Multi-Agent Programming*, pages 149–174. Springer Science+Business Media Inc., USA.
- [60] Prillwitz, S., Leven, R., Zienert, H., Hanke, T., und Henning, J. (1989). *HamNoSys version 2.0: Hamburg Notation System for Sign Languages: An introductory guide* (Vol. 5). Hamburg: Signum Press.
- [61] Rao, A. und Georgeff, M. (1998). Decision procedures of BDI logics,. *Journal of Logic and Computation*, 8(3):293–344.
- [62] Raux, A. und Eskenazi, M. (2009). A finite-state turn-taking model for spoken dialog systems. In *Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL*, pages 629–637.
- [63] Reichardt, D. M., editor (2011). *KI – Künstliche Intelligenz*. Volume 25, Issue 3 / August 2011 (Special Issue on Emotion and Computing).
- [64] Reiter, E. und Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press.
- [65] Rosenbloom, P., Laird, J., und Newell, A. (1993). *The Soar Papers: Research on Integrated Intelligence*. MIT Press, Cambridge, MA, USA.
- [66] Sacks, H., Schegloff, E., und Jefferson, G. (1974). A simplest systematics for the organization of turn-taking for conversation. *Language*, 50:696–735.
- [67] Scherer, K., Bänzinger, T., und Roesch, E., editors (2010). *Blueprint for Affective Computing*. Oxford University Press.
- [68] Searle, J. (1969). *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, New York.
- [69] Sidner, C., Lee, C., und Lesh, N. (2003). Engagement Rules for Human-Robot Collaborative Interactions. In *IEEE International Conference on Systems, Man & Cybernetics (CSMC)*, volume 4, pages 3957–3962. IEEE Press.
- [70] Sowa, T. (2006). *Understanding Coverbal Iconic Gestures in Shape Descriptions*. PhD thesis, Doctoral dissertation, DISKI series, Vol. 294, Berlin: AKA/Infix.
- [71] Stone, M., DeCarlo, D., Oh, I., Rodriguez, C., Stere, A., Lees, A., und Bregler, C. (2004). Speaking with hands: Creating animated conversational characters from recordings of human performance. In *Proceedings of SIGGRAPH '04*, pages 506–513.

- [72] Thorisson, K. R. (2002). Natural Turn-Taking Needs No Manual: Computational Theory and Model, from Perception to Action. In *Multimodality in Language and Speech Systems*, pages 173–207. Kluwer Academic Publishers, The Netherlands.
- [73] Traum, D. (1996). Conversational agency: the TRAINS-93 dialogue manager. In *Proc. 11th Workshop on Language Technology: Dialogue Management in Natural Language Systems*, Universiteit Twente, Enschede, The Netherlands.
- [74] Traum, D. und Larsson, S. (2003). The information state approach to dialogue management. In Smith, R. und Kuppeveld, J., editors, *Current and New Directions in Dialogue*. Kluwer.
- [75] Wachsmuth, I., Lenzen, M., und Knoblich, G., editors (2008). *Embodied Communication in Humans and Machines*. Oxford University Press.
- [76] Waibel, A., Vo, M. T., Duchnowski, P., und Manke, S. (1996). Multimodal interfaces. *Artificial Intelligence Review*, 10:299–319.
- [77] Winograd, T. (1972). *Understanding Natural Language*. Academic Press, New York.
- [78] Wooldridge, M. (2000). *Reasoning about Rational Agents*. MIT Press, Cambridge, MA, USA.
- [79] Yngve, V. H. (1970). On getting a word in edgewise. In *Papers from the 6th Regional Meeting of the Chicago Linguistics Society*, pages 567–578. University of Chicago.
- [80] Zimbardo, P. G. (1995). *Psychologie*. Springer-Verlag.

16 Semantic Web

Gerd Gröner, Ansgar Scherp und Steffen Staab

16.1 Einleitung

Im World Wide Web werden unstrukturierte Informationen und informelles Wissen in Form von Hypertext dargestellt, um die Verbreitung von Informationen und Wissen zu erleichtern. Ziel des *Semantic Webs* ist es, strukturierte Informationen und formales Wissen verteilt im Web bereitzustellen, um daraus mittels automatisierten Schlussfolgerungen Antworten ableiten zu können. Auf diese Weise lassen sich Wissensbestandteile aus verschiedenen Quellen intelligent miteinander integrieren und komplexe Fragen beantworten. Anfragen, die sich auf diese Weise beantworten lassen, sind zum Beispiel „Welche Arten von Musik werden in britischen Radiostationen gespielt?“ oder „Welche Radiostation spielt Lieder von schwedischen Künstlern?“ Heutige Suchmaschinen können solche Anfragen nur unzureichend beantworten, obwohl alle benötigten Informationen bereits im Web verfügbar sind. Insbesondere liegt eine große Menge an Daten bereits in maschinenverarbeitbaren Formaten im Semantic Web vor. Allerdings können die Inhalte nur sehr eingeschränkt von heutigen Suchmaschinen analysiert werden. Im Wesentlichen indizieren Suchmaschinen die Hypertexte und Dokumente im Web, die in natürlicher Sprache vorliegen, einzeln und sind nicht in der Lage, dort vorhandene Inhalte intelligent miteinander zu kombinieren. Die Stärke des Semantic Webs liegt nun darin, Daten und deren Bedeutung aus verschiedenen Quellen miteinander zu kombinieren.

Schauen wir uns dazu an, wie man mit Hilfe des Semantic Webs die oben genannten Fragen beantworten kann: Die BBC veröffentlicht die Abspiellisten ihrer Radiostationen online in Formaten des Semantic Webs. Die Musikgruppe „ABBA“ hat einen eindeutigen Bezeichner in Form einer URI (<http://www.bbc.co.uk/music/artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist>). Diese URI kann verwendet werden, um die Musikgruppe mit Informationen aus dem Musikportal MusicBrainz¹ zu verknüpfen. MusicBrainz kennt die Mitglieder der Band, wie beispielsweise Benny Andersson, sowie das Genre und die Lieder. Zudem ist MusicBrainz mit Wikipedia verknüpft, um beispielsweise Informationen zu Künstlern, wie Biographien auf DBpedia [3], nutzen zu können. Informationen über britische Radiostationen können in Form von Listen auf Web-Seiten wie beispielsweise ListenLive² gefunden werden, welche ebenfalls in eine Semantic Web Repräsentation, eine sogenannte Ontologie, überführt werden könnten – nämlich als eine Beschreibung von Objekten und ihren Beziehungen. MusicBrainz ist mittels der Playcount-Ontologie³ mit der BBC verbunden. Derzeit werden von der BBC mindestens neun verschiedene Ontologien mit unterschiedlichem Grad an Formalität und verschiedenen Beziehungen zueinander genutzt um ihre Daten zu beschreiben

¹ <http://musicbrainz.org/>

² <http://www.listenlive.eu/uk.html>

³ <http://dbtune.org/bbc/playcount/>

(vgl. auch [47]). Die Bedeutung von *Beziehungen* zwischen Daten wird ebenfalls mittels Ontologien des Semantic Webs beschrieben, z.B. bietet Dublin Core⁴ ein Metadaten-Schema zur Beschreibung allgemeiner Eigenschaften von Informationsobjekten. Dadurch dass diese Daten im Semantic Web verfügbar sind, können weitere Fragen gestellt werden, z.B. wie oft Musik eines bestimmten Genre von britischen Radiostationen gespielt wird oder welche Radiostationen Lieder von schwedischen Künstlern spielen. Um Fragen in diesem und in anderen Szenarien zu beantworten, werden generische Software-Komponenten, Sprachen und Protokolle benötigt, die nahtlos miteinander interagieren können.

Das Szenario zeigt, dass die Daten im Semantic Web aus verschiedenen Quellen stammen können und miteinander vernetzt sind. Abgesehen von technischen Gesichtspunkten ist das Semantic Web auch als ein gesellschaftspolitisches Phänomen zu verstehen. Ähnlich dem World Wide Web veröffentlichen verschiedene Personen und Organisationen im Semantic Web Daten und arbeiten zusammen, um diese miteinander zu verknüpfen und zu verbessern. Neben dem oben genannten Beispiel ist das Semantic Web auch für eine Vielzahl anderer Anwendungsgebiete einsetzbar, wie beispielsweise für persönliches Informationsmanagement [22], für die interaktive Exploration sozialer Medien [52] oder für mobile Informationssysteme [11].

In diesem Kapitel wird der prinzipielle Aufbau des Semantic Webs vorgestellt. Analog zu einschlägiger Grundlagenliteratur im Bereich Semantic Web [1, 36, 38] befasst sich dieses Kapitel mit den verschiedenen grundlegenden Semantic Web Technologien und deren Anwendung. Sehr viele dieser Technologien stammen aus der Künstlichen Intelligenz oder haben zumindest einen starken Bezug dazu. Ontologien dienen zur formalen Repräsentation von Wissen. Die Web Ontology Language (OWL) [29] basiert auf Beschreibungslogik und entsprechend werden Schlussfolgerungsdienste von wissensbasierten Systemen für OWL-Ontologien angewendet.

Im folgenden Abschnitt stellen wir zunächst die allgemeine Architektur des Semantic Webs vor. Anschließend zeigen wir in Abschnitt 16.3, wie verteilte Daten mit Hilfe von Technologien des Semantic Webs verwaltet, das heißt verknüpft und angefragt werden können. Das im Beispiel verwendete Netzwerk von Ontologien wird in Abschnitt 16.4 näher analysiert und allgemeine Strategien zur verteilten Wissensrepräsentation und -integration im Semantic Web werden dargestellt. In Abschnitt 16.5 zeigen wir, wie Schlussfolgerungen aus semantischen Daten gezogen werden können. Um die Daten in einer sich über das Web erstreckenden Wissensbasis verwalten zu können, sind eine Reihe von Herausforderungen zu lösen. Diese sind zum einen die Identifizierung von Ressourcen und deren Verknüpfung (Abschnitt 16.6), Herkunft und Vertrauenswürdigkeit der Daten (Abschnitt 16.7) sowie generische Benutzungsschnittstellen und Semantic Web Anwendungen (Abschnitt 16.8). Der Artikel schließt mit einer Übersicht der aktuellen, praktischen Nutzung von semantischen Technologien und einer Zusammenfassung.

16.2 Semantic Web Architektur

Das Szenario in Abschnitt 16.1 beschreibt *was* das Semantic Web als Infrastruktur realisiert, aber nicht *wie* dies erreicht wird. In der Tat wurden die beschriebenen Fähigkeiten *im Kleinen* bereits von einigen wissensbasierten Systemen, die aus der Tradition der Künstlichen Intelligenz stammen, umgesetzt. Für eine Umsetzung *im Großen* mangelte es diesen Systemen aber an Flexibilität, Robustheit und Skalierbarkeit. Teilweise lag dies an der Komplexität der verwendeten

⁴ <http://dublincore.org/documents/dc-rdf/>

Algorithmen. So waren beispielsweise Wissensbasen in Beschreibungslogik, die als Grundlage von Webontologien dienen, in den 1990-er Jahren bezüglich ihrer Größe so eingeschränkt, dass sie höchstens einige hundert Konzepte handhaben konnten (vgl. [37]). Zwischenzeitlich wurden enorme Verbesserungen erreicht. Stark angestiegene Rechenleistung und optimierte Algorithmen ermöglichen eine praktische Handhabung von großen Ontologien wie SNOMED⁵ mit hunderttausenden von Axiomen. Allerdings gibt es einige grundlegende Unterschiede zwischen klassischen wissensbasierten Systemen und dem Semantic Web.

Die Verwaltung von Daten in traditionellen wissensbasierten Systemen weist Schwachstellen im Bezug auf die Verarbeitung großer Mengen an Daten und Datenquellen auf, unter anderem wegen (1) unterschiedlicher zugrundeliegender Formalismen, (2) verteilten Standorten, (3) verschiedenen Befugnissen, (4) unterschiedlicher Datenqualität und (5) einer hohen Änderungshäufigkeit der verwendeten Daten. Um mit diesen Problemen umgehen zu können, wendet das Semantic Web einige grundlegende Prinzipien an:

1. *Explizite und einfache Datendarstellung*: Eine allgemeine Datenrepräsentation abstrahiert von den zugrundeliegenden Formaten und erfasst nur das Wesentliche.
2. *Verteilte Systeme*: Ein verteiltes System arbeitet auf einer großen Menge an Datenquellen, ohne dass eine zentrale Steuerung regelt, welche Informationen wohin und wem gehören.
3. *Querverweise*: Die Vorteile eines Netzwerks von Daten bei der Beantwortung von Anfragen begründen sich nicht alleine aus der reinen Mengen von Daten sondern aus ihrer Verknüpfung, die es erlaubt, Daten und Datendefinitionen aus anderen Quellen wieder zu verwenden.
4. *Lose Koppelung mit gemeinsamen Sprachkonstrukten*: Das World Wide Web und ebenso das Semantic Web sind Mega-Systeme, also Systeme, die aus vielen Teilsystemen bestehen, die ihrerseits groß und komplex sind. In einem solchen Mega-System müssen einzelne Bestandteile lose gekoppelt sein, um größtmögliche Flexibilität zu erreichen. Die Kommunikation zwischen den Bestandteilen erfolgt auf Grundlage von standardisierten Sprachen, wobei diese individuell an spezifische Systeme angepasst werden können.
5. *Einfaches Veröffentlichen und einfacher Konsum*: Gerade in einem Mega-System muss die Teilnahme, also das Veröffentlichen und der Konsum von Daten, möglichst einfach sein.

Diese Prinzipien werden durch einen Mix an Protokollen, Sprachdefinitionen und Softwarekomponenten erzielt. Einige dieser Bestandteile sind bereits durch das W3C standardisiert, das sowohl Syntax als auch formale Semantik der Sprachen und Protokolle festgelegt hat. Weitere Bestandteile sind noch nicht standardisiert, aber sie sind bereits im sogenannten *Semantic Web Layer Cake* von Tim Berners-Lee vorgesehen (vgl. <http://www.w3.org/2007/03/layerCake.png>). Wir stellen eine Variante des Semantic Web Layer Cakes vor, wobei wir zwischen standardisierten Sprachen und derzeitigen Entwicklungen unterscheiden. Eine graphische Darstellung des Semantic Web Layer Cakes ist in Abbildung 16.1 dargestellt. Nachfolgend werden die entsprechenden Bausteine kurz vorgestellt.

HTTP/URI/IRI

Da das Semantic Web dezentral organisiert ist, benötigt man Mechanismen, um nicht nur auf selbst eingeführte Entitäten zu verweisen, sondern auch auf Entitäten die von Dritten veröffentlicht wurden. Entitäten (auch Ressourcen genannt) werden im Internet durch sogenannte Uniform Resource Identifiers (URIs) [5] identifiziert. Wie im Web halten sich URIs

⁵ <http://www.snomed.org/>

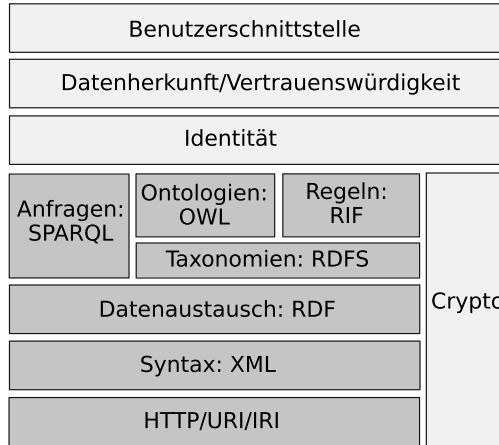


Abbildung 16.1: Darstellung der Bestandteile als Semantic Web Layer Cake. W3C-Sprachstandards sind in dunkelgrau dargestellt. Derzeitige Entwicklungen sind in hellgrau abgebildet.

auch im Semantic Web an das Domain Name System (DNS), das die globale Eindeutigkeit von Domainnamen und URIs garantiert, welche mit „http“ beginnen. In unserem Beispiel, beschreibt die URI `http://www.bbc.co.uk/music/artists/2f031686-3f01-4f33-a4fc-fb3944532efa#artist` Benny Andersson von ABBA. Ein Anwender kann eine URI, die beispielsweise auf ABBA verweist, dereferenzieren, indem ein sogenannter Look-up mittels HTTP ausgeführt wird, um eine ausführliche Beschreibung der URI zu erhalten. Internationalized Resource Identifiers (IRIs) [15] ergänzen URIs um internationale Zeichensätze aus Unicode/ISO10646. HTTP-Anfragen und URI/IRI-Referenzen werden in Abschnitt 16.3 detaillierter behandelt.

XML

Nachdem nun Ressourcen eindeutig referenzierbar und dereferenzierbar sind, wird eine Syntax benötigt, um Beschreibungen von Ressourcen im Web auszutauschen. Die Extensible Markup Language (XML) wird zur Strukturierung von Dokumenten verwendet und ermöglicht die Spezifikation und Serialisierung strukturierter Daten.

RDF

Neben der Referenzierbarkeit von Ressourcen und einer einheitlichen Syntax für den Austausch von Daten, benötigt man ein Datenmodell, das es erlaubt, Ressourcen sowohl im Einzelnen als auch in ihrer Gesamtheit und ihrer Verknüpfung zu beschreiben. Eine integrierte Darstellung von Daten aus mehreren Quellen wird durch ein auf gerichteten Graphen basierendes Datenmodell erreicht [46]. Die entsprechende W3C-Standardsprache ist RDF (Resource Description Framework). RDF-Graphen können auf verschiedene Arten serialisiert werden. Am häufigsten ist die XML-basierte Serialisierung. Ein RDF-Graph besteht aus einer Menge von RDF-Tripeln, wobei ein Tripel aus Subjekt, Prädikat (Eigenschaft) und Objekt besteht. Auf RDF wird, in Verbindung mit der Verteilung von Daten, in Abschnitt 16.3 weiter eingegangen.

SPARQL

Nachdem RDF die Integration von Daten verschiedener Quellen ermöglicht, ist der nächste Schritt die Anfrage von RDF-Graphen. SPARQL⁶ (ein rekursives Akronym für SPARQL Protocol and RDF Query Language) ist eine deklarative Anfragesprache für RDF-Graphen. SPARQL 1.1⁷ ist die aktuelle Version von SPARQL. SPARQL-Anfragen werden ausführlicher in Abschnitt 16.3 vorgestellt.

RDFS

RDF-Graphen können die Bedeutung von Daten nur teilweise beschreiben. Sehr oft werden Konstrukte zur Modellierung von hierarchischen Beziehungen zwischen Klassen und Eigenschaften benötigt. Derartige Beziehungen werden typischerweise in Taxonomien und Ontologien beschrieben. RDFS (RDF Schema) ist eine Ontologie-Beschreibungssprache, die unter anderem Hierarchien zwischen Klassen und Eigenschaften beschreiben kann [12].

OWL

Daten von verschiedenen Quellen sind sehr heterogen. RDFS ist nicht ausdrucksstark genug, um Daten aus verschiedenen Quellen zusammenzuführen und darüber Konsistenzkriterien zu definieren, wie beispielsweise die Disjunkttheit von Klassen. OWL (Web Ontology Language) ist eine Ontologie-Beschreibungssprache, die im Vergleich zu RDFS ausdrucksstärkere Sprachkonstrukte bereit hält. Beispielsweise ermöglicht OWL die Spezifikation von Äquivalenzen zwischen Klassen und Kardinalitätseinschränkungen von Eigenschaften [29].

Die Bedeutung von OWL-Konstrukten kann durch zwei alternative Semantiken definiert werden. Die RDF-basierte Semantik und die direkte modelltheoretische Semantik, auf der Semantik von Beschreibungslogik ($\mathcal{S}\mathcal{R}\mathcal{O}\mathcal{J}\mathcal{Q}$) aufbaut und Schlussfolgerungen in Form von deduktiver Inferenz ermöglicht. OWL 2 als die aktuelle Version von OWL beinhaltet mehrere Untersprachen, sogenannte Profile, die je nach Anwendungskontext zwischen Ausdrucksmächtigkeit und effizienter Schlussfolgerung abwählen. OWL wird ausführlich in Abschnitt 16.4 behandelt und Inferenz in OWL wird in Abschnitt 16.5 beispielhaft dargestellt.

RIF

Regeln sind weitere Formalismen zur Repräsentation von Wissen im Semantic Web. RIF (Rule Interchange Format)⁸ beinhaltet eine Menge von Regelsprachen, die in logische Regelsprachen und ereignisbasierte Regelsprachen unterteilt werden können. RIF Core Dialect [8] beschreibt eine Teilmenge von verbreiteten Regelsprachen aus beiden Mengen.

Crypto

Weitere Aspekte im Semantic Web sind Verschlüsselung und Authentifizierung, um sicher zu stellen, dass Datenübertragungen nicht abgehört, gelesen oder modifiziert werden können. Crypto-Module, wie beispielsweise SSL (Secure Socket Layer), verifizieren digitale Zertifikate und ermöglichen Datenschutz und Authentifizierung.

⁶ <http://www.w3.org/TR/rdf-sparql-query/>.

⁸ http://www.w3.org/2005/rules/wiki/RIF_Working_Group.

Identifizierung und Verknüpfung

Inhalte, die aus vielen Datenquellen aggregiert sind, können viele verschiedene Identitäten beinhalten, die jedoch das selbe reale Objekt repräsentieren. Integration und Verknüpfungsmechanismen ermöglichen Bezüge zwischen Daten aus verschiedenen Quellen herzustellen. Eine weitere Betrachtung dieser Themen erfolgt in Abschnitt 16.6.

Herkunft und Vertrauenswürdigkeit

Daten im Semantic Web können mit zusätzlicher Information über ihre Vertrauenswürdigkeit und Herkunft erweitert werden. Abschnitt 16.7 beschreibt diesen Aspekt ausführlich.

Benutzeroberfläche

Eine Benutzeroberfläche ermöglicht Anwendern die Interaktion mit Daten im Semantic Web. Aus funktioneller Sicht sind einige Benutzeroberflächen generisch und arbeiten auf der Graphstruktur der Daten, wobei andere auf bestimmte Aufgaben, Anwendungen oder Ontologien zugeschnitten sind. Neue Paradigmen untersuchen aktuell das Spektrum an möglichen Benutzeroberflächen zwischen Allgemeingültigkeit und speziellen Anforderungen von Endanwendern. Benutzeroberflächen werden im Zusammenhang mit verschiedenen Beispielanwendungen in Abschnitt 16.8 behandelt.

16.3 Verteilte semantische Daten im Web

Das zentrale Ziel des Semantic Webs ist das Teilen von Wissen und Informationen und das Zusammenwirken und die Kooperation von menschlichen und maschinellen Akteuren. Jeder kann eine Ontologie erstellen und sie mit anderen Datenquellen so verknüpfen, dass daraus ein Mehrwert für diesen Akteur entsteht und dabei aber als „Abfallprodukt“ andere Akteure diese neuen Verknüpfungen ebenfalls verwenden können. Auf diese Weise entsteht eine Daten- und Wissensbasis, die es erlaubt aus der enormen Menge an Informationen und ihrer Verknüpfungen neue Zusammenhänge abzufragen und zu entdecken. Um diese Art der kollektiver Wissenserzeugung zu unterstützen sind effiziente Zugriffe auf verteilte Daten, klar definierte Veröffentlichungsprinzipien und geeignete Anfragemöglichkeiten notwendig.

16.3.1 Verknüpfte Daten

Die *Linked Data* Prinzipien⁹ beschreiben relevante Methoden zur Darstellung, Veröffentlichung und Verwendung von Daten. Sie können wie folgt zusammengefasst werden:

1. URIs werden als Namen für Entitäten verwendet.
2. Das Protokoll HTTP GET wird verwendet, um Beschreibungen zu einer URI abzurufen.
3. Datenprovider sollen auf HTTP GET Abfragen von URIs relevante Informationen mit Hilfe von Standardsprachen (z.B. in RDF) zurückgeben.
4. Verknüpfungen (Links) zu anderen URIs sollen verwendet werden, um die Entdeckung und Verwendung von weiteren Informationen zu erleichtern.

⁹ <http://www.w3.org/DesignIssues/LinkedData.html>.

Die Veröffentlichung von Daten anhand der Linked Data Prinzipien ermöglicht einfachen Remote-Zugriff auf Daten via HTTP. Dies erlaubt die Erkundung von Ressourcen und die Navigation durch Ressourcen im Web. URIs (1) werden mittels HTTP-Anfragen dereferenziert (2), um zusätzliche Informationen über eine bestimmte Ressource zu erhalten, insbesondere können diese Informationen mittels standardisierter Syntax (3) ebenfalls Verknüpfungen zu anderen Ressourcen enthalten (4). Abbildung 16.2 stellt ein Beispiel für Linked Data zu ABBA dar. Das Beispiel stammt von MusicBrainz. Es beschreibt verschiedene Prädikate, die Entitäten mit der URI von ABBA verknüpfen, wie beispielsweise `foaf:member` und `rdf:type`. In der Abbildung ist ABBA, oder genauer die URI von ABBA, das Subjekt, Property bezieht sich auf Prädikate und die Werte (Value) stellen Objekte der RDF-Tripel dar. Das Prädikat `owl:sameAs` wird im weiteren Verlauf noch ausführlicher betrachtet. Die Präfixe `foaf`, `rdf` und `owl` beziehen sich auf Vokabulare der FOAF-Ontologie¹⁰, bzw. der Sprachspezifikationen von RDF und OWL.



The screenshot shows a linked data interface for the artist ABBA. At the top, it says "ABBA" and "Resource URI: <http://dbtune.org/musicbrainz/resource/artist/d87e52c5-bb8d-4da8-b941-9f4928627dc8>". Below this is a table with two columns: "Property" and "Value". The properties listed are: vocab:alias, bio:event, foaf:homepage, foaf:member, mo:musicbrainz, foaf:name, owl:sameAs, and rdf:type. The corresponding values are: Abba, <<http://dbtune.org/musicbrainz/resource/artist/d87e52c5-bb8d-4da8-b941-9f4928627dc8/birth>>, <<http://dbtune.org/musicbrainz/resource/artist/d87e52c5-bb8d-4da8-b941-9f4928627dc8/death>>, <<http://www.abbasite.com>>, <<http://dbtune.org/musicbrainz/resource/artist/042c35d3-0756-4804-b2c2-be57a683efa2>>, <<http://dbtune.org/musicbrainz/resource/artist/2f031686-3f01-4f33-a4fc-fb3944532efa>>, <<http://dbtune.org/musicbrainz/resource/artist/aebbbb417-0d18-4fec-a2e2-ce9663d1fa7e>>, <<http://dbtune.org/musicbrainz/resource/artist/fb77292-9712-4d03-94aa-bdb1d4771d38>>, <<http://musicbrainz.org/artist/d87e52c5-bb8d-4da8-b941-9f4928627dc8>>, ABBA, <<http://dbpedia.org/resource/ABBA>>, <<http://sv.wikipedia.org/wiki/Abba>>, <<http://www.bbc.co.uk/music/artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist>>, mo:MusicArtist.

Abbildung 16.2: Linked Data Beispiel für ABBA.

16.3.2 Anfragen mit SPARQL

Die Verknüpfung von Daten nach den Linked Data Prinzipien ermöglicht Anfragen über sehr große (verknüpfte) Datenmengen. Allerdings sind in vielen Anwendungen einfache Anfragen, wie beispielsweise die Ausgabe aller Prädikate der Ressource ABBA (vgl. Abbildung 16.2), nicht ausreichend, sondern komplexe Anfragen mit mehreren Anfragebedingungen werden benötigt. SPARQL ist eine Anfragesprache für RDF. Ähnlich zu SQL beschreibt die WHERE-Klausel unter welchen Bedingungen Daten selektiert werden.

¹⁰ <http://xmlns.com/foaf/spec/>.

Neben der eigentlichen Anfragesprache definiert SPARQL auch Zugriffsprotokolle und Datenformate. Ein Repotorium, das SPARQL unterstützt, muss diese Protokolle und Datenformate einhalten. Einige der meistgenutzten Repotorien sind Sesame [13], Jena [65], Virtuoso¹¹ und OWLIM¹². Repotorien übernehmen neben der Speicherung von Daten auch die Bereitstellung von Anfragemöglichkeiten in Form von *SPARQL-Endpunkten*. Datensätze, die einen SPARQL-Endpunkt haben sind in der Regel mittels REST-Protokoll [19] zu erreichen.

Neben der Anfrage von explizit vorhandenen Fakten sieht SPARQL auch die Unterstützung durch *Entailment-Regimes* vor. Diese erweitern die Abfrage von explizit vorhandenen Fakten um Fakten, die anhand von RDFS- und OWL-Konstrukten geschlussfolgert wurden (vgl. Abschnitt 16.5). Je nach Funktionsumfang des jeweiligen Repotoriums werden verschiedene (oder auch gar keine) Entailment-Regimes unterstützt. Die entsprechenden Schlussfolgerungsdienste werden durch die Verwendung von geeigneten Reasoners bereitgestellt.

SPARQL 1.1 liegen verschiedene Semantiken zugrunde und entsprechend werden alternative Entailment-Regimes realisiert. Ein Entailment-Regime ist das RDF- und das RDFS-Entailment für SPARQL-Anfragen. Ein RDF-Graph G_1 folgt aus einem anderen RDF-Graph G_2 wenn jede RDF-Interpretation die G_2 erfüllt auch G_1 erfüllt. Analog gilt dies für RDFS-Interpretationen, d.h. zusätzlich werden RDFS-Konstrukte wie *rdfs:subClassOf* interpretiert. Das D-Entailment basiert auf dem RDF-Entailment und berücksichtigt noch die Interpretation von Datentypen. Ein weiteres Entailment-Regime folgt der RDFS-basierten OWL 2 Semantik. Es erweitert das D-Entailment, d.h. die Interpretation von RDF-Graphen entspricht der Interpretation von dem RDF-Entailment. Zusätzlich wird das D-Entailment aber noch um die Interpretation von OWL 2-Konstrukten erweitert. Eine andere Semantik ist durch die OWL 2 Direct Semantic gegeben, die unter anderem auch die OWL 2 Profile OWL 2 DL, EL und QL abdeckt. Dieser Semantik liegt eine Abbildung von OWL 2 Konstrukten nach RDF-Tripel zugrunde.

Nachfolgend wird nochmals das Beispiel von MusicBrainz betrachtet, in dem Informationen über ABBA gesucht werden. Wir sind nun an den Interpreten von ABBA interessiert, die auch Mitglieder anderer Bands sind. Folgen wir dem Linked Data Prinzip, so würde das bedeuten, dass zunächst nach der URI angefragt wird, die für ABBA steht, dann würde zu den einzelnen Bandmitgliedern navigiert werden, und anschließend den Links zu allen Bands der Mitglieder gefolgt werden.

SPARQL geht über das prozedurale Verfolgen von Links hinaus und basiert auf dem Abgleich von Graphmustern (Graph Pattern Matching). Graphmuster aus der SPARQL-Anfrage werden mit vorhandenen RDF-Tripeln in den angefragten RDF-Graphen verglichen und eventuell nach weiteren Kriterien gefiltert. Das Graphmuster für Bands, deren Mitglieder ebenfalls ABBA-Mitglied sind, ist in Abbildung 16.3 dargestellt, und die entsprechende SPARQL-Anfrage ist in Abbildung 16.4 beschrieben.

Die eigentlichen Vergleichsbedingungen der Graphmuster mit RDF-Tripeln werden in der WHERE-Klausel beschrieben. Das Graphmuster besteht aus einzelnen Tripelmustern. An jeder Stelle eines Tripelmusters (Subjekt, Prädikat, Objekt) kann entweder eine URI oder eine Variable stehen, in der Objektposition auch ein Literal (eine Art String). Wenn die Nichtvariablen eines Tripelmusters mit vorhandenen Tripeln übereinstimmen, dann können die Variablen an die entsprechenden Werte dieser Tripel gebunden werden. Wenn mehrere Tripelmuster ein Graph-

¹¹ <http://virtuoso.openlinksw.com>.

¹² <http://www.ontotext.com/owlim>.

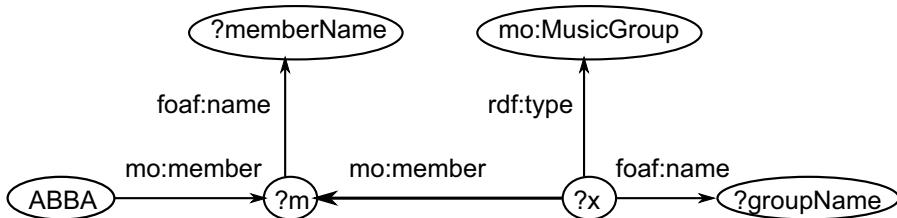


Abbildung 16.3: Graphische Darstellung einer Anfrage für Musikgruppen (dargestellt durch die Variable `?groupName`), deren Mitglieder ebenfalls Mitglieder bei ABBA sind. Die Variable `?m` bezieht sich auf die Mitglieder von ABBA. Der Knoten mit Beschriftung „ABBA“ stellt die URI für ABBA dar. Das Präfix `mo` bezieht sich auf die Musikontologie, `foaf` auf die FOAF-Ontologie und `rdf` auf das Vokabular der RDF-Spezifikation.

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mo: <http://purl.org/ontology/mo/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX bbc: <http://www.bbc.co.uk/music/>
SELECT ?memberName ?groupName
WHERE { bbc:artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist mo:member ?m .
       ?x mo:member ?m .
       ?x rdf:type mo:MusicGroup .
       ?m foaf:name ?memberName .
       ?x foaf:name ?groupName }
FILTER (?groupName <> "ABBA")
  
```

Abbildung 16.4: SPARQL-Anfrage für Musikgruppen, deren Mitglieder auch Mitglieder bei ABBA sind. Im ersten Tripelmuster des WHERE-Teils ist die URI von ABBA das Subjekt.

muster bilden, dann muss es Kombinationsmöglichkeiten dieser Variablenbindungen geben, die in der Belegung der gemeinsamen Variablen übereinstimmen, damit das Gesamt muster passen kann. Der WHERE-Klausel kann ein FILTER-Ausdruck folgen, der die Ergebnisse nach dem angegebenen Kriterium (hier muss der Name der Gruppe ungleich zum Namen „ABBA“ sein) reduziert. Das Schlüsselwort PREFIX ermöglicht die Einführung von Bezeichnern für URIs (vgl. Abbildung 16.4).

16.3.3 Anfragen auf verknüpfte und verteilte Daten

Anfragen auf einzelne Datenquellen sind die Grundbausteine verteilter Anfragen, allerdings bedarf es noch Erweiterungen um mehrere Datenquellen anzufragen. Ein möglicher Ansatz sind Anfragen von mehreren benannten RDF-Graphen (engl. named graphs). Ein benannter Graph in RDF ist eine Menge von RDF-Tripel. Eine Datenquelle kann aus mehreren benannten Graphen bestehen. Einige konkrete Implementierungen realisieren Anfragen über mehrere RDF-Graphen. Eine Architektur, sowie Indexstrukturen und Algorithmen zur Ausführung von verteilten Anfragen wurden in [60] vorgestellt. Dieser Ansatz wurde in Form von sogenannten Networked Graphs verfeinert und erweitert [53]. Networked Graphs ermöglichen neben der Anfrage auf verteilten Quellen und der Generierung von Sichten auf verteilte Graphen auch die Verbindung dieser Graphen in Form von rekursiven Sichten, die als CONSTRUCT-Anfragen definiert werden.

Eine Beispieldatenanfrage auf DBpedia und MusicBrainz ist in Abbildung 16.5 dargestellt. Es werden alle Künstler von ABBA (dargestellt durch das Subjekt `bbc:artists/d87e5`

2c5-bb8d-4da8-b941-9f4928627dc8#artist mit Namen und Biographie gesucht. Die Ergebnisse sind verknüpfte Informationen über Künstler die aus zwei Datenquellen stammen: DBpedia und MusicBrainz. Die Variable für Mitglieder (?member) ist das Verknüpfungselement beider Graphen, wie in Abbildung 16.6 dargestellt. Es ist durchaus möglich, dass URIs der selben Interpreten in DBpedia und MusicBrainz verschieden sind. In diesem Fall kann mittels der Beziehung owl:sameAs die Äquivalenz beider URIs definiert werden. Aus Sicht des Anwenders wird eine Anfrage, wie in Abbildung 16.5 dargestellt, über SPARQL-Endpunkte ausgeführt. Über SPARQL-Endpunkte werden Daten zweier benannter RDF-Graphen extrahiert. Networked Graphs verborgen die Komplexität von Anfragen auf verteilten und entfernten Repositoryn. Sie kombinieren die Ergebnisse der verteilten Anfragen zu einem Gesamtergebnis.

```

PREFIX mo: <http://purl.org/ontology/mo/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX bbc: <http://www.bbc.co.uk/music/>
CONSTRUCT { ?member mo:wikipedia ?biography . ?member foaf:name ?name}
FROM NAMED :MusicBrainz      FROM NAMED :DBpedia
WHERE {
  GRAPH :MusicBrainz {
    bbc:artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist mo:member ?member .
    ?member foaf:name ?name }
  GRAPH :DBpedia {
    ?member mo:wikipedia ?biography }
}
}
```

Abbildung 16.5: SPARQL-Anfrage für ABBA-Mitglieder.

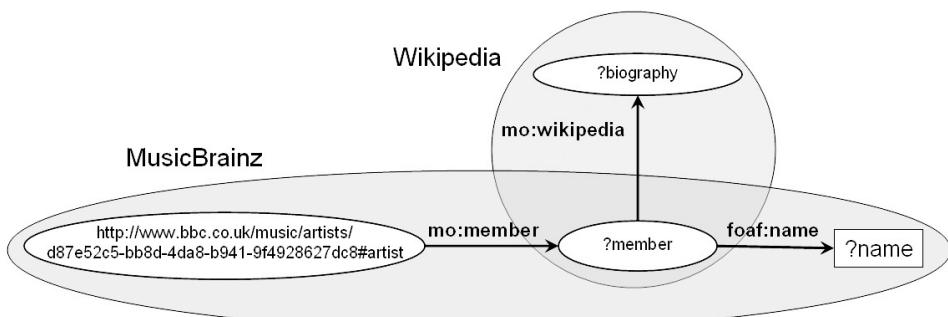


Abbildung 16.6: Verknüpfte Information über ABBA aus zwei Datenquellen.

Anfragen auf verteilte RDF-Daten sind sehr ähnlich zu Anfragen auf schemalosen und verteilten Datenbanken oder auf Peer-to-peer-Datenbanken (vgl. [31]). Es gibt noch weitere Anfrageparadigmen, welche direkt die Vorteile von Linked Data und den Linked Data Prinzipien nutzen. Das Anfrageprinzip von Hartig et al. [34] kombiniert die grundlegende Eigenschaft der Dereferenzierbarkeit von URIs in Linked Data mit Daten-Crawling. Es umgeht das Problem mangelnder Indizes, indem während der Anfrageausführung den RDF-Links gefolgt wird, die für die Anfrage relevant sein könnten, um weitere Informationen zu entdecken. Der Suchraum wird erweitert anhand der Tripelmuster in der Anfrage zu den nächst verbundenen Datenquellen, die zu einem Anfrageergebnis beitragen könnten. Umgesetzt wird dies mittels eines Indexmechanismus, der Datenbeschreibungen benutzt, die auf Informationen über Instanzen und Schemata beruhen [31]. Solche Indexstrukturen ermöglichen die Auswahl von relevanten Datenquellen für

Anfragen, die sich auf mehrere, anfangs eventuell noch unbekannte Datenquellen beziehen. Dieser Ansatz ermöglicht auch Anfragen an Linked Data Quellen, die keinen SPARQL-Endpunkt anbieten, allerdings auf Kosten der Vollständigkeit der Anfrageergebnisse. Detaillierte Aussagen zur Vollständigkeit dieses Anfrageparadigmas werden in [33, 35] getroffen.

16.4 Wissensrepräsentation und -integration

Eine Ontologie wird verstanden als formale, maschinenverarbeitbare Repräsentation von relevanten Begriffen und Relationen einer Domäne [43, 45]. Ontologien stellen damit eine gemeinsame Sicht dar [45], das heißt die formale Begriffsbildung von Ontologien drückt eine übereinstimmende Meinung verschiedener Ersteller aus.

16.4.1 Analyse des einführenden Beispiels

Um das Beispieldaten aus Abschnitt 16.1 zur Verknüpfung von MusicBrainz und dem BBC-Programm zu modellieren, werden mehrere Ontologien verwendet und zu einem Netzwerk im Web verbunden. Der Ansatz eine Ontologie nicht monolithisch zu definieren, sondern als einen Baustein in einem miteinander verknüpften Netz von Ontologien zu betrachten, ist eine wesentliche Neuorientierung im Semantic Web im Vergleich zu den klassischen KI-Ansätzen. Ein Ausschnitt für das im Szenario verwendete Netzwerk von Ontologien ist in Abbildung 16.7 dargestellt. Es wird eine an UML angelehnte grafische Notation zur Darstellung der Ontologien verwendet, da diese weit verbreitet und leicht verständlich ist. Die Rechtecke wie beispielsweise *ProgrammeItem* oder *MusicArtist* stellen die relevanten Konzepte der Musikdomäne dar. Diese sind aus verschiedenen Ontologien entnommen, die über die gestrichelten Kästen angedeutet sind. Beziehungen zwischen Konzepten sind über einen beschrifteten Pfeil dargestellt. Zur Illustration von Vererbungsbeziehungen zwischen Konzepten wird wie in UML ein Dreieckspfeils verwendet.

In MusicBrainz werden Künstler durch das Konzept *MusicArtist* aus der Music Ontology¹³ repräsentiert. Diese werden über die Relation *object* mit dem Konzept *Playcount* der Playcount Ontology verknüpft. verwendet, um die Anzahl der gespielten Musikstücke eines Künstlers darzustellen. Die Playcount Ontology ist über das Konzept *Brand* (auf Deutsch: Handelsmarke) mit der Programmontology der BBC¹⁴ verbunden. Des Weiteren ist das Konzept *Brand* über seine Oberklasse *Programme* mit dem Konzept *Service* verbunden, welches beschreibt wo eine Handelsmarke bzw. Künstler gespielt wird. Ein *Service* kann beispielsweise ein auf eine bestimmte Region zugeschnittene Version des BBC-Programms sein. Ein *Programme* wird von einem *Broadcaster* übertragen, welches vom Konzept *Organization* der FOAF-Ontologie spezialisiert wurde. Zudem wird das Ereignis einer Übertragung, das heißt eines *Broadcast* in der BBC-Ontologie als Spezialisierung des Konzeptes *Event* der Event Ontology¹⁵ modelliert. Ein *Broadcast* steht über die Relation *broadcast_of* in Bezug auf eine bestimmte Version eines übertragenen Elements wie beispielsweise eine bestimmte Version eines gekürzten Radioprogramms. Das Konzept *Version* hat die Relationen *time* und assoziiert damit ein *Interval* mit dem übertragenen Element für temporale Annotatio-

¹³ <http://musicontology.com>.

¹⁴ <http://www.bbc.co.uk/ontologies>.

¹⁵ <http://motools.sourceforge.net/event>.

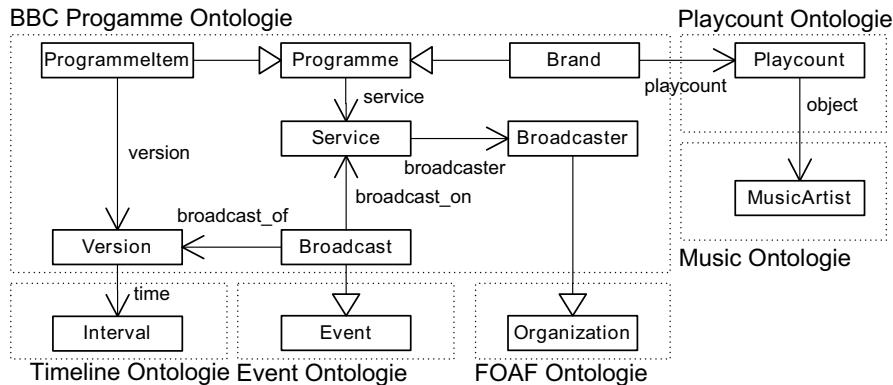


Abbildung 16.7: Ausschnitt der BBC-Ontology mit Verknüpfungen zu anderen Ontologien (Notation angelehnt an UML).

nen wie beispielsweise Untertitel und abgespielte Tracks. Das **Interval**-Konzept stammt von der Timeline Ontology¹⁶. Die Ontologien in dem Netzwerk, wie es in Abbildung 16.7 dargestellt ist, sind hinsichtlich ihrer Größe und formalen Beschreibung sehr homogen. Dies muss jedoch nicht so sein. So können die Ontologien auch sehr verschieden groß sein oder können sehr formal definiert sein oder auch nicht. Insgesamt lassen sich Ontologien auf Grund unterschiedlicher nicht-funktionaler Eigenschaften in drei verschiedene Arten unterscheiden [56]. Diese verschiedenen Arten von Ontologien werden im folgenden Abschnitt eingeführt, bevor wir das oben genannte Beispiel noch einmal und zusammen mit diesem Hintergrundwissen betrachten.

16.4.2 Verschiedene Arten von Ontologien

Ein Netzwerk von Ontologien, wie das in Abbildung 16.7 dargestellte Beispiel, kann aus einer Vielzahl von Ontologien bestehen, die von unterschiedlichen Akteuren und Communities erstellt wurden. Ontologien können das Ergebnis einer Transformation oder einer Reengineering-Tätigkeit eines Altsystems sein, wie beispielsweise einer relationalen Datenbank oder existierender Taxonomie wie zum Beispiel der Dewey Decimal Classification¹⁷ oder Dublin Core. Andere Ontologien werden von Grund auf neu erstellt. Dabei werden existierende Methoden und Werkzeuge zum Ontologie-Engineering angewendet und eine geeignete Repräsentationssprache für die Ontologie wird gewählt (siehe Abschnitt 16.5). Ontologien können sehr einfach sein, wie die genannte FOAF-Ontologie oder Event-Ontologie, oder sehr komplex und umfangreich, da sie von Domänenexperten entwickelt wurden, wie die medizinische Ontologie SNOMED. Ontologie-Engineering beschäftigt sich also mit Methoden zur Erstellung von Ontologien [28] und hat seinen Ursprung im Software-Engineering in der Erstellung von Domänenmodellen und im Datenbankentwurf in der Erstellung von konzeptuellen Modellen. Eine gute Übersicht zum Thema Ontologie-Engineering ist in verschiedenen Referenzbüchern zu finden [28, 58]. Ontologien unterscheiden sich stark in ihrer Struktur, Größe, angewendeten Entwicklungsmethode und betrachteten Anwendungsbereich. Komplexe Ontologien werden zudem hinsichtlich ihres Zwecks und ihrer Granularität unterschieden:

¹⁶ <http://motools.sourceforge.net/timeline>.

¹⁷ <http://dewey.info/>.

Domänenontologien

wie SNOMED stellen die Repräsentationen von Wissen dar, das spezifisch ist für eine bestimmte Domäne [17, 43]. Domänenontologien werden als externe Quellen von Hintergrundwissen verwendet [17]. Sie können auf Basisontologien [44] oder Kernontologien [54] aufbauen, die der Domänenontologie Strukturierungen vorgeben und damit die Interoperabilität zwischen verschiedenen Domänenontologien verbessern.

Kernontologien

stellen eine präzise Definition strukturierter Wissens in einem bestimmten Bereich dar, der sich über mehrere Anwendungsdomänen hin erstreckt [43, 56]. Beispiele für Kernontologien sind die Kernontologie für Software-Komponenten und Web-Services [43], für Ereignisse und Ereignisbeziehungen [55], für persönliches Informationsmanagement [22] oder für Multimedia Metadaten [49]. Kernontologien sollten dabei auf Basisontologien aufsetzen, um von deren Formalisierung und starker Axiomatisierung zu profitieren [56]. Dazu werden in Kernontologien neue Konzepte und Relationen für die betrachtete Anwendungsdomäne hinzugefügt und von den Basisontologien spezialisiert.

Basisontologien

haben einen sehr breiten Anwendungsbereich und können in den verschiedensten Modellierungsszenarien wiederverwendet werden [10]. Sie dienen daher zu Referenzzwecken [43] und haben zum Ziel die allgemeinsten und generischen Konzepte und Relationen zu modellieren mit denen fast beliebige Aspekte unserer Welt beschrieben werden können [10, 43], wie beispielsweise Objekte und Ereignisse. Ein Beispiel ist die Basisontologie Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [10]. Basisontologien haben eine reichhaltige Axiomatisierung, die zum Entwicklungszeitpunkt von Ontologien wichtig ist. Sie helfen dem Ontologie-Entwickler eine formale und in sich konsistente Konzeptualisierung des betrachteten Ausschnitts der Welt, die zu modellieren und auf Konsistenz zu überprüfen ist. Für die spätere Anwendung von Basisontologien in einer konkreten Anwendung, das heißt während der Laufzeit einer Anwendung, kann die reichhaltige Axiomatisierung oft entfernt und durch eine leichtgewichtigere Version der Basisontologie ersetzt werden.

Im Gegensatz dazu werden Domänenontologien spezifisch dafür gebaut, um zur Laufzeit automatische Schlussfolgerungen ziehen zu können. Daher ist beim Entwurf und der Entwicklung von Ontologien immer die Vollständigkeit und Komplexität auf der einen Seite mit der Effizienz auf der anderen Seite abzuwegen. Um strukturiertes Wissen wie das in Abbildung 16.7 dargestellte Szenario abzubilden werden vernetzte Ontologien benötigt, die in einem Netzwerk über das Internet aufgespannt werden. Dazu müssen die verwendeten Ontologien zueinander passen und abgeglichen werden.

16.4.3 Verteiltes Netzwerk von Ontologien im Web

Ein Netzwerk von Ontologien muss hinsichtlich der ihr auferlegten funktionalen Anforderungen flexibel sein. Dies liegt daran, dass Systeme über die Zeit hin verändert, erweitert, kombiniert oder integriert werden. Zudem müssen die vernetzten Ontologien zu einem gemeinsamen Verständnis der modellierten Domäne führen. Dieses gemeinsame Verständnis kann durch ein ausreichendes Maß an Formalisierung und Axiomatisierung sowie durch Verwendung von Ontologiemustern erzielt werden. Ontologiemuster erlauben, Teile aus der Originalontologie

auszuwählen und entweder alle oder nur bestimmte Teile der Ontologie im Netzwerk wiederzuverwenden. Um ein Netzwerk von Ontologien zu schaffen, können also beispielsweise bereits existierende Ontologien im Web zusammengeführt werden. Auf der anderen Seite kann der Ontologieentwickler auch die Modularisierung von Ontologien mit Hilfe von Ontologiemustern vorantreiben beziehungsweise explizit vorsehen. Einen Ansatz um ein Netzwerk von Ontologien zu entwerfen stellen die Kernontologien dar. Sie erlauben strukturiertes Wissen in komplexen Domänen zu erfassen und auszutauschen. Wohldefinierte Kernontologien erfüllen die im vorangegangenen Abschnitt genannten Eigenschaften und ermöglichen eine einfache Integration und ein reibungsloses Zusammenspiel der Ontologien (siehe auch [56]). Der Ansatz der vernetzten Ontologien führt zu einer flachen Struktur, wie in Abbildung 16.7 dargestellt, bei der alle verwendeten Ontologien auf derselben Ebene verweilen. Solche Strukturen lassen sich bis zu einem gewissen Grad an Komplexität beherrschen.

Der Ansatz der vernetzten Kernontologien wird am Beispiel von Ontologieschichten beginnend bei Basis- über Kern- zu Domänenontologien veranschaulicht. Wie in Abbildung 16.8 dargestellt, ist DOLCE als Basisontologie auf der unteren Schicht, die Multimedia Metadata Ontology (M3O) [50] als Kernontologie für Multimedia-Metadaten und eine Erweiterung der M3O für die Musikdomäne.

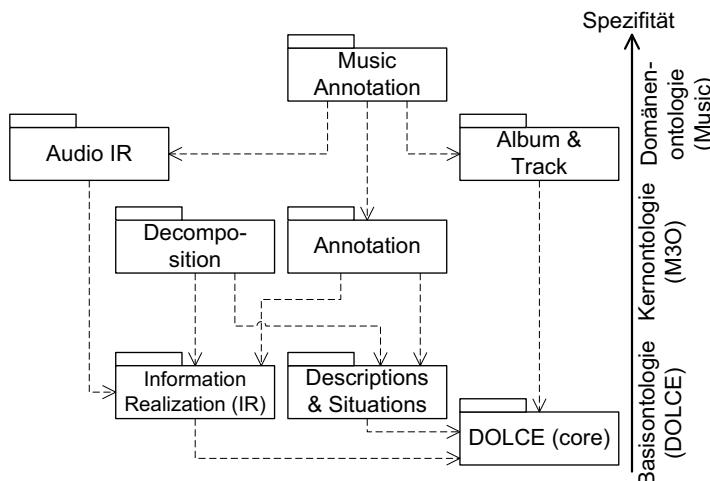


Abbildung 16.8: Ontologieschichten mit der Kombination von DOLCE, M3O, domänen spezifische Erweiterungen der M3O zur Annotation von Audio-Daten und Musik und eine Domänen-ontologie für Alben und Tracks.

Kernontologien sind typischerweise groß und decken mit ihrer Wissensmodellierung einen Bereich ab, der vielleicht größer ist als es die spezifische Anwendungsdomäne erfordert [23]. Konkrete Informationssysteme werden typischerweise nur einen Teil der Kernontologien nutzen. Um eine Modularisierung von Kernontologien zu erreichen, sollten Sie mit Hilfe von Ontologiemustern entworfen sein. Ein Ontologiemuster stellt ähnlich wie ein Entwurfsmuster in der Software-Technik eine generische Lösung für ein wiederkehrendes Modellierungsproblem dar. Durch eine präzise Abstimmung der Konzepte in der Kernontologie mit den angebotenen Konzepten der Basisontologie stellen sie eine solide Basis für zukünftige Erweiterungen dar. Neue Muster können hinzugefügt werden und existierende Muster können durch Spezialisierung der

Konzepte und Rollen erweitert werden. Abbildung 16.8 zeigt verschiedene Muster der M3O und DOLCE Ontologien. Im Idealfall werden die Ontologiemuster der Kernontologien in den Domänenontologien wiederverwendet [23] wie in Abbildung 16.8 dargestellt. Da jedoch nicht davon ausgegangen werden kann, dass alle Domänenontologien mit einer Basis- oder Kernontologie abgestimmt sind, muss auch die Option berücksichtigt werden, dass Domänenontologien unabhängig davon entwickelt und gepflegt werden. In diesem Fall kann Domänenwissen in Kernontologien durch die Anwendung des Ontologiemusters Descriptions and Situations (DnS) der Basisontologie DOLCE wiederverwendet werden. Das Ontologiemuster DnS ist eine ontologische Formalisierung von Kontext [43] durch die Definition verschiedener Sichten mittels Rollen. Diese Rollen können sich auf Domänenontologien beziehen und erlauben eine klare Trennung des strukturierten Wissens der Kernontologie und domänenspezifischen Wissens. Wir erwarten, dass in Zukunft weitere Ontologie-Entwurfsmuster und Kernontologien entstehen. Beispielsweise könnte das Musikszenario in Abschnitt 16.1 von einer Kernontologie für das Medienmanagement profitieren.

Zur Modellierung eines Netzwerkes von Ontologien, wie das oben beschriebene Beispiel, wird oftmals die Web Ontology Language (OWL) und deren Möglichkeit zur Axiomatisierung mittels Beschreibungslogik [4] verwendet. Neben der Verwendung zur Modellierung einer verteilten Wissensrepräsentation und -integration wird OWL wie in Abschnitt 16.5 beschrieben insbesondere auch verwendet, um Schlussfolgerungen mittels Inferenz aus diesem Wissen abzuleiten.

16.5 Inferenz im Web

In Abschnitt 16.2 wurden verschiedene formale Sprachen zur Wissensrepräsentation im Semantic Web vorgestellt. RDF ermöglicht die Beschreibung einfacher Fakten (Aussagen mit Subjekt, Prädikat und Objekt, sogenannte RDF-Tripel), z. B. „Anni-Frid Lyngstad“ „ist Mitglied von“ „ABBA“. Entitäten werden mit benannten Beziehungen verknüpft. Die Menge von solchen Verknüpfungen bildet einen gerichteten Graphen. RDFS ermöglicht die Definition von Typen von Entitäten (Klassen), Beziehungen zwischen Klassen und eine Sub- und Superklassenhierarchie zwischen Typen. OWL ist noch ausdrucksstärker als RDF und RDFS. OWL erlaubt beispielsweise die Definition von disjunkten Klassen (Begriffen) oder die Beschreibung von Klassen in Form von Schnitt, Vereinigung und Komplement anderer Klassen (vgl. Abschnitt 16.4).

Basierend auf diesen formalen Sprachen und deren Semantik können durch deduktive Inferenz weitere (implizite) Fakten aus der Wissensbasis abgeleitet werden. Im Folgenden wird beispielhaft die Herleitung von impliziten Fakten aus einer Menge von explizit gegebenen Fakten mittels des RDFS-Konstrukts `rdfs:subClassOf` und des OWL-Konstrukts `owl:sameAs` dargestellt. `rdfs:subClassOf` beschreibt hierarchische Beziehungen zwischen Klassen und mit `owl:sameAs` können zwei Ressourcen als identisch definiert werden.

Als erstes Beispiel betrachten wir die Klasse `foaf:Person`, welche in der FOAF-Ontologie definiert ist, und die Klassen `mo:Musician` und `mo:Group`, die in der Musikontologie definiert sind. In der Musikontologie gibt es zusätzlich ein Axiom, welches `mo:Musician` als Subklasse von `foaf:Person` mittels `rdfs:subClassOf` definiert. Aufgrund dieses Axioms kann durch deduktive Inferenz hergeleitet werden, dass Instanzen von `mo:Musician` auch Instanzen von `foaf:Person` sind. Falls es nun eine solche Hierarchie von Klassen gibt und

zusätzlich noch eine Aussage das Anni-Frid Lyngstad vom Typ `mo:Musician` ist, so kann mittels Inferenz hergeleitet werden, dass Anni-Frid Lyngstad auch vom Typ `foaf:Person` ist. Dies bedeutet, dass alle Anfragen die nach Entitäten vom Typ `foaf:Person` fragen auch Anni-Frid Lyngstad im Anfrageergebnis enthalten, auch wenn diese Instanz nicht explizit als Instanz von `foaf:Person` definiert ist. Abbildung 16.9 stellt diese Fakten und die entsprechende Klassenhierarchie in RDFS als gerichteten Graph dar.

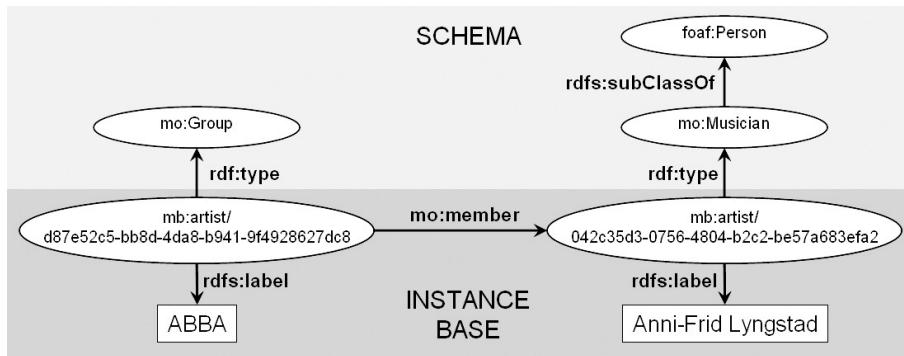


Abbildung 16.9: Visualisierung der RDF-Beispieldaten über ABBA und Anni-Frid Lyngstad.

Im zweiten Beispiel werden mittels des OWL-Konstrukts `owl:sameAs` zwei Ressourcen als identisch definiert, z.B. <http://www.bbc.co.uk/music/artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist> und <http://dbpedia.org/resource/ABBA>. Durch Schlussfolgerung können Informationen über ABBA aus verschiedenen Quellen verbunden werden (vgl. Abschnitt 16.3.3). Da Ontologien im Web unabhängig voneinander erstellt werden und URIs lokalen Namenskonventionen unterliegen, ist es durchaus möglich, dass ein reales Objekt durch verschiedene URIs (in verschiedenen Ontologien) referenziert wird.

OWL bietet noch eine Vielzahl weiterer Konstrukte zur Beschreibung von Klassen, Beziehungen und konkreten Fakten. OWL ermöglicht beispielsweise die Definition von transitiven Beziehungen und inversen Beziehungen (z.B. „ist-Mitglied“ ist invers zu „hat-Mitglieder“). Für OWL-Ontologien gibt es Schlussfolgerungsdienste, die unter anderem Konsistenzprüfung einer Ontologie oder die Überprüfung der Erfüllbarkeit von Klassen ermöglichen. Ein Klasse ist erfüllbar, wenn es Instanzen dieser Klasse geben kann.

16.6 Identität und Verknüpfung von Objekten und Begriffen

Im Semantic Web kann nicht die Annahme getroffen werden, dass zwei URIs auf zwei verschiedene Entitäten verweisen. Eine URI hat von sich aus, beziehungsweise in sich, keine Identität [30]. Vielmehr wird die Identität beziehungsweise Interpretation einer URI durch den Kontext, in dem sie im Semantic Web verwendet wird, deutlich. Zu bestimmen, ob zwei URIs auf dieselbe Entität verweisen oder nicht, ist keine einfache Aufgabe und wurde in der Vergangenheit intensiv im Data Mining und im Sprachverstehen untersucht. Um zu erkennen, ob sich die

Autorennamen auf Forschungsbeiträgen auf dieselbe Person beziehen oder nicht, ist es oftmals nicht ausreichend den Namen, den Veranstaltungsort, Titel und Koautoren aufzulösen und zu betrachten [41].

Der Vorgang zur Bestimmung der Identität einer Ressource wird oftmals als Entitätenauflösung (englisch entity resolution) [41], Koreferenzauflösung [63], Objektidentifikation [48] und Normalisierung [63, 64] bezeichnet. Die korrekte Bestimmung der Identität von Entitäten im Internet wird zunehmend wichtiger, da immer mehr Datensätze im Internet erscheinen und dies eine signifikante Hürde für sehr große Semantic Web Anwendungen darstellt [25].

Um dem gerecht zu werden, existieren eine Reihe von Diensten, die Entitäten erkennen und ihre Identität bestimmen können: Thomson Reuters bietet mit OpenCalais¹⁸ einen Dienst an, mit dem Text in natürlicher Sprache mittels Erkennung von Entitäten mit anderen Ressourcen verknüpft werden kann. Ziel des OKKAM-Projekt¹⁹ ist die Entwicklung von skalierbaren Systemen zur Erkennung von Entitäten im Internet wie beispielsweise Personen, Orte, Organisationen und Ereignisse und diese mit anderen Entitäten im Internet zu verknüpfen. Der Dienst sameAs²⁰ zielt auf die Erkennung von doppelten Ressourcen auf dem Semantic Web unter Verwendung der Rolle `owl:sameAs` ab. Damit können Koreferenzen zwischen verschiedenen Datensätzen aufgelöst werden, zum Beispiel wird für die Anfrage mit der URI `http://dbpedia.org/resource/ABBA` eine Liste von 21 Ressourcen zurückgegeben, die ebenfalls auf die Musikgruppe ABBA verweisen. Eine davon ist die BBC mit der Ressource `http://www.bbc.co.uk/music/artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist`.

Weiterhin ist das Problem des Schema-Matching [64] sehr verwandt mit dem der Auflösung von Entitäten, Koreferenzauflösung und Normalisierung. Ziel von Schema-Matching ist die, selbst für kleine Schemata nicht-triviale Frage, wie Daten integriert werden können [64]. Im Semantic Web bedeutet Schema-Matching der Abgleich von verschiedenen Ontologien beziehungsweise die in den diesen Ontologien definierten Konzepte. Verschiedene (halb-)automatische Verfahren oder Verfahren des maschinellen Lernen zum Abgleich von Ontologien wurden in der Vergangenheit entwickelt [7, 16, 18]. Kernontologien wie in Abbildung 16.4.2 stellen generische Modellierungsframeworks zur Integration und Abgleich mit anderer Ontologien dar. Zudem können Kernontologien auch Linked Open Data integrieren, das typischerweise keine oder nur sehr wenige Schema-Informationen beinhaltet. Die YAGO-Ontology [61] wurde generiert aus der Verschmelzung von Wikipedia und Wordnet unter Verwendung von regelbasierten und heuristischen Methoden. Eine manuelle Evaluation konnte eine Genauigkeit von 95% nachweisen. Ein manueller Abgleich von verschiedenen Datenquellen wird auch im Linked Open Data Projekt der Deutschen Nationalbibliothek verfolgt²¹. Beispielsweise wurde die Datenbank mit den Autoren aller in Deutschland publizierten Dokumente händisch mit der DBpedia und anderen Datenquellen verknüpft. Eine besondere Herausforderung war dabei die Identität der Autoren wie oben beschrieben zu identifizieren. Zum Beispiel hat der frühere Bundeskanzler Helmut Kohl einen Schiedsrichter als Namensvetter, dessen Arbeiten nicht mit dem DBpedia-Eintrag des Kanzlers verknüpft werden sollten. Beziehungen zwischen Schlüsselwörtern zur Beschreibung von Publikationen werden mit dem SKOS-Vokabular beschrieben. Beispielsweise werden Schlüsselwörter über die Relation `skos:related` zueinander in Beziehung gesetzt. Hypo-

¹⁸ <http://www.opencalais.com/>.

¹⁹ <http://www.okkam.org/>.

²⁰ <http://sameas.org/>.

²¹ <http://www.d-nb.de/>.

nyme und Hypernyme werden durch die Relationen `skos:narrower` und `skos:broader` ausgedrückt. Schließlich sei die Ontology Alignment Evaluation Initiative²² erwähnt, die zum Ziel hat, einen etablierten Konsensus zur Evaluation von Methoden des Ontologie-Abgleichs zu erreichen.

16.7 Herkunft und Vertrauenswürdigkeit von Daten

Vertrauenswürdigkeit von Web-Seiten und Daten im Web kann anhand verschiedener Indikatoren erkannt werden, z.B. durch Zertifikate, anhand der Platzierung von Ergebnissen von Suchmaschinen, und über Links (Forward- und Backwardlinks) zu anderen Seiten. Allerdings gibt es im Semantic Web für Benutzer nur wenig Möglichkeiten, die Vertrauenswürdigkeit von einzelnen Daten zu bewerten.

Vertrauenswürdigkeit von Daten im Web kann von der Vertrauenswürdigkeit anderer Benutzer („Wer sagt das?“), der zeitlichen Gültigkeit von Fakten („Wann wurde ein Fakt beschrieben?“) oder in Bezug auf Unsicherheit von Angaben („Zu welchem Grad ist die Aussage wahr?“) abgeleitet werden. Artz et Gil [2] fassen Vertrauenswürdigkeit wie folgt zusammen: „Vertrauenswürdigkeit von Daten ist kein neues Forschungsgebiet der Informatik, sondern sie existiert bereits in verschiedenen Bereichen der Informatik, z. B. in Form von Sicherheit und Zugriffskontrolle in Netzwerken, Zuverlässigkeit in verteilten Systemen, in Agentensystemen, und bei Richtlinien und Regeln zur Entscheidungsfindung unter Unsicherheit. Vertrauenswürdigkeit wird in jedem dieser Bereiche unterschiedlich behandelt.“

Obwohl Vertrauenswürdigkeit in diesen Bereichen schon lange betrachtet wird, bringt die Bereitstellung und Veröffentlichung von Daten durch viele Benutzer an verschiedenen Quellen im Semantic Web neue und einzigartige Herausforderungen mit sich. Des Weiteren spielt Vertrauenswürdigkeit auch für Schlussfolgerungsdienste im Semantic Web eine Rolle, da bei der Herleitung von Daten Angaben bezüglich der Vertrauenswürdigkeit zu beachten sind und Daten nach ihrer Vertrauenswürdigkeit zu bewerten sind. Wichtige Aspekte für Vertrauenswürdigkeit von Daten sind unter anderem: (i) die Herkunft von Daten, (ii) das Vertrauen, das anhand vorheriger Interaktionen bereits gewonnen wurde, (iii) Bewertungen, die durch Richtlinien eines Systems zugewiesen wurden, und (iv) Zugriffskontrollen und teilweise auch Sicherheit und Wichtigkeit von Informationen.

Diese Aspekte werden in verschiedenen Systemen realisiert. Datenherkunft und Vertrauenswürdigkeit von Daten im Semantic Web wurde für RDF-Daten in [14, 20] und für OWL und Regeln in [14] behandelt. Vertrauen in sozialen und semantischen Netzwerken wird in [26, 27] betrachtet, und im Zusammenhang von Richtlinien in Semantic Web Anwendungen in [9, 57] erläutert. Andere Arbeiten befassen sich mit Zugriffskontrolle über verteilten Daten im Semantic Web [24]. Schließlich gibt es noch Ansätze zur Berechnung von Vertrauenswerten in [59], und zur Beurteilung der Relevanz von Datenquellen in [21] und von Teilgraphen in [42].

²² <http://oaei.ontologymatching.org/>

16.8 Semantic Web Anwendungen und Benutzerschnittstellen

Mit der zunehmenden Verbreitung und Verwendung von semantischen und verknüpften Daten im Web sind gleichzeitig neben den Anforderungen an Semantic Web Anwendungen auch deren Anwendungsmöglichkeiten gestiegen. Die Anforderungen sind anhand der Daten im Web gegeben. Anwendungen, die Daten aus relationalen Datenbanken oder XML-Dokumenten verwenden, können von einem festgelegten Schema ausgehen. Dies kann allerdings bei Daten im Web nicht vorausgesetzt werden. Oft sind weder die Datenquellen noch die Art und Menge der Daten einer Quelle vollständig bekannt.

Die Dynamik von semantischen Daten im Web muss von Anwendungen entsprechend berücksichtigt werden, sowohl bei der Anfrage und Aggregation von Daten, als auch bei der Visualisierung von Daten. Die eigentliche Herausforderung von Semantic Web Anwendungen liegt darin, eine bestmögliche Flexibilität der Anwendung zu garantieren, um die Dynamik von Datenquellen, Daten, und Schemas bei der Eingabe, Verarbeitung und Ausgabe zu berücksichtigen.

Nachfolgend werden vier Beispiele von Semantic Web Anwendungen bzw. Anwendungsberächen vorgestellt. Sie verdeutlichen wie Flexibilität und Qualität der Suche, Integration, Aggregation und Darstellung von Daten aus dem Web realisiert werden kann. Sie zeigen zugleich das Potential von Semantic Web Anwendungen. Zunächst werden einheitliche Vokabulare und Schemas am Beispiel von *schema.org* vorgestellt. Sie dienen als Grundlage für eine semantische Suche, um Suchmaschinen Informationen über die Bedeutung von Inhalten von Web-Dokumenten zu geben. Die Suche und Integration von Daten aus verschiedenen Quellen wird von *Sig.ma* unterstützt. *Sig.ma* ist ein Semantic Web Browser. Andere Anwendungen realisieren semantische Suche durch weitere Repräsentationsformalismen (z.B. *Knowledge Graph*). Anschließend wird die *Facebook Graph-API*, eine Programmierschnittstelle zum Facebook-Graph, kurz vorgestellt. Schließlich wird *SemaPlorer* zur Visualisierung von heterogenen und verteilten Daten betrachtet. *SemaPlorer* verwendet verschiedene Facetten zur Suche, wobei eine Facette im Wesentlichen ein Kriterium zur Aufteilung einer Datenmenge ist [51].

16.8.1 Vokabulare und Schemas

In HTML-Dokumenten kann die Struktur und der Aufbau von Seiten mit Tags beschrieben werden, nicht aber die Bedeutung der Informationen. Vokabulare, Schemas und Mikrodaten können als Mark-up in HTML-Dokumenten verwendet werden, um Angaben über Seiteninhalte und deren Bedeutung so zu beschreiben, dass Suchmaschinen diese Information verarbeiten können. *Schema.org*²³ ist eine Sammlung von Vokabularen und Schemas um HTML-Seiten mit zusätzlicher Information anzureichern.

Das Vokabular von *Schema.org* beinhaltet eine Menge von Entitäten und deren Eigenschaften. Eine universelle Entität „Thing“ ist die allgemeinste Entität, die eine Art Oberbegriff aller Entitäten ist. Weitere geläufige Entitäten sind *Organization*, *Person*, *Event* und *Place*. Eigenschaften werden zur genaueren Beschreibung von Entitäten verwendet. Z.B. hat eine Person (Entität *Person*) Eigenschaften wie Name, Adresse und Geburtsdatum.

²³ <http://schema.org>.

Neben Vokabularen wird in Schema.org auch die Anwendung von HTML-Mikrodaten festgelegt, mit dem Ziel, Daten in HTML-Dokumenten in einer möglichst eindeutigen Form darzustellen, so dass Suchmaschinen diese richtig interpretieren können. Ein Beispiel hierfür sind Formate für eindeutige Datum- und Zeitangaben, die auch Intervalle zur Angabe der Dauer von Ereignissen beschreiben können.

Unterstützt wird Schema.org unter anderem von den Suchmaschinen Bing, Google und Yandex. Es gibt Erweiterungen und Bibliotheken für verschiedene Programmiersprachen, u.a. für PHP, JavaScript, Ruby und Python, um Webseiten und Webanwendungen mit Vokabularen und Mikrodaten von Schema.org zu erstellen. Ebenso gibt es Abbildungen von Vokabularen und Mikrodaten aus Schema.org zu RDFS.

16.8.2 Semantic Web Browser und Semantische Suche

Ein Web Browser ermöglicht die Darstellung von Web-Seiten. Ein Semantic Web Browser geht noch einen Schritt weiter, indem zusätzlich noch die zugrundeliegende Informationen einzelner Seiten, die z.B. in Form von RDF-Metadaten vorliegen, dem Anwender visualisiert werden können. Somit sind gewöhnliche Nutzer in der Lage, Semantic Web Daten für ihre Informati-onssuche zu verwenden und auszunutzen.

Sig.ma [62] ist eine Anwendung zum (Durch-)Suchen von Semantic Web Daten, die aus mehreren verteilten Datenquellen stammen können. Sig.ma stellt eine API zur automatischen Integration von mehreren Datenquellen im Web zur Verfügung. Die angefragten Datenquellen beschreiben Informationen in RDF. Eine Suche in Sig.ma wird durch eine textuelle Anfrage vom Anwender gestartet. Dabei kann nach Entitäten wie Personen, Orte oder Produkten gesucht werden. Ergebnisse einer Anfrage werden in aggregierter Form (Profil einer Entität) dargestellt, d.h. Eigenschaften der gesuchten Entität, z.B. einer Person, werden aus verschiedenen Datenquel-len zusammengefasst dargestellt. Beispielsweise können bei einer Personensuche Informationen wie E-mail-Adresse, Anschrift oder aktueller Arbeitgeber angezeigt werden. Neben den eigent-lichen Informationen werden auch Links zu den zugrundeliegenden Datenquellen angezeigt, um Anwendern eine Navigation zur Verfeinerung ihrer Suche zu ermöglichen. Sig.ma unterstützt auch strukturierte Anfragen, in denen zu einer Entität bestimmte Merkmale angefragt werden können, z.B. Kontaktdaten einer bestimmten Person.

Anfragen an Datenquellen erfolgen parallel. Die Ergebnisse aus den einzelnen Datenquellen in Form von RDF-Graphen werden zusammengefasst, indem Eigenschaften von Links in RDF-Daten, wie beispielsweise *owl:sameAs* oder invers-funktionale Prädikate, verwendet werden. Bei der Suche in Datenquellen werden Techniken wie Indexe, logische Inferenz und Heuristiken zur Datenaggregation verwendet.

Watson²⁴ ist ein Programm von IBM um Fragen, die in natürlicher Sprache gestellt sind, zu beantworten. Watson verwendet eine Vielzahl von Algorithmen, Techniken zur Verarbeitung von natürlichen Sprachen, Methoden aus dem Information Retrieval und Machine Learning, aber auch Wissensrepräsentation und Inferenz.

Google bietet mit Google Knowledge Graph²⁵ eine semantische Suchfunktion. Die englisch-sprachige Version der Suchmaschine ist um den *Knowledge Graph* erweitert. Ein Knowledge

²⁴ <http://www-03.ibm.com/innovation/us/watson/index.html>.

²⁵ <http://www.google.com/insidesearch/features/search/knowledge.html>.

Graph ist ein Graph, in dem Entitäten (als Knoten) miteinander verknüpft sind, wobei diese Verknüpfungen Beziehungen zwischen den Entitäten darstellen. Tritt nun ein Suchbegriff in einer Anfrage auf, so wird nach der entsprechenden Entität im Knowledge Graph gesucht. Ausgehend von dieser Entität (Knoten) kann dann mittels der Verknüpfungen zu weiteren Entitäten navigiert werden.

16.8.3 Zugriff auf soziale Netzwerke

Ein soziales Netzwerk ist im Wesentlichen ein Graph, in dem Verbindungen von Benutzern zu anderen Benutzern (z.B. in Form einer Freundschaftsbeziehung) oder zu Ereignissen und Gruppen existieren. Die Graph-API von Facebook beschreibt eine Programmierschnittstelle zum Facebook-Graph (genannt *Open Graph*). Innerhalb des Graphen werden Personen, Ereignisse, Seiten und Fotos als Objekte dargestellt, wobei jedes Objekt einen eindeutigen Bezeichner hat, z.B. ist <https://graph.facebook.com/abba> der Bezeichner der Facebook-Seite von ABBA. Für die möglichen Beziehungsarten eines Objekts gibt es ebenfalls eindeutige Bezeichner, die das Navigieren von einem Objekt zu allen verbundenen Objekten bezüglich einer bestimmten Beziehung ermöglichen.

Die Graph-API ermöglicht neben dem Navigieren im Facebook-Graph und dem Lesen von Objekten, einschließlich deren Eigenschaften und Beziehungen zu anderen Objekten, auch das Erstellen von neuen Objekten im Facebook-Graph und das Bereitstellen von Applikationen. Die API unterstützt ebenfalls Anfragen von Metadaten eines Objekts, wie beispielsweise *wann* und *von wem* ein Objekt erstellt wurde.

16.8.4 Visualisierung semantisch heterogener und verteilter Daten

Neben der Bereitstellung und Suche im Web ist die Entwicklung flexibler Benutzerschnittstellen zur interaktiven Visualisierung von verteilten, semantischen Daten ein weiterer wichtiger Aspekt. Ein Beispiel einer solchen Anwendung ist die interaktive Anwendung SemaPlorer [52] zur facettierten Suche und Navigation. Facetten sind insbesondere bei der Visualisierung von komplexen und großen Datenmengen hilfreich, z.B. SMILE Timeline Widget [39] für zeitliche Information und Google Maps²⁶ für räumliche Informationen. SemaPlorer ermöglicht ein interaktives Durchsuchen und Visualisieren von sehr großen und heterogenen semantischen Daten in Echtzeit entlang verschiedener Facetten. Die von SemaPlorer benutzten Datenquellen sind DBpedia, GeoNames²⁷, WordNet²⁸ und persönliche FOAF-Dateien. Zusätzlich wird ein Live-Wrapper zur Flickr-API²⁹ verwendet. SemaPlorer hat vier Facetten zur Suche und Visualisierung anhand von Orten, Personen, Tags und Zeit. Ein Screenshot von SemaPlorer ist in Abbildung 16.10 dargestellt. Textuelle Anfragen können auf der linken Seite eingegeben werden. Das entsprechende Anfrageergebnis kann über eine geeignete Facette betrachtet werden. Der Inhalt der aktuellen Facette, wie beispielsweise Orte und ortsbezogenen Informationen, ist auf der rechten Seite zu sehen. In der Mitte sind Visualisierungen und Bilder eines bestimmten Ortes zu sehen.

²⁶ <http://maps.google.com>.

²⁷ <http://geonames.org>.

²⁸ <http://wordnet.princeton.edu>.

²⁹ <http://flickr.com>.

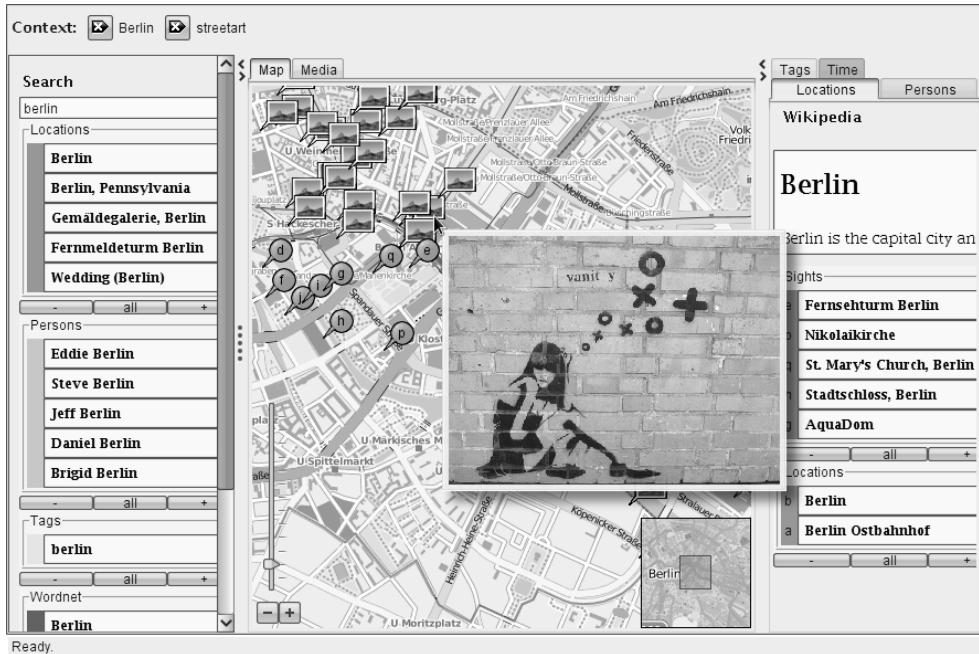


Abbildung 16.10: Facettierte Suche und Navigation von Semantischen Daten in SemaPlorer [52].

Bezüglich der Systemarchitektur ist SemaPlorer eine sehr flexible und generisch implementierte Anwendung. Allerdings sind die verschiedenen Facetten und die Daten, die in den jeweiligen Facetten bearbeitet und visualisiert werden können, fest in die Applikation eingebunden. Ähnlich zu vielen anderen Semantic Web Anwendungen sind die verwendeten Datenquellen direkt mit den einzelnen Bestandteilen der Applikation verbunden. Ein weiteres Beispiel für eine Anwendung mit flexiblem Benutzungsinterface ist Paggr³⁰. Paggr verwendet strukturierte, selbstbeschreibende Daten auf dem Web um ad-hoc semantische Mashups zu erzeugen und in einem personalisierten Dashboard anzuzeigen. Schließlich bietet der RDF-Browser Lena [21] eine flexible Darstellung von RDF-Daten mit Hilfe des Fresnel Display Vocabulary³¹.

16.9 Zusammenfassung und Ausblick

Das Semantic Web besteht aus einer Vielzahl von Techniken, die stark von der Langzeitforschung der Künstlichen Intelligenz und deren Ergebnissen beeinflusst wurden. Methoden aus der Künstlichen Intelligenz zur Modellierung, Darstellung und Inferenz werden um verteiltes Wissen und verteilte Wissensrepräsentation erweitert. Ausgehend von dem Web in seiner aktueller Form beinhaltet das Semantic Web weitere Standards und Sprachen um die Semantik von Dokumenten und Daten in maschinenverarbeitbarer Form darzustellen. Das volle Potential des Semantic Webs wurde allerdings noch nicht vollständig ausgenutzt, da vor allem einige wichti-

³⁰ <http://www.paggr.com/>

³¹ <http://www.w3.org/2005/04/fresnel-info/>

ge Komponenten der Semantic Web Architektur noch erforscht werden, z.B. die Datenherkunft und Vertrauenswürdigkeit.

Allerdings gewinnt das Semantic Web stetig an Bedeutung. In den letzten Jahren hat die Veröffentlichung von Linked Data enorm zugenommen, insbesondere in den Bereichen bibliographischer Informationsverwaltung, Bioinformatik und E-Government. DBpedia ist dabei die Kern-datenquelle, um die verschiedenen Bereiche gruppiert sind (vgl. [6]). Dies wird beispielsweise an dem enormen Wachstum der Linked Open Data Cloud³² seit 2007 deutlich. Dieses schnelle Wachstum hat schon Einfluss auf die Industrie. Mit Schema.org werden Schemata zur ausführlicheren Beschreibung von Daten auf Webseiten definiert, um Informationen über die zugrundeliegende Datenstrukturen und die Bedeutung der Daten zu geben. Suchmaschinen können diese zusätzliche Information nutzen, um die Inhalte von Webseiten besser analysieren zu können. Schema.org wird von den Suchmaschinen Bing, Google und Yandex unterstützt.

Studien auf ausgewählten Quellen haben gezeigt, dass Webseiten unter den Top-10 Ergebnissen eine um bis zu 15 % höhere Klickrate haben³³. Andere Unternehmen wie BestBuy.com berichten sogar von bis zu 30 % höheren Zugriffsraten, seit der Erweiterung ihrer Webseiten mit semantischen Daten (vgl. Abschnitt 16.8) in 2009. BestBuy.com benutzt das GoodRelations Vokabular³⁴ um Online-Angebote zu beschreiben. Ebenso hat Google begonnen semantische Daten von Online-Handelsportalen, die das GoodRelations Vokabular benutzen, bei der Suche zu berücksichtigen³⁵.

Ein weiterer Erfolg ist die Veröffentlichung von Regierungsdaten. Zum Beispiel stellt die US-Regierung mit Data.gov³⁶ Regierungsdaten öffentlich bereit, und US Census³⁷ veröffentlicht statistische Daten über die USA. In Großbritannien ist data.gov.uk³⁸ ein wesentlicher Teil eines Programms zu mehr Transparenz von Daten im öffentlichen Sektor. Auch in Deutschland werden zunehmend offene Daten frei zur Verfügung gestellt. Eine Übersicht über offene Daten in Deutschland ist u.a. im Katalog für offene Daten unter <http://de.ckan.net> dargestellt.

Schließlich kann ein starkes Wachstum von semantischen Daten der Biomedizin im Web festgestellt werden. Im Rahmen von Bio2RDF³⁹ wurden viele Datenbanken der Bioinformatik miteinander verknüpft. Die Transinsight GmbH bietet die wissensbasierte Suchmaschine GoPubMed⁴⁰ an, um Forschungsartikel der Biomedizin zu finden. Ontologien werden zur Suche verwendet.

Zusammenfassend kann beobachtet werden, dass semantische Daten im Web einen echten Einfluss auf kommerzielle Anbieter von Produkten und Dienstleistungen und auch auf Regierungen und öffentliche Verwaltungen haben. Dies verspricht eine erfolgreiche Zukunft des Semantic Webs.

³² Das Wachstum der Linked Open Data Cloud wird dokumentiert unter: <http://linkeddata.org/>.

³³ <http://developer.yahoo.net/blog/archives/2008/07/>.

³⁴ <http://www.heppnetz.de/projects/goodrelations/>.

³⁵ http://www.ebusiness-unibw.org/wiki/GoodRelationsInGoogle#GoodRelations_in_Google_Rich_Snippets.

³⁶ <http://www.data.gov/>.

³⁷ <http://www.rdfabout.com/demo/census/>.

³⁸ <http://data.gov.uk>.

³⁹ <http://bio2rdf.org/>.

⁴⁰ <http://www.gopubmed.org/>.

Danksagung

Teile dieses Kapitels basieren auf den Veröffentlichungen von Janik et al. [40] und Harth et al. [32].

Literaturverzeichnis

- [1] D. Allemand and J. Hendler. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann, 2011.
- [2] D. Artz and Y. Gil. A Survey of Trust in Computer Science and the Semantic Web. *J. Web Sem.*, 5(2):58–71, 2007.
- [3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A Nucleus for a Web of Open Data. In *Semantic Web Conference and Asian Semantic Web Conference*, pages 722–735, November 2008.
- [4] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
- [5] T. Berners-Lee. Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web. RFC 1630, Internet Engineering Task Force, June 1994.
- [6] C. Bizer. The Emerging Web of Linked Data. *IEEE Intelligent Systems*, 24(5):87–92, 2009.
- [7] E. Blomqvist. Ontocase-automatic ontology enrichment based on ontology design patterns. In *International Semantic Web Conference*, pages 65–80, 2009.
- [8] H. Boley, G. Hallmark, M. Kifer, A. Paschke, A. Polleres, and D. Reynolds. RIF Core Dialect. W3C candidate recommendation, W3C, October 2009. <http://www.w3.org/TR/rif-core/>.
- [9] P. A. Bonatti and D. Olmedilla. Rule-Based Policy Representation and Reasoning for the Semantic Web. In *Reasoning Web Summer School*, volume 4636 of *Lecture Notes in Computer Science*, pages 240–268. Springer, 2007.
- [10] S. Borgo and C. Masolo. *Handbook on Ontologies*, chapter Foundational choices in DOLCE. Springer, 2nd edition, 2009.
- [11] M. Braun, A. Scherp, and S. Staab. Collaborative semantic points of interests. In *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part II*, volume 6089 of *Lecture Notes in Computer Science*, pages 365–369. Springer, 2010.
- [12] D. Brickley and R. Guha. RDF vocabulary description language 1.0: RDF schema. W3C recommendation, W3C, February 2004. <http://www.w3.org/TR/rdf-schema/>.
- [13] J. Broekstra, A. Kampman, and F. V. Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *International Semantic Web Conference*, pages 54–68. Springer, 2002.
- [14] R. Q. Dividino, S. Schenk, S. Sizov, and S. Staab. Provenance, Trust, Explanations – and all that other Meta Knowledge. *KI*, 23(2):24–30, 2009.
- [15] M. Duerst and M. Suignard. Internationalized resource identifiers (IRIs). RFC 3987, Internet Engineering Task Force, Jan. 2005.

- [16] M. Ehrig. *Ontology Alignment: Bridging the Semantic Gap*, volume 4 of *Semantic Web and Beyond*. Springer, 2007.
- [17] J. Euzenat and P. Shvaiko. *Ontology matching*, chapter Classifications of ontology matching techniques. Springer, 2007.
- [18] J. Euzenat and P. Shvaiko. *Ontology matching*. Springer, 2007.
- [19] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, USA, 2000.
- [20] G. Flouris, I. Fundulaki, P. Pediaditis, Y. Theoharis, and V. Christophides. Coloring RDF Triples to Capture Provenance. In *International Semantic Web Conference*, volume 5823 of *LNCS*, pages 196–212. Springer, 2009.
- [21] T. Franz, A. Schultz, S. Sizov, and S. Staab. TripleRank: Ranking Semantic Web Data by Tensor Decomposition. In *International Semantic Web Conference*, volume 5823 of *LNCS*, pages 213–228. Springer, 2009.
- [22] T. Franz, S. Staab, and R. Arndt. The X-COSIM integration framework for a seamless semantic desktop. In *Knowledge capture*, pages 143–150. ACM, 2007.
- [23] A. Gangemi and V. Presutti. *Handbook on Ontologies*, chapter Ontology Design Patterns. Springer, 2nd edition, 2009.
- [24] R. Gavriloaie, W. Nejdl, D. Olmedilla, K. E. Seamons, and M. Winslett. No Registration Needed: How to Use Declarative Policies and Negotiation to Access Sensitive Resources on the Semantic Web. In *European Semantic Web Symposium*, volume 3053 of *LNCS*, pages 342–356. Springer, 2004.
- [25] H. Glaser, A. Jaffri, and I. Millard. Managing co-reference on the semantic web. In *WWW2009 Workshop: Linked Data on the Web*, 2009.
- [26] J. Golbeck and J. A. Hendler. Inferring binary trust relationships in Web-based social networks. *ACM Trans. Internet Techn.*, 6(4):497–529, 2006.
- [27] J. Golbeck and A. Mannes. Using Trust and Provenance for Content Filtering on the Semantic Web. In *Models of Trust for the Web*, CEUR Workshop Proceedings. CEUR-WS.org, 2006.
- [28] A. Gómez-Pérez, M. F. López, and O. Corcho. *Ontological engineering*. Springer, 2004.
- [29] W. O. W. Group. OWL 2 Web Ontology Language Document Overview . W3C recommendation, W3C, October 2009. <http://www.w3.org/TR/owl2-overview/>.
- [30] H. Halpin and V. Presutti. An ontology of resources: Solving the identity crisis. In *European Semantic Web Conference*, pages 521–534, 2009.
- [31] A. Harth, K. Hose, M. Karnstedt, A. Polleres, K.-U. Sattler, and J. Umbrich. Data Summaries for On-Demand Queries over Linked Data. In *World Wide Web*. ACM, 2010.
- [32] A. Harth, M. Janik, and S. Staab. Semantic Web Architecture. *Handbook of Semantic Web Technologies*, 1:43–76, 2011.
- [33] O. Hartig. SPARQL for a Web of Linked Data: Semantics and Computability. In *9th Extended Semantic Web Conference (ESWC)*, volume 7295 of *LNCS*, pages 8–23. Springer, 2012.
- [34] O. Hartig, C. Bizer, and J. C. Freytag. Executing SPARQL Queries over the Web of Linked Data. In *International Semantic Web Conference*, pages 293–309, 2009.
- [35] O. Hartig and J.-C. Freytag. Foundations of Traversal Based Query Execution over Linked Data. In *23rd ACM Conference on Hypertext and Social Media, HT*, pages 43–52, 2012.
- [36] J. Hebeler, M. Fisher, R. Blace, and A. Perez-Lopez. *Semantic Web Programming*. Wiley, 2011.

- [37] J. Heinsohn, D. Kudenko, B. Nebel, and H.-J. Profitlich. An Empirical Analysis of Terminological Representation Systems. *Artif. Intell.*, 68(2):367–397, 1994.
- [38] P. Hitzler, M. Krötzsch, S. Rudolph, and Y. Sure. *Semantic Web: Grundlagen*. Springer-Verlag, 2008.
- [39] D. F. Huynh, D. R. Karger, and R. C. Miller. Exhibit: Lightweight Structured Data Publishing. In *World Wide Web*, pages 737–746. ACM, 2007.
- [40] M. Janik, A. Scherp, and S. Staab. The Semantic Web: Collective Intelligence on the Web. *Informatik Spektrum*, 34(5):469–483, 2011.
- [41] P. Kanani, A. McCallum, and C. Pal. Improving author coreference by resource-bounded information gathering from the web. In *Conference on Artificial intelligence*, pages 429–434, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.
- [42] G. Kasneci, S. Elbassuoni, and G. Weikum. MING: Mining Informative Entity Relationship Subgraphs. In *Information and Knowledge Management*, pages 1653–1656. ACM, 2009.
- [43] D. Oberle. *Semantic Management of Middleware*. Springer, 2006.
- [44] D. Oberle, A. Ankolekar, P. Hitzler, P. Cimiano, M. Sintek, M. Kiesel, B. Mouguouie, S. Baumann, S. Vembu, M. Romanelli, P. Buitelaar, R. Engel, D. Sonntag, N. Reithinger, B. Loos, H.-P. Zorn, V. Micelli, R. Porzel, C. Schmidt, M. Weiten, F. Burkhardt, and J. Zhou. Dolce ergo sumo: On foundational and domain models in the smartweb integrated ontology (swinto). *Web Semant.*, 5(3):156–174, Sept. 2007.
- [45] D. Oberle, N. Guarino, and S. Staab. What is an ontology? In S. Staab and R. Studer, editors, *Handbook on Ontologies*. Springer, 2nd edition, 2009.
- [46] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object Exchange Across Heterogeneous Information Sources. In *Data Engineering*, pages 251–260, Washington, DC, USA, 1995. IEEE Computer Society.
- [47] Y. Raimond, C. Sutton, and M. B. Sandler. Interlinking Music-Related Data on the Web. *IEEE MultiMedia*, 16(2):52–63, 2009.
- [48] S. Rendle and L. Schmidt-Thieme. Object identification with constraints. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18–22 December 2006, Hong Kong, China*, pages 1026–1031. IEEE Computer Society, 2006.
- [49] C. Saathoff and A. Scherp. Unlocking the semantics of multimedia presentations in the web with the multimedia metadata ontology. In M. Rappa, P. Jones, J. Freire, and S. Chakrabarti, editors, *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26–30, 2010*, pages 831–840. ACM, 2010.
- [50] C. Saathoff and A. Scherp. Unlocking the semantics of multimedia presentations in the web with the multimedia metadata ontology. In *World Wide Web*, 2010.
- [51] G. M. Sacco and Y. Tzitzikas, editors. *Dynamic Taxonomies and Faceted Search : Theory, Practice, and Experience*. Springer, Berlin, 2009.
- [52] S. Schenk, C. Saathoff, S. Staab, and A. Scherp. SemaPlorer – interactive semantic exploration of data and media based on a federated cloud infrastructure. *Journal of Web Semantics*, 2009.
- [53] S. Schenk and S. Staab. Networked graphs: a declarative mechanism for SPARQL rules, SPARQL views and RDF data integration on the web. In *World Wide Web*, pages 585–594. ACM, Apr. 21–25, 2008.

- [54] A. Scherp, D. Eißing, and C. Saathoff. A method for integrating multimedia metadata standards and metadata formats with the multimedia metadata ontology. *Int. Journal on Semantic Computing*, 2012.
- [55] A. Scherp, T. Franz, C. Saathoff, and S. Staab. A core ontology on events for representing occurrences in the real world. *Multimedia Tools Appl.*, 58(2):293–331, 2012.
- [56] A. Scherp, C. Saathoff, T. Franz, and S. Staab. Designing core ontologies. *Applied Ontology*, 6(3):177–221, 2011.
- [57] F. Schwagereit, A. Scherp, and S. Staab. Representing Distributed Groups with dgFOAF. In *Extended Semantic Web Conference*, LNCS. Springer, 2010.
- [58] S. Staab and R. Studer, editors. *Handbook on Ontologies*. Springer, 2009.
- [59] G. Stilois, G. B. Stamou, J. Z. Pan, V. Tzouvaras, and I. Horrocks. Reasoning with Very Expressive Fuzzy Description Logics. *J. Artif. Intell. Res.*, 30:273–320, 2007.
- [60] H. Stuckenschmidt, R. Vdovjak, J. Broekstra, and G.-J. Houben. Towards distributed processing of RDF path queries. *Int. J. Web Eng. Technol.*, 2(2/3):207–230, 2005.
- [61] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 697–706, New York, NY, USA, 2007. ACM.
- [62] G. Tummarello, R. Cyganiak, M. Catasta, S. Danielczyk, and S. Decker. Sig.ma: live views on the Web of Data. In *Semantic Web Challenge 2009 at the 8th International Semantic Web Conference (ISWC2009)*, 2009.
- [63] M. L. Wick, A. Culotta, K. Rohanimanesh, and A. McCallum. An entity based model for coreference resolution. In *SIAM International Conference on Data Mining*, pages 365–376, 2009.
- [64] M. L. Wick, K. Rohanimanesh, K. Schultz, and A. McCallum. A unified approach for schema matching, coreference and canonicalization. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 722–730, New York, NY, USA, 2008. ACM.
- [65] K. Wilkinson, C. Sayers, H. A. Kuno, and D. Reynolds. Efficient RDF storage and retrieval in Jena2. In I. F. Cruz, V. Kashyap, S. Decker, and R. Eckstein, editors, *SWDB*, pages 131–150, 2003.

17 Universelle Spielprogramme

Michael Thielscher

*It's the program that has to do the thinking.
It's like opening a box of Scrabble and saying, „Here's the rule book. Play.“*
MICHAEL GENESERETH, STANFORD

Der spielerische Wettbewerb zwischen Mensch und Maschine übt seit langem eine besondere Faszination aus. Schon der Computer- und KI-Pionier Alan Turing hat sich ernsthaft mit der Programmierung von Schachcomputern auseinandergesetzt, denn er sah darin ein wichtiges Experimentierfeld für die KI – erfordert doch das Schachspielen eine ganze Reihe von Fähigkeiten, die als charakteristisch für höhere Intelligenz gelten: vorausschauendes Denken, Planen, Entscheiden, Erwerben von Spezialwissen und vieles mehr. Wenig später begann Arthur Samuel mit der Entwicklung eines selbstlernenden Programms für das Brettspiel Dame, das als eine der frühesten Anwendungen der KI nach einiger Zeit in der Lage war, seinen eigenen Programmierer zu schlagen.

Als ultimativer Erfolg in der Schachprogrammierung gilt der legendäre Computer „Deep Blue“, der 1997 den damaligen Schachweltmeister in einem 6-Partien-Match bezwingen konnte. Mittlerweile gibt es mehrere kommerzielle Schachprogramme, deren Spielstärke die der weltbesten menschlichen Spieler überragt. Ein weiterer Meilenstein war die vollständige Lösung des Damesspiels nach mehreren Jahren Berechnungszeit. Weitaus größere Probleme bereitet den KI-Forschern das Spiel Go, aber auch auf diesem Gebiet hat das beste Programm mittlerweile das Niveau von Profispielern erreicht. Ein anderes KI-Produkt von besonderem kommerziellem Interesse sind automatische Pokerspieler, die ebenfalls mit Weltklassespieldern mithalten können. Generell gibt es heute für eine Vielzahl von Spielen – seien es Brett-, Karten- oder Computerspiele selbst – Wettbewerbe, bei denen KI-Systeme gegeneinander oder gegen menschliche Gegner antreten, und die zum Teil sehr lukrativ sein können.

Erfolge der KI-Spielprogrammierung in einzelnen Spielen tragen jedoch wenig zu der von Alan Turing erhofften Erkenntnis über die generelle Funktionsweise der menschlichen Intelligenz, wie sie sich im Spielen erweist, bei. Jedes speziell für ein bestimmtes Spiel entwickelte Programm basiert nämlich im Wesentlichen auf einer reinen Suche nach vorgegebenen Heuristiken, gegebenenfalls unterstützt durch große Datenbanken. Oft sind diese Heuristiken sehr ausgeklügelt, stets aber von menschlichen Programmierern vorgegeben. Damit endet aber die Intelligenz zum Beispiel eines Schachcomputers an den Grenzen dieses einen Spiels, also sobald er mit einem anderen als dem vorprogrammierten Spiel konfrontiert wird – und sei dies noch so trivial. Aufgrund dieser Einsicht besteht seit wenigen Jahren ein besonderes Interesse der KI an der Automatisierung der menschlichen Fähigkeit, immer wieder auch neue Spiele erlernen zu können. Dieses sogenannte *universelle Spielen* (engl. „general game playing“) erfordert von einem Programm, die Regeln eines beliebigen, zuvor unbekannten Spiels zu verstehen und ohne weiteres

menschliches Zutun zu erlernen. Solche Systeme sind für die KI-Forschung von großem Interesse, weil sie neben der klassischen Problemlösung durch Suche viele weitere Aspekte menschlicher Intelligenz erfordern: Repräsentation von Wissen, logisches Schließen, selbstständiges Erarbeiten von Heuristiken sowie Lernen aus Erfahrung, um nur die Wichtigsten zu nennen.

Vereinzelt haben sich KI-Wissenschaftler bereits seit den 1960’er Jahren mit den Prinzipien universeller Spielprogramme beschäftigt, aber erst mit der Ausrufung eines internationalen Wettbewerbs im Jahr 2005 – der seitdem jährlich ausgetragenen „AAAI General Game Playing Competition“ – wurde die Erforschung solcher Systeme zu einem wichtigen Teilgebiet der KI und gilt dort als große Herausforderung (engl. „grand challenge“). Ein Grund für die Popularität universeller Spielprogramme sind auch deren vielfältige Anwendungsmöglichkeiten: Schachcomputer etwa, die es dem Endbenutzer ermöglichen, die Spielregeln nach Belieben zu ändern und somit jede denkbare Variante des klassischen Schachs zu spielen, oder generische KI-Programme, die sich nur durch Eingabe der Spielregeln zu computergesteuerten Gegenspielern für im Prinzip jedes Computerspiel instanzieren lassen, Software-Agenten (vgl. Kapitel 14), die sich automatisch an neue Umgebungen und Anforderungen anpassen können – im Grunde eignet sich jedes Entscheidungsproblem, das als Spiel modelliert werden kann, beispielsweise im Wirtschafts- oder Finanzbereich, für die Anwendung universeller Spielprogramme.

Dieses Kapitel behandelt die Grundlagen universeller KI-Spielprogramme und die wichtigsten Methoden auf diesem Gebiet, von denen viele nach 2005 entwickelt wurden und einen Einblick in die aktuelle Forschung gewähren lassen.

17.1 Spielregeln beschreiben: Wissensrepräsentation

Die praktische Umsetzung von universellen Spielprogrammen erfordert zunächst eine Eingabesprache, in der die Regeln beliebiger Spiele für einen Computer einfach und unmissverständlich beschrieben werden können. Dabei sollte eine Spielbeschreibung stets vollständig sein, d.h., sie sollte neben den angegebenen Regeln kein zusätzliches (Allgemein-) Wissen erfordern, um ein Spiel korrekt spielen zu können. Prinzipiell können Spiele in jeder Programmiersprache algorithmisch codiert werden. Für die KI ist es jedoch von besonderem Interesse, eine *deklarative* Sprache zu verwenden, die sich stark an natürlichsprachlichen Spielregeln orientiert. Die in diesem Abschnitt behandelte Spielbeschreibungssprache *GDL* (engl. „Game Description Language“) erfüllt dieses Kriterium und hat sich als Standard in der Forschung und bei internationalen Wettbewerben etabliert.

17.1.1 Spielzustände und Züge

Wer ein Spiel in GDL beschreiben möchte, muss sich zunächst überlegen, wie die verschiedenen *Spielzustände* (Positionen) mit Hilfe einzelner *Stellungsmerkmale* repräsentiert werden sollen. Als Beispiel betrachten wir im Folgenden das klassische und einfache Spiel Tic Tac Toe. Abbildung 17.1 illustriert, wie ein Spielzustand auf dem zweidimensionalen Spielbrett mit Stellungsmerkmalen der Form `cell(_, _, _)` beschrieben werden kann. Jedes dieser Merkmale gibt für ein bestimmtes Feld an, ob es mit `x` oder `o` markiert bzw. leer (`b` für engl. „blank“) ist. Das zusätzliche Stellungsmerkmal `control(_)` besagt, welcher der beiden Spieler am Zug ist.

	<pre> cell(1,1,x) cell(1,2,b) cell(1,3,b) cell(2,1,b) cell(2,2,o) cell(2,3,b) cell(3,1,b) cell(3,2,b) cell(3,3,x) control(oplayer) </pre>
--	---

Abbildung 17.1: Ein Spielzustand in Tic Tac Toe.

Als Nächstes muss die Codierung der einzelnen Züge der Spieler festgelegt werden. Ein Zug in Tic Tac Toe besteht im Markieren einzelner Felder, zum Beispiel (2,1), das mit `mark(2, 1)` bezeichnet werden soll. Darüber hinaus benötigen wir noch einen Namen – nennen wir ihn `noop` – für den einzigen möglichen „Zug“, den ein Spieler ausführt, der nicht als Nächstes markieren darf. Dies ist notwendig, da in GDL generell jeder Spieler zu jedem Zeitpunkt zieht, um auf diese Weise auch Spiele mit echt gleichzeitigen Zügen modellieren zu können.

17.1.2 Spielregeln

GDL-Spielbeschreibungen setzen sich aus Fakten und Regeln zusammen. Dabei werden alle Ausdrücke in Präfixnotation geschrieben und Variablen durch ein führendes „?“ gekennzeichnet. Fakten sind „atomare“ Aussagen (vgl. Kapitel 5), während Regeln die Form

(<= Konklusion Bedingungen)

haben, wobei für die *Bedingungen* beliebige logische Ausdrücke verwendet werden können und die *Konklusion* wiederum eine atomare Aussage ist. In GDL werden einige *Schlüsselwörter* verwendet, die im Folgenden in jeder angegebenen Regel zur besseren Kennzeichnung fettgedruckt sind.

Die Spieler

Die einzelnen Rollen in einem Spiel werden mit Hilfe von Fakten und dem Schlüsselwort `role` definiert. An der Zahl dieser Fakten lässt sich ablesen, ob es sich um ein Ein-, Zwei- oder Mehrpersonenspiel handelt. Die beiden Spieler in Tic Tac Toe können beispielsweise folgendermaßen benannt werden:

```

1 (role xplayer)
2 (role oplayer)

```

Ausgangsstellung

Die Ausgangsstellung wird mit Hilfe des Schlüsselworts `init` gebildet. In Tic Tac Toe sind zu Beginn alle Felder leer und der Kreuz-Spieler beginnt:

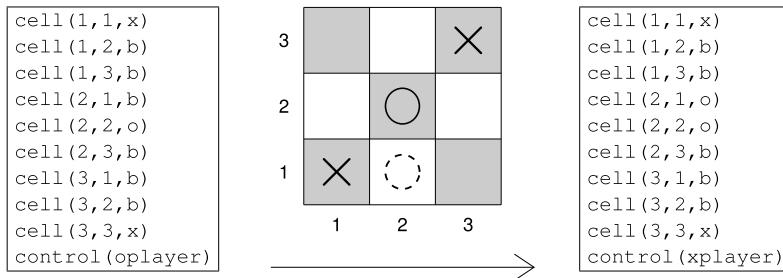


Abbildung 17.2: So ändert sich der Spielzustand, wenn der zweite Spieler in der Position aus Abbildung 17.1 das Feld (2,1) markiert.

```

3 (init (cell 1 1 b))
4 (init (cell 1 2 b))
5 (init (cell 1 3 b))
6 (init (cell 2 1 b))
7 (init (cell 2 2 b))

8 (init (cell 2 3 b))
9 (init (cell 3 1 b))
10 (init (cell 3 2 b))
11 (init (cell 3 3 b))
12 (init (control xplayer))

```

Diese Fakten verdeutlichen die GDL-Syntax, die im Unterschied zu der in Abbildung 17.1 verwendeten Schreibweise ohne Kommata auskommt.

Zugmöglichkeiten

Die möglichen Züge der Spieler werden mit Hilfe des Schlüsselworts `legal` spezifiziert. In den meisten Fällen hängt die Ausführbarkeit eines Zugs von bestimmten Bedingungen an die jeweilige Position ab. Dies wird mit Hilfe des Schlüsselworts `true` ausgedrückt, das stets ein Stellungsmerkmal als Argument hat.

In Tic Tac Toe kann ein Spieler ein beliebiges freies Feld markieren, wenn er am Zug ist, während der jeweils andere Spieler stillhalten muss:

```

13 (<= (legal ?r (mark ?m ?n))
14     (true (cell ?m ?n b))
15     (true (control ?r)))
16

17 (<= (legal xplayer noop)
18     (true (control oplayer)))
19 (<= (legal oplayer noop)
20     (true (control xplayer)))

```

Neue Position

Die Effekte der einzelnen Züge werden mit Regeln beschrieben, die zusammengenommen alle Stellungsmerkmale der resultierenden Spielposition angeben. Als einfaches Beispiel illustriert Abbildung 17.2 den Positionsübergang nach Markieren eines Felds in Tic Tac Toe. In GDL werden die neuen Stellungsmerkmale mit dem Schlüsselwort `next` gekennzeichnet. In den meisten Fällen ist eine neue Position sowohl von Merkmalen der Vorgängerstellung (Schlüsselwort `true`) als auch von den ausgeführten Zügen abhängig, welche mit dem Schlüsselwort `does` referenziert werden. Beim Aufstellen der Regeln für die Nachfolgestellung ist darauf zu achten, dass alle Stellungsmerkmale abgeleitet werden können, einschließlich derer, die bereits in der Vorgängerstellung gelten.

Wie aus Abbildung 17.2 ersichtlich, führt in Tic Tac Toe das Markieren eines Felds dazu, dass sich der Inhalt dieses Felds entsprechend ändert, während der Rest des Spielbretts nicht beeinflusst wird. Darüber hinaus wandert nach jedem Zug die Kontrolle zu dem jeweils anderen Spieler:

```

21 (<= (next (cell ?m ?n x))
22   (does xplayer (mark ?m ?n)))
23
24 (<= (next (cell ?m ?n o))
25   (does oplayer (mark ?m ?n)))
26
27 (<= (next (control xplayer))
28   (true (control oplayer)))
29
30 (<= (next (cell ?m ?n ?w))
31   (true (cell ?m ?n ?w)))
32   (does ?r (mark ?j ?k)))
33   (or (distinct ?m ?j)
34     (distinct ?n ?k)))
35
36 (<= (next (control oplayer))
37   (true (control xplayer)))

```

Die Regel in den Zeilen 30–34 betrifft alle Spielfelder (m, n) , deren Inhalt *nicht* verändert wird, wobei die Bedingung ist, dass (m, n) nicht gleich dem gerade markierten Feld (j, k) ist, d.h., die Felder müssen sich in wenigstens einer ihrer beiden Koordinaten voneinander unterscheiden. Dies wird mit Hilfe des GDL-Schlüsselwort **distinct** ausgedrückt, mit dem getestet werden kann, ob zwei Ausdrücke syntaktisch verschieden sind.

Spielende

Die Bedingungen, unter denen ein Spiel zu Ende ist, werden mit Regeln für das Schlüsselwort **terminal** beschrieben. Für unser Beispiel erweist es sich im Sinne einer modularen, leicht lesbaren Beschreibung als hilfreich, dass GDL die Definition beliebiger Hilfsprädikate erlaubt:

```

38 (<= (row ?m ?w)                                51 (<= (diagonal ?w)
39   (true (cell ?m 1 ?w))                      52   (true (cell 1 1 ?w))
40   (true (cell ?m 2 ?w))                      53   (true (cell 2 2 ?w))
41   (true (cell ?m 3 ?w)))                     54   (true (cell 3 3 ?w)))
42
43 (<= (column ?n ?w)                               55
44   (true (cell 1 ?n ?w))                      56 (<= (diagonal ?w)
45   (true (cell 2 ?n ?w))                      57   (true (cell 1 3 ?w))
46   (true (cell 3 ?n ?w)))                     58   (true (cell 2 2 ?w))
47
48 (<= (line ?w) (row ?m ?w))                    59   (true (cell 3 1 ?w)))
49 (<= (line ?w) (column ?m ?w))                  60
50 (<= (line ?w) (diagonal ?w))                  61 (<= open
51
52
53
54
55
56
57
58
59
60
61
62
63

```

Mit diesen Definitionen lässt sich leicht festlegen, dass Tic Tac Toe beendet ist, sobald ein Spieler eine komplette Linie (Zeile, Spalte oder Diagonale) mit seinem Symbol markiert hat. Ebenfalls zu Ende ist das Spiel, wenn das Brett nicht mehr offen ist, bzw. kein freies Feld mehr zur Verfügung steht:

```

64  (<= terminal
65    (or (line x) (line o) (not open)))

```

Spielziel

Zu guter Letzt muss noch das Ziel eines Spiels beschrieben werden. Dies geschieht durch das Schlüsselwort **goal**, mit dem für jeden Spieler definiert werden kann, welchen Wert für ihn eine bestimmte Endstellung hat. Einer Konvention folgend werden üblicherweise Punkte zwischen 0 (Verlust) und 100 (Sieg) vergeben, so dass 50 als Mittelwert ein Unentschieden bedeutet. Natürlich sind auch Zwischenwerte denkbar; darüber hinaus müssen sich die Ergebnisse aller Spieler im gegebenen Endzuständen nicht notwendigerweise zu 100 addieren lassen, so dass auch sogenannte kooperative Spiele beschrieben werden können. Für Tic Tac Toe gelten jedoch folgende Regeln:

```

66  (<= (goal xplayer 100)           75  (<= (goal oplayer 100)
67    (line x))                         76    (line o))
68
69  (<= (goal xplayer 50)            77
70    (not (line x)))                  78  (<= (goal oplayer 50)
71    (not (line o)))                  79    (not (line x)))
72
73  (<= (goal xplayer 0)             80    (not (line o)))
74    (line o))                         81
82  (<= (goal oplayer 0)             82    (goal oplayer 0)
83    (line x))

```

17.1.3 GDL: Zusammenfassung

Die Sprache GDL erlaubt es, allgemeine Spiele modular und kompakt zu beschreiben, so dass auch komplexe Spiele wie Schach oder Go, deren Regeln selbst vergleichsweise einfach und leicht erlernbar sind, nur wenige hundert Zeilen GDL-Code erfordern. Dabei genügt die Kenntnis der insgesamt neun Schlüsselwörter, um eine Spielbeschreibung vollständig und ohne weiteres Allgemeinwissen verstehen zu können. Die Beschreibungssprache ist darüber hinaus sehr vielseitig. So können neben den klassischen Zweipersonenspielen auch beliebige Mehrpersonenspiele beschrieben werden, einschließlich solcher, bei denen simultan gezogen wird, sowie Spiele, die kooperativer Natur sind oder in denen Spieler in Teams gegeneinander antreten.

GDL hat jedoch auch Einschränkungen: Spielzustände sind diskret, erlauben also keine kontinuierlichen Parameter, und es wird stets synchron gezogen. Darüber hinaus lassen sich nicht ohne Weiteres Spiele beschreiben, in denen zusätzliches Wissen gefragt ist, wie etwa bei Scrabble, für dessen vollständige Beschreibung in GDL die Codierung eines kompletten Lexikons benötigt würde. Weiterhin lassen sich in der Grundversion von GDL weder Spiele mit Zufallszügen (zum Beispiel Backgammon) noch solche mit Informationsasymmetrien (zum Beispiel die meisten

Kartenspiele) beschreiben; diese Einschränkungen sind allerdings mit der Spracherweiterung zu GDL-II (vgl. Abschnitt 17.7) aufgehoben worden.

Da die vollständige Spezifikation der GDL-Syntax den Rahmen dieses Kapitels sprengen würde, sei hierfür auf [11] verwiesen.

17.1.4 Kommunikationsprotokoll für GDL

Für die Durchführung von Spielen und Wettbewerben mit universellen Spielprogrammen gibt es ein Standardkommunikationsprotokoll zwischen einem *Spieleleiter* (engl. „Game Controller“) und den an einem konkreten Match teilnehmenden Programmen:

1. Zunächst erhalten die teilnehmenden Spielprogramme die GDL-Regeln eines neuen Spiels. Gleichzeitig wird ihnen mitgeteilt, welche der jeweiligen Rollen (zum Beispiel `xplayer`) sie übernehmen sollen, wie viel Zeit ihnen zunächst gegeben wird, das neue Spiel zu erlernen (beispielsweise 20 Minuten), und wie lange sie dann während des eigentlichen Spiels über jeden Zug nachdenken können (zum Beispiel 30 Sekunden).
2. Nach Ablauf der Lernphase verkündet der Spieleleiter den Beginn des Spiels, woraufhin jeder Spieler innerhalb der vorgegebenen Zeit seinen ersten Zug übermittelt.¹
3. Anschließend informiert der Spieleleiter jeden Spieler über die gewählten Züge aller Teilnehmer, so dass diese anhand der Spielregeln den aktuellen Zustand berechnen können. Innerhalb der vorgegebenen Zeit müssen die Spieler dann ihren nächsten Zug an den Spieleleiter übermitteln.
4. Schritt 3 wird so lange wiederholt, bis nach den Spielregeln ein Endzustand erreicht ist, was abschließend vom Spieleleiter verkündet wird.

Ein solches Kommunikationsprotokoll ist sehr einfach zu implementieren, und es gibt frei verfügbare Software für Spieleleiter, mit der komplett Spiele durchgeführt werden können.² Darüber hinaus werden Online-Server unterhalten, auf denen ununterbrochen universelle Spielprogramme gegeneinander antreten.³

17.2 Spielregeln verstehen: Inferenz

Die erste Aufgabe eines universellen Spielprogramms besteht darin, mit Hilfe der gegebenen Regeln zu schließen, wie ein neues Spiel überhaupt gespielt wird. Im Einzelnen bedeutet dies, folgende Fragen beantworten zu können:

1. Wie sieht die Ausgangsstellung aus?
2. Welche Zugmöglichkeiten hat ein Spieler in einer gegebenen Stellung?

¹ Sollte es ein Spieler nicht geschafft haben, rechtzeitig einen legalen Zug zu senden, so wählt in der Praxis meist der Spieleleiter automatisch einen zufälligen Zug aus, wobei bei internationalen Wettbewerben strenge Regeln gelten, nach denen ein Spiel als verloren gewertet wird, sobald eine vorgegebene Maximalzahl erlaubter Verstöße überschritten ist.

² Siehe zum Beispiel sourceforge.net/projects/ggpserver/files/gamecontroller/.

³ Siehe zum Beispiel ggpserver.general-game-playing.de.

3. Welche Nachfolgestellung ergibt sich aus einer gegebenen Stellung bei ebenfalls gegebenen Zügen für jede Rolle?
4. Ist das Spiel in einer gegebenen Stellung zu Ende, und wenn ja, wie lautet das Ergebnis für die einzelnen Spieler?

Für ein in GDL beschriebenes Spiel lässt sich die Beantwortung dieser Fragen mittels automatisierter *Inferenz* implementieren. Die GDL-Schlüsselwörter weisen darauf hin, welche logischen Schlüsse dazu im Einzelnen erforderlich sind. Dafür definieren wir zunächst zwei Abkürzungen. Die Codierung einer beliebigen Spielstellung S , die durch eine endliche Menge von Stellungsmerkmalen $\{f_1, \dots, f_n\}$ charakterisiert ist, sei durch n Fakten wie folgt gegeben:

$$S^{\text{true}} = \{(\text{true } f_1) \dots (\text{true } f_n)\}$$

Auf ähnliche Weise sei eine Zuordnung Z von Zügen z_1, \dots, z_k zu den $k \geq 1$ Spielern r_1, \dots, r_k eines Spiels durch k Fakten wie folgt codiert:

$$Z^{\text{does}} = \{(\text{does } r_1 z_1) \dots (\text{does } r_k z_k)\}$$

Definition 17.2.1. (Semantik einer GDL-Beschreibung) Eine gegebene GDL-Regelmenge G beschreibt ein aus den folgenden Komponenten bestehendes Spiel:

1. Spieler sind alle r , für die $(\text{role } r)$ aus G ableitbar ist.
2. Die Ausgangsstellung ist die Menge aller f , für die $(\text{init } f)$ aus G ableitbar ist.
3. Zug z ist ein legaler Zug für Spieler r in Stellung S , wenn $(\text{legal } r z)$ aus $G \cup S^{\text{true}}$ ableitbar ist.
4. Die aus den Zügen Z in einer Stellung S resultierende Nachfolgeposition ist durch die Menge aller f gegeben, für die $(\text{next } f)$ aus $G \cup S^{\text{true}} \cup Z^{\text{does}}$ ableitbar ist.
5. Position S ist eine Endstellung, wenn terminal aus $G \cup S^{\text{true}}$ ableitbar ist.
6. Der Ergebniswert v für Spieler r in einer Endstellung S ergibt sich durch Ableitung von $(\text{goal } r v)$ aus $G \cup S^{\text{true}}$.

Abbildung 17.3 zeigt die beschriebene Verwendung der Schlüsselwörter anhand des Spiels Tic Tac Toe. Dabei repräsentiert das Symbol \vdash die Ableitbarkeit aus den gegebenen Fakten unter Anwendung der in Abschnitt 17.1 beschriebenen Spielregeln.

Für die Automatisierung solcher logischen Schlüsse (Inferenzen) gibt es verschiedene Methoden (siehe dazu auch Kapitel 5). Im Folgenden beschreiben wir ein Verfahren, mit dem ausgehend von einer *Anfrage* diese mit Hilfe einer *Ableitung* beantwortet wird. Dabei besteht eine Ableitung aus einer Verkettung von *Ableitungsschritten*, von denen jeder einzelne wiederum eine gegebenen Regel anwendet (bzw. einen Fakt, wobei im Folgenden Fakten der Einfachheit halber als Regeln mit „leerer“ Bedingung betrachtet werden). Als Beispiel seien etwa folgende Regeln gegeben:

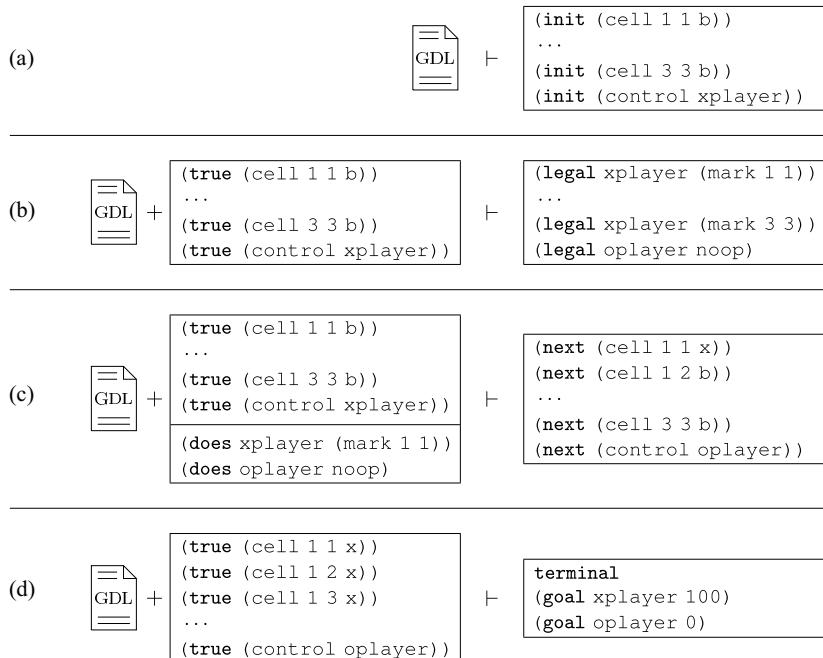


Abbildung 17.3: (a) Berechnung der Ausgangsposition. (b) Berechnung der legalen Züge in einer gegebenen Stellung. (c) Berechnung einer Nachfolgeposition. (d) Erkennen eines Endzustands und Bestimmung des Ergebnisses.

```

1 (true (cell 1 1 x))          9 (true (cell 3 3 x))
2 (true (cell 1 2 b))          10 (true (control oplayer))
3 (true (cell 1 3 b))          11
4 (true (cell 2 1 b))          12 (<= (legal ?r (mark ?m ?n))
5 (true (cell 2 2 o))          13 (true (cell ?m ?n b))
6 (true (cell 3 3 b))          14 (true (control ?r)))
7 (true (cell 3 1 b))          15 (<= (legal xplayer noop)
8 (true (cell 3 2 b))          16 (true (control oplayer)))

```

Interessieren wir uns für die Suche nach einem legalen Zug für den zweiten Spieler (wie in Abbildung 17.3(b)), so können wir die Anfrage (legal oplayer ?z) stellen, für die sich unter anderem folgende Ableitung finden lässt, wobei in jedem Ableitungsschritt eine bestimmte Regel „rückwärts“ angewendet wird, um das jeweils erste Teilziel durch neue Teilziele zu ersetzen:

$ \begin{aligned} & (\text{legal oplayer ?z}) \\ \implies & (\text{true (cell ?m ?n b)}) \\ \implies & (\text{true (control oplayer)}) \\ \implies & (\text{true (control oplayer)}) \text{ Regel 10} \\ \implies & \square \end{aligned} $	Regel 12–14 mit $?r = \text{oplayer}$, $?z = (\text{mark ?m ?n})$ Regel 2 mit $?m = 1, ?n = 2$ Regel 10
---	---

Das Symbol \square steht für die „leere“ Anfrage, die das erfolgreiche Ende einer Ableitung markiert. Die Variablenbindungen ergeben zusammengenommen Aufschluss über die *berechnete Antwort* für die in der Ausgangsfrage verwendeten Variablen, in diesem Fall $?z = (\text{mark } 1\ 2)$.

Die hier am Beispiel skizzierte Vorgehensweise zur automatisierten Inferenz auf der Basis von GDL-Regeln sei im Folgenden formal beschrieben.

17.2.1 Unifikation/Grundinstanzierung

Die Anwendung einer Regel auf ein Teilziel ist nur möglich, wenn eine Variablenbindung gefunden werden kann, durch die das Teilziel und die Konklusion der Regel identisch werden. Dieser Vorgang wird auch als *Unifikation* bezeichnet; die resultierende Bindung von Variablen entsprechend als *Unifikator*. Ein Beispiel ist die in obiger Ableitung im ersten Schritt verwendete Variablenbindung, die einen Unifikator für die beiden Ausdrücke (`legal oplayer ?z`) und (`legal ?r (mark ?m ?n)`) darstellt. Lässt sich eine solche Variablenbindung nicht finden, sind die beiden Ausdrücke nicht *unifizierbar*; dies gilt zum Beispiel für (`legal ?r (mark ?x ?x)`) und (`legal ?s (mark 1 2)`), da Variable *x* nicht zugleich durch 1 und 2 ersetzt werden kann. Für das Berechnen von Unifikatoren gibt es einfache Algorithmen, bei denen rekursiv die jeweiligen Argumente zweier gegebener Ausdrücke miteinander verglichen und dabei einzelne Variablenersetzungen erzeugt werden (siehe dazu auch Abschnitt 5).

Eine Alternative zur Unifikation ist die vollständige *Grundinstanzierung* einer gegebenen GDL-Spielbeschreibung vor Beginn eines Spiels. Dies erfordert, für jede Regel jede mögliche Kombination der Ersetzung von Variablen durch Werte zu erzeugen. Dabei können die für die einzelnen Variablen relevanten Wertebereiche mittels des weiter unten beschriebene *Domänengraphen* (vgl. Abschnitt 17.6.1) bestimmt werden, so dass beispielsweise für Regel 12–14 nur solche Grundinstanzen generiert werden, bei denen Variable *r* durch einen der beiden Spieldaternamen und Variablen *m, n* durch Koordinaten ersetzt werden. Die Grundinstanzierung als Vorverarbeitungsschritt hat den Vorteil, die Verwendung von Datenstrukturen, mit denen sehr effizient gerechnet werden kann, zu ermöglichen. Dem steht der Nachteil eines um mehrere Größenordnungen erhöhten Speicheraufwands gegenüber, der nicht bei allen Spielen praktikabel ist.

17.2.2 Ableitungsschritt (ohne Negation)

Wir betrachten zunächst den einfachen Fall, dass wie im obigen Beispiel keine Teilziele mit Negation (`not`) vorkommen. In diesem Fall besteht ein einzelner Ableitungsschritt darin, für das erste Teilziel eine Regel zu finden, deren Konklusion mit dem Teilziel unifizierbar ist. Ein entsprechender Unifikator wird auf alle betroffenen Variablen angewendet, und anschließend wird das alte Teilziel entfernt und durch die neuen Teilziele, die sich aus den Bedingungen der angewendeten Regel ergeben, ersetzt. Ist der Bedingungsteil der Regel leer, so wird das fragliche Teilziel ersatzlos gestrichen. Bei der Unifikation ist darauf zu achten, dass die Variablen in einer Regel stets lokal sind, also bei wiederholter Anwendung ein und derselben Regel stets neu gebunden werden können.

17.2.3 Ableitungen

Eine Ableitung ergibt sich aus der Aneinanderreihung einzelner Ableitungsschritte. Eine *erfolgreiche* Ableitung endet mit der leeren Anfrage. Die im Erfolgsfall berechnete *Antwort* für die Variablen in der ursprünglichen Anfrage ergibt sich aus der Verknüpfung der in den einzelnen Schritten verwendeten Unifikatoren. Eine Ableitung *schlägt fehl*, sobald auf das nächste Teilziel keine Regel anwendbar ist, sich also keine Konklusion einer vorhandenen Regel mit dem Teilziel unifizieren lässt. Ein Beispiel ist der folgende Ableitungsversuch eines legalen Zugs für den Kreuz-Spieler in der oben betrachteten Situation:

```
(legal xplayer ?z)          Regel 12-14 mit ?r = xplayer,  

                            ?z = (mark ?m ?n)  

⇒ (true (cell ?m ?n b))    Regel 2 mit ?m = 1, ?n = 2  

    (true (control xplayer))  

⇒ (true (control xplayer)) Fehlschlag
```

Im Fall eines Fehlschlags kann die Methode des *Backtracking* angewendet werden, bei der systematisch alle an früheren Stellen einer Ableitung möglichen Alternativen (d.h. ebenfalls anwendbare Regeln) betrachtet werden. Auf diese Weise wird etwa für unsere zunächst fehlgeschlagene Anfrage eine erfolgreiche Ableitung gefunden, bei der im ersten Schritt Regel 15–16 an Stelle von Regel 12–14 verwendet wird, was schließlich in der Antwort $?z = noop$ resultiert.

Die Methode des Backtracking wird auch zur systematischen Berechnung aller Antworten für eine Anfrage verwendet. Dies ist etwa erforderlich, um alle Stellungsmerkmale der Ausgangsposition als Antworten auf die Anfrage (`init ?f`) zu bestimmen oder alle legalen Züge eines Spielers r mittels der Anfrage (`legal r ?f`) bzw. alle Stellungsmerkmale einer Nachfolgeposition mittels (`next ?f`) zu generieren.

17.2.4 Regeln mit Negation

Eine negierte Bedingung ($\text{not } Q$) in einer Regel wird nach dem Prinzip „Negation durch Fehlschlag“ behandelt. Hierzu wird versucht, eine Ableitung für Q selbst zu konstruieren. Ist dies *nicht* möglich, d.h., schlägt jeder Ableitungsversuch fehl, so gilt das negierte Teilziel als gelöst und kann aus der Anfrage entfernt werden. Findet sich hingegen auch nur eine Ableitung für Q , so schlägt die Ableitung für die Anfrage, in der ($\text{not } Q$) gefordert wird, fehl. Unter Umständen kann es zu verschachtelten Anwendungen dieses Prinzips kommen, wie ein einfaches Beispiel zeigt. Dafür nehmen wir eine gedachte Beschreibung eines Dreipersonenspiels an, die folgende Fakten und Regeln enthält:

```

1 (role red)                                7 (role green)
2 (role blue)                                 8 (true (lifeline blue))
3
4 (<= (goal ?p 100)                         9
5   (role ?p)                               10 (<= (trapped ?p)
6   (not (trapped ?p)))                   11   (role ?p)
7
8
9
10
11
12   (not (true (lifeline ?p))))
```

Anfrage (`goal ?r 100`) hat nur die eine Antwort `?r = blue`:

$(goal ?r 100)$	Regel 4–6 mit $?p = ?r$
$\Rightarrow (role ?r)$	Regel 2 mit $?r = \text{blue}$
$(\text{not} (\text{trapped} ?r))$	
$\Rightarrow (\text{not} (\text{trapped blue}))$	Nebenrechnung (*)
$\Rightarrow \square$	
$(*) (\text{trapped blue})$	Regel 10–12 mit $?p = \text{blue}$
$\Rightarrow (role \text{blue})$	Regel 2
$(\text{not} (\text{true} (\text{lifeline blue})))$	
$\Rightarrow (\text{not} (\text{true} (\text{lifeline blue})))$	Fehlschlag laut Nebenrechnung (**)
$(**) (\text{true} (\text{lifeline blue}))$	Regel 8
$\Rightarrow \square$	

17.2.5 Regeln mit Disjunktion

Enthält eine Regel eine disjunktive Bedingung ($\text{or } Q_1 \dots Q_m$), so bietet jedes der Argumente Q_i für sich eine Alternative bei der Ableitung der Bedingung. Diese Alternativen können gegebenenfalls auch zu unterschiedlichen Antworten ein und derselben Anfrage führen, müssen als beim *Backtracking* entsprechend berücksichtigt werden. Damit ist jede Regel der Form

$$(<= P \dots (\text{or } Q_1 \dots Q_m) \dots R)$$

äquivalent zu den m Regeln

$$\begin{aligned} &(<= P \dots Q_1 \dots R) \\ &\dots \\ &(<= P \dots Q_m \dots R) \end{aligned}$$

und wird entsprechend behandelt.

17.3 Spielbaumsuche

Ein universelles Spielprogramm, das die grundlegenden Komponenten eines Spiels aus einer gegebenen GDL-Beschreibung berechnen kann, ist in der Lage, beliebige Spiele legal zu Ende zu spielen. Um nun auch gute von schlechten Zügen zu unterscheiden, betrachten wir zunächst den sogenannten *Spielbaum*, der mit Hilfe der Spielregeln generiert wird und in dem alle möglichen Verläufe des Spiels repräsentiert sind.

17.3.1 Minimax-Verfahren

Der Aufbau eines Spielbaums beginnt mit dem Ausgangszustand als Wurzelknoten. Für diesen werden alle legalen Zugmöglichkeiten und die daraus jeweils resultierenden Nachfolger berechnet, die dann selbst wieder auf gleiche Weise expandiert werden. Positionen, in denen den Spielregeln zufolge das Spiel zu Ende ist, bilden die Blätter des Baums.

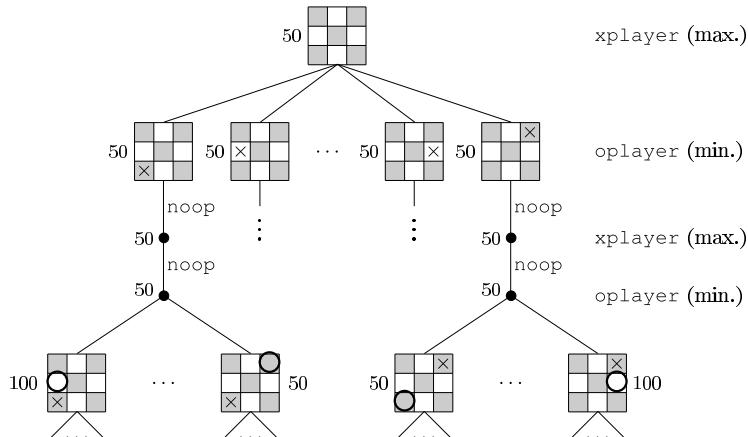


Abbildung 17.4: Der Anfang des Spielbaums für Tic Tac Toe für die ersten zwei Runden, wobei die eigentlich gleichzeitig stattfindenden Züge beider Spieler getrennt wurden – zuerst zieht jeweils der Kreuz-Spieler(xplayer).

Obwohl in GDL stets alle Spieler gleichzeitig ziehen, ist es für die Beurteilung der eigenen Züge hilfreich, diese beim Aufbau eines Spielbaums von den gegnerischen zu trennen. Dies geschieht durch Einfügen von Zwischenknoten, bei denen zunächst unser Spieler selbst zieht. Anschließend werden die Zwischenknoten um alle Kombinationen von Zügen des oder der Gegner (im Fall von Mehrpersonenspielen) erweitert. Diese Trennung ist dadurch motiviert, dass das Programm seinen eigenen Zug immer frei wählen kann, aber keine Kontrolle über die Züge der anderen Spieler hat. Abbildung 17.4 zeigt den obersten Teil des auf diese Weise gebildeten Spielbaums für Tic Tac Toe aus Sicht des Kreuz-Spielers, während Abbildung 17.5 einen

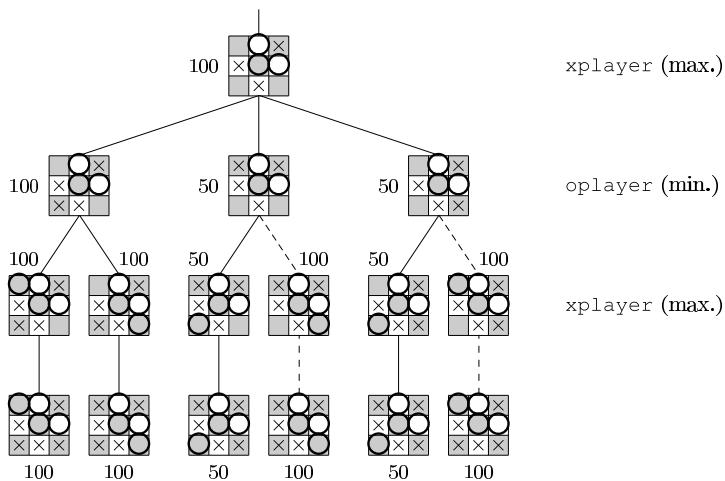


Abbildung 17.5: Ein kleiner Ausschnitt aus dem unteren Teil des Tic-Tac-Toe-Spielbaums, wobei der Kürze halber die jeweiligen Zwischenknoten mit den noop-Zügen weggelassen wurden.

Ausschnitt aus dem unteren Teil dieses Baums mit Blattknoten zeigt. (Die in den Abbildungen aufgeführten Zahlen und die Tatsache, dass einige Verbindungslien gestrichelt sind, sollten für den Moment ignoriert werden.)

Für einfache Spiele wie Tic Tac Toe sollte ein effizient implementiertes, universelles Spielprogramm in der Lage sein, alle Pfade eines Spielbaums bis zum Ende zu expandieren. Die Bewertung der einzelnen Züge basiert dann auf den Werten, die die einzelnen Endknoten für den Spieler haben und die sich aus den Regeln für das GDL-Schlüsselwort `goal` ergeben. Diese Werte können nach oben propagierte werden, um auf diese Weise auch Zwischenzustände zu bewerten. Das Standardverfahren hierfür ist unter dem Namen *Minimax* bekannt. Dabei berechnet sich der Wert eines Knoten, in dem der Spieler seinen eigenen Zug wählt, als das Maximum der Werte aller Nachfolgeknoten, während für die Knoten, in denen die anderen Spieler ziehen, das Minimum der Nachfolgeknoten genommen wird. Dies trägt der Tatsache Rechnung, dass der Spieler stets den für sich besten Zug wählen kann, aber keine Kontrolle über die Züge der Gegner hat, also im schlechtesten Fall mit dem für ihn ungünstigsten Wert rechnen muss.

Abbildung 17.5 illustriert die Minimax-Methode am Beispiel. Es ergibt sich daraus, dass der zuerst gezeigte Knoten für den Kreuz-Spieler den Wert 100 hat, da er mit dem linken Zug das Spiel in jedem Fall gewinnt. Werden diese Werte weiter nach oben propagierte, dann ergeben sich für die Knoten am Anfang des Spielbaums die in Abbildung 17.4 gezeigten Bewertungen. Tic Tac Toe endet demnach bei bestem Spiel in einem Unentschieden. Für die beiden Positionen links und rechts außen im unteren Teil von Abbildung 17.4 gibt es jedoch für den Kreuz-Spieler jeweils einen garantierten Gewinnweg, wie sich aus der Bewertung 100 ablesen lässt.

Bei der *Minimax-Suche* wird eine vorgegebene, systematische Suchmethode auf einen Spielbaum mit dem Ziel angewendet, einen eigenen Zug mit maximalem Wert in der momentanen Spielstellung, die im Wurzelknoten des Baums repräsentiert ist, zu finden. Dabei bietet sich für ein universelles Spielprogramm zunächst die sogenannte uninformierte Suche an (vgl. Abschnitt 3.2), wobei in der Spielprogrammierung die Methode der Wahl meist die iterative Tieffensuche ist, da auf diese Weise zum Beispiel ein kurzer Gewinnweg auch tatsächlich schnell gefunden wird.

17.3.2 Optimierungen

In nahezu jedem Spiel gibt es Stellungen, die auf mehreren Wegen erreicht werden können und die dann in einem Spielbaum mehrfach auftauchen, da sie auf unterschiedlichen Pfaden liegen. So kann beispielsweise die in Abbildung 17.5 zuerst gezeigte Spielsituation offensichtlich durch $3! \cdot 3! = 36$ verschiedene Zugfolgen entstehen, d.h., es gibt im vollständigen Spielbaum für Tic Tac Toe 36 Kopien des gesamten, in der Abbildung dargestellten Teilbaums. Da diese Kopien identisch sind, ist auch der berechnete Wert der oberen Stellung in all diesen Spielbaumknoten gleich. Es genügt daher, ihn bei der Anwendung des Minimax-Verfahrens nur einmal zu berechnen und sich das Expandieren des gleichen Knotens an anderen Stellen zu ersparen. Dies geschieht mit Hilfe einer Hashtabelle, die üblicherweise *Transpositionstabelle* genannt wird⁴ und in der alle während der Minimax-Suche gefundenen und bewerteten Positionen gespeichert werden. Auf diese Weise kann ein Spielbaum speziell in den tieferen Regionen oft drastisch beschnitten und damit die Suche signifikant verkürzt werden. So gibt es beispielswei-

⁴ Der Name nimmt auf die Zugvertauschungen (Transpositionen) Bezug, die Anlass zur Verwendung dieser Tabelle geben.

se in Tic Tac Toe lediglich 5478 erreichbare Zustände, während der vollständige Spielbaum aus rund 550 000 Knoten besteht (nicht mitgezählt die noop-Zwischenknoten).

Eine weitere allgemeine Methode zur Beschränkung des Spielbaums beruht auf der Einsicht, dass es für die Suche nach einem besten Zug nicht notwendig ist, alle erreichbaren Positionen zu bewerten. Vielmehr genügt es zu wissen, dass ein bestimmter Zug keinesfalls besser als eine bereits gefundener und bewerteter Zug ist. Dies ist ebenfalls in Abbildung 17.5 illustriert: Bei systematischer Suche von links nach rechts ist nach Analyse des linken der drei Teilbäume unterhalb des Ausgangsknotens der Wert des ersten Zugs bekannt (100). Nach Analyse des ersten kompletten Pfads des mittleren Teilbaums ist klar, dass der zweite Zug höchstens zu einem Wert von 50 führen kann. In diesem Teilbaum muss daher nicht weitergesucht werden, wenn das eigentliche Ziel der Suche nur ist, den Wert der Ausgangsstellung und einen besten Zug zu bestimmen. Gleiches gilt für den dritten Teilbaum, so dass insgesamt keine der gestrichelten Verbindungen betrachtet werden muss.⁵

Für die Implementierung dieses Verfahrens wird während der Suche ein Parameter α verwendet, der auf einer Ebene, auf der maximiert wird, stets den bisher besten Wert speichert. Sobald in einem Nachfolgeknoten ein Wert $\leq \alpha$ ermittelt wurde, kann die Suche in dem weiteren Teilbaum für diesen Knoten abgeschnitten werden. Das gleiche Prinzip kann natürlich auch auf den Ebenen, auf denen minimiert wird, angewendet werden, indem ein Parameter β den zu jedem Zeitpunkt bis dahin niedrigsten Wert speichert. Erreicht ein Nachfolgeknoten einen Mindestwert $\geq \beta$, so kann die Suche in dessen Teilbaum ebenfalls abgebrochen werden. Aufgrund dieser gängigen Bezeichnung der beiden Parameter wird die Methode auch als *Alpha-Beta-Heuristik* (engl. „alpha-beta pruning“) bezeichnet.

17.3.3 Gegenspielermodelle

Das Minimax-Verfahren lässt sich im Prinzip auf beliebige Spiele anwenden, wobei es für jeden Zuge desjenigen Spielers, aus dessen Sicht der Spielbaum erzeugt wurde, das durch diesen Zug garantiert mindestens erreichbare Ergebnis berechnet. Diese „pessimistische“ Vorgehensweise rechtfertigt sich dadurch, dass ein Spieler im schlechtesten Fall immer mit den für ihn ungünstigsten Zügen seitens der anderen Spieler rechnen muss, über die er keine Kontrolle ausübt. Eine oftmals realistischere Bewertung der Züge kann auf der Grundlage eines geeigneten Prognosemodells für das Verhalten der anderen Spieler erfolgen. Dabei zeigt sich, dass die Minimax-Methode immer dann den wahren Wert eines Zugs ergibt, wenn es sich um ein Zweipersonen-nullsummenspiel mit perfekter Information⁶ handelt und der Gegenspieler rational agiert, da unser Verlust gleichzeitig sein Gewinn ist.

Für kooperative Spiele wurde die Alternative der *OM-Suche* (von engl. „Opponent Model“, dt. Gegenspielermodell) entwickelt, das bei der Propagierung von Stellungsbewertungen auf der Annahme beruht, dass alle Spieler bestrebt sind, ihre eigenen Resultate zu maximieren, anstatt das Ergebnis unseres Spielers zu minimieren. Selbst bei Nullsummenspielen kann dies einen Unterschied zu Minimax machen, sobald mehr als nur zwei Spieler beteiligt sind, da sich

⁵ In dem speziellen Fall könnte die Suche natürlich schon unmittelbar nach dem Finden eines Zugs mit Wert 100 abgebrochen werden, da dieser Wert den Regeln nach den überhaupt maximal erzielbaren Wert darstellt.

⁶ Im Fall von GDL bedeutet *Nullsummenspiel*, dass sich die Endergebnisse für die Spieler stets zu 100 addieren, und ein Spiel ist von *perfekter Information*, wenn wie in Tic Tac Toe in jeder Position höchstens ein Spieler mehr als eine Zugmöglichkeit hat.

in solchen Spielen implizite Kooperationen ergeben können: Sind beispielsweise zwei Spieler nur gemeinsam in der Lage, den unmittelbaren Sieg eines Dritten zu verhindern, so würden sie das schon aus Eigeninteresse tatsächlich tun. Das Minimax-Verfahren berücksichtigt diese Möglichkeit nicht (es sei denn, man wäre selbst der Dritte, dessen Sieg es durch gemeinsame Anstrengung aller anderen Spieler zu verhindern gilt). Andererseits ist der Vorteil der OM-Suche, einem Mitspieler rationales Verhalten zu unterstellen, zugleich auch ihr Nachteil, denn sie kann im Unterschied zum Minimax-Verfahren den errechneten Wert eines Zugs nicht garantieren und unter Umständen zu einem schlechteren als den durch einen anderen Zug garantierten Mindestwert führen.

Ein weiteres Problem ergibt sich sowohl bei der Minimax- als auch bei der OM-Suche durch das gleichzeitige Ziehen in manchen Spielen. Tatsächlich fehlt bei der oben beschriebenen OM-Methode ein konkretes Verfahren zur Bewertung eines Baumknotens, in dem mehrere Gegenspieler ihre Züge wählen, da verschiedene Kombinationsmöglichkeiten für die Gegner selbst unterschiedliche Werte haben können. Für diesen Fall ist gar nicht festgelegt, in welchen gemeinsamen Zügen sich rationales Verhalten zeigt und in welchen nicht. Ein ähnliches Problem wirft schon bei Zweipersonenspielen die Trennung der eigenen Züge von denen des Gegenspielers auf, wenn sie auf ein Spiel ohne perfekte Information angewendet wird, in dem also echt gleichzeitig gezogen wird. Diese Trennung unterstellt nämlich, wie aus Abbildung 17.4 deutlich wird, dass der Gegner bei seiner Zugentscheidung bereits weiß, welchen Zug unser Spieler gewählt hat. Für Tic Tac Toe ist dies unkritisch, da in jeder Position jeweils einer der beiden Spieler ohnehin nur eine Möglichkeit hat, aber man denke nur an ein einfaches Spiel wie Stein-Schere-Papier, das wenig sinnvoll wäre, wenn einer der beiden Spieler vor seiner Wahl zunächst den Zug des Anderen zu sehen bekäme. In der Tat würde die Anwendung der oben beschriebene Minimax-Methode auf dieses Spiel den Wert 0 für jeden der drei möglichen Züge unseres Spielers ergeben.

Als Fazit bleibt festzuhalten, dass die Bewertung von Zügen nach der Minimax-Methode in jedem Fall eine einfache und sichere Strategie für eine vorsichtige Spielweise bietet, die aber weder den durch echt gleichzeitiges Ziehen hervorgerufenen Informationsmangel berücksichtigt, noch der Tatsache Rechnung trägt, dass rationale Mitspieler in Mehrpersonen- und insbesondere kooperativen Spielen nicht notwendigerweise danach trachten, ein für unseren Spieler möglichst schlechtes Spielergebnis zu erzielen. Die Konstruktion von Modellen für das Verhalten der anderen Spieler und die Verwendung dieser Modelle bei der Bewertung und Auswahl von Zügen, zum Beispiel das Stellen von Fallen gegen vermutet schwache Gegner, bildet ein interessantes und ergiebiges Forschungsthema auf dem Gebiet der universellen Spielprogramme.

17.4 Stochastische Baumsuche

Die Größe eines Spielbaums wächst im Allgemeinen exponentiell mit zunehmender Tiefe. Aus diesem Grund haben die weitaus meisten Spiele, die für die KI von Interesse sind, einen Suchraum, der nicht einmal annähernd vollständig durchsucht werden kann. Die *stochastische Suche* bietet eine Möglichkeit für eine uninformierte Suche durch größere Spielbäume, bei der nur ein zufällig ausgewählter, kleiner Teil eines Suchraums betrachtet wird. Angewendet auf Spielbäumen bedeutet dies, mittels Zufallszügen ganze Partien bis zum Ende durchzuspielen und die gesammelten Resultate dann zur Einschätzung für den Wert der möglichen Züge in der momentanen Stellung, aus der heraus diese *Simulationen* gespielt wurden, heranzuziehen.

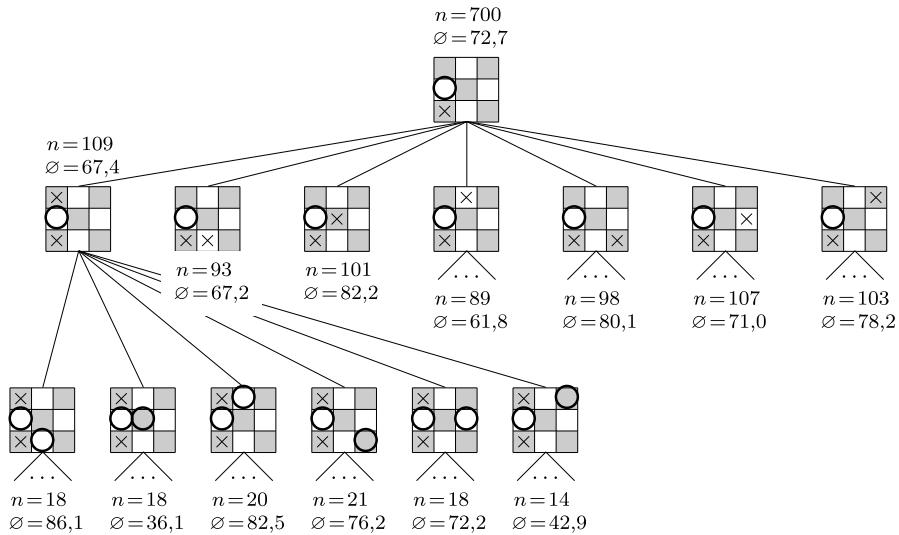


Abbildung 17.6: Beispiel für eine stochastische Baumsuche.

17.4.1 MCT-Suche

Für die Implementierung eines stochastischen Verfahrens in einem universellen Spielprogramm als sogenannte *MCT-Suche* (engl. „Monte Carlo Tree Search“) werden für jede Rolle r eines gegebenen Spiels Schätzwerte $Q^r(s, z)$ für die möglichen Züge z in einer Spielsituation s gesammelt. Das Ergebnis jeder Simulation, in der Spieler r in Situation s den Zug z wählt, fließt dann gewichtet in die Berechnung von $Q^r(s, z)$ mit ein – wurde die Simulation gewonnen, so erhöht sich der Wert, wurde sie verloren, verringert er sich.

Abbildung 17.6 illustriert, zu welcher Einschätzung eine solche zufallsgesteuerte Simulation für eine Beispieldposition in Tic Tac Toe kommen kann. Am Ende dieser stochastischen Suche kann Spieler r in der zuoberst gezeigten Stellung s denjenigen Zug z wählen, der den höchsten Schätzwert $Q^r(s, z)$ erzielt hat. Das Ergebnis in Abbildung 17.6 kann also dahingehend interpretiert werden, dass das Markieren des Mittelfelds in der gegebenen Position die größten Chancen bietet.

17.4.2 UCT-Bonus

Die Aussagekraft der stochastischen Baumsuche wird deutlich erhöht, wenn Züge, die im Verlauf der Suche vielversprechend erscheinen, verstärkt simuliert werden. Um dies zu realisieren, werden die Züge eines Spielers in einer Spielstellung s nur solange zufällig gewählt, bis jeder seiner legalen Züge in s mindestens einmal selektiert wurde. Landet die nächste Simulation wieder in s , so wird für den Spieler jetzt derjenige Zug gewählt, der den bis dahin besten Schätzwert erreicht hat. Um jedoch einen anderen guten Zug, der nur aus Zufall in der ersten Simulation einen schlechten Wert erzielt hat, nicht gänzlich zu ignorieren, werden die Schätz-

werte um einem sogenannten *UCT-Bonus*⁷ angereichert, der umso höher ist, je *seltener* ein Zug simuliert wurde.

Konkret seien $N(s)$ die Gesamtzahl der bislang durchgeführten Simulationen, die durch Spiel-situation s liefen, und $N^r(s, z)$ die Anzahl dieser Simulationen, bei denen Spieler r Zug z in s gewählt hat. Wenn dann die stochastische Suche das nächste Mal in s landet, so wird der zu wählende Zug für den betreffenden Spieler nach folgender Formel bestimmt:

$$\arg \max_z \left\{ Q^r(s, z) + C \cdot \sqrt{\frac{\log N(s)}{N^r(s, z)}} \right\}$$

Hierbei stellt der Wurzelausdruck den UCT-Bonus dar, und C ist eine frei wählbare Konstante zu dessen Gewichtung bei der Auswahl desjenigen Zugs, der den Wert in den geschweiften Klammern maximiert. Je öfter ein Zug bereits betrachtet wurde, desto geringer fällt sein UCT-Bonus aus, da der Wert $N^r(s, z)$ im Nenner auftaucht. Umgekehrt steigt damit auch gleichzeitig der UCT-Bonus aller anderen Züge, da sich der Wert $N(s)$ in deren Zähler mit jeder Simulation durch s erhöht.

Die statistischen Werte, die während der UCT-Suche ermittelt werden, werden in einem Spielbaum gespeichert. Aus Gründen der Speichereffizienz wird dabei der Baum nach jeder Simulation immer nur um die erste noch nicht im Baum enthaltene Position auf dem zufällig generierten Pfad erweitert. Da im Verlauf der Suche vielversprechende Züge verstärkt simuliert werden, wächst der gespeicherte Spielbaum asymmetrisch: Ein tiefer Teilbaum deutet auf gutes Spiel hin, während der Baum im uninteressanten Teil flach bleibt.

Abbildung 17.7 illustriert, wie sich die stochastische Suche mit Hilfe des UCT-Verfahrens stärker zu vielversprechenden Zügen hin konzentrieren lässt, was sich vor allem an den Knoten

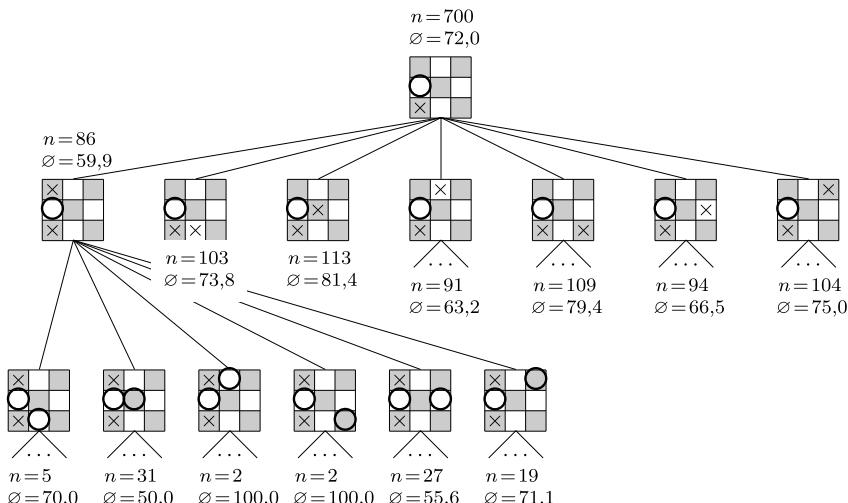


Abbildung 17.7: Die Baumsuche aus Abbildung 17.6 nach der UCT-Methode ($C = 1$).

⁷ Die Bezeichnung ist aus dem engl. Begriff „upper confidence bound tree search“ für dieses Verfahren abgeleitet.

der dritten Ebene zeigt (wobei zu beachten ist, dass gute Züge für den zweiten Spieler gleichbedeutend mit einem niedrigen Durchschnittswert sind). Die UCT-Methode verdeutlicht auch, weshalb ein universelles Spielprogramm bei der stochastischen Baumsuche nicht nur die Abschätzung der eigenen Züge vornimmt, sondern zugleich die Ergebnisse einer Zufallspartie aus Sicht aller anderen Spieler speichert. Auf diese Weise werden nämlich im weiteren Verlauf der Suche nach obiger Formel auch verstärkt die besseren Gegenzüge gewählt, was wiederum zu einer realistischeren Einschätzung der Qualität der eigenen Züge führt.

17.4.3 Optimierungen

Ein Merkmal der zufallsgesteuerten Suche durch einen Spielbaum ist, dass der Schätzwert für einen Knoten umso unsicherer wird, je tiefer sich dieser im Baum befindet, da die Wahrscheinlichkeit, dass ein zufällig gewählter Pfad durch einen bestimmten Punkt führt, mit zunehmender Tiefe geringer wird. Aus diesem Grund ist es sinnvoll, das Ergebnis einer Simulation umso höher zu bewerten, je weniger Züge bis zum Spielende benötigt wurden. Eine allgemein in stochastischen Suchverfahren gängige Vorgehensweise ist die Verwendung eines *Diskontierungsfaktors* $\gamma < 1$, um den der berechnete Wert einer Simulation in jedem Schritt reduziert wird. Konkret heißt dies für die UCT-Suche, dass beispielsweise eine Simulation, die von der Situation s ausgehend noch fünf Züge bis zum Partieende benötigt und dabei ein Ergebnis von 100 (Gewinn) ergeben hat, für den in s gewählten Zug den Wert $\gamma^5 \cdot 100$ erhält. Eine andere Simulation, die zu dem gleichen Endergebnis in nur zwei Schritten führt, erhält hingegen den Wert $\gamma^2 \cdot 100$, wobei γ ein geeignet zu wählender, reduzierender Multiplikator ist, zum Beispiel $\gamma = 0,999$.

Ein spezieller Vorteil stochastischer Suchverfahren ist ihre vergleichsweise einfache Parallelisierbarkeit. Da die Qualität einer stochastischen Abschätzung im Allgemeinen mit zunehmender Zahl an Stichproben immer besser wird, kann auch der Wert eines Zugs umso besser abgeschätzt werden, je mehr Zufallspartien mit ihm in der zur Verfügung stehenden Zeit simuliert werden können. Dabei sind die einzelnen Simulationen weitgehend voneinander unabhängig. Im einfachsten Fall werden mehrere UCT-Suchen parallel und ohne Kommunikation untereinander durchgeführt, so dass erst am Ende die Ergebnisse zusammengeführt werden müssen. Eine andere Möglichkeit ist die Verwendung einer von mehreren Prozessen gemeinsam verwalteten Tabelle für die Werte $Q^r(s, z)$ und $N^r(s, z)$. In jedem Fall eignet sich die UCT-Suche besonders gut für die Verwendung einer parallelen Computerarchitektur.

17.4.4 Grenzen

Die stochastische Simulation ist ein probates Mittel für ein universelles Spielprogramm, ohne weiteres Zutun mittels uninformerter Suche ein neues Spiel zu spielen. Dies gilt insbesondere für Spiele, bei denen es einem System nicht leicht fällt, eine geeignete Strategie zu entwickeln.

Für Spiele, in denen es in den meisten Positionen nur wenig gute, dafür aber viele schlechte Züge gibt, kann jedoch das zufallsgesteuerte Ausspielen von Partien leicht zu einer völlig falschen Einschätzung von Zügen führen. Dies lässt sich anhand eines schönen Spiels aus früheren Wettbewerben für universelle Spielprogramme verdeutlichen: In der in Abbildung 17.8 gezeigten Schachvariante beginnen beide Seiten mit Springer auf allen Feldern ihrer zwei Grundreihen. Ziel des Spiels ist es, mit einem Springer die gegenüberliegende letzte Reihe zu erreichen. Eine stochastische Suche in der abgebildeten Stellung wird früh einen hohen Wert für den per Pfeil

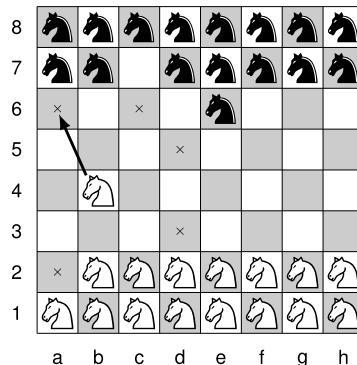


Abbildung 17.8: Eine Partie Knight Breakthrough nach zwei Zügen. (Für Nichtschachspieler verdeutlichen die markierten Felder die Zugmöglichkeiten eines Springers am Beispiel des $\mathbb{Q}b4$.)

angedeuteten Zug $\mathbb{Q}b4-a6$ berechnen, da dieser in fast allen Simulationen zu einem schnellen Sieg führt. Offensichtlich ist dies aber kein guter Zug, da Schwarz einfach mit $\mathbb{Q}b8 \times a6$ den Springer schlägt und Weiß dadurch ersatzlos eine Figur verliert. Da dies die einzige Antwort von Schwarz ist, die nicht sofort verliert, wird dieser Gegenzug nach einiger Zeit bei der UCT-Suche bevorzugt gewählt und damit die Schwäche des ersten Zugs erkannt. Allerdings erfordert dies eine genügend große Anzahl an Simulationen, wie sie üblicherweise nur für Knoten weit oben im Spielbaum durchgeführt werden können. Hingegen werden für Spielsituationen weiter unten im Baum schlechte Züge, die nur durch einen oder wenige Gegenzüge widerlegbar sind, von einer stochastischen Suche meist irrtümlich als gut bewertet.

17.5 Heuristische Suche

Die meisten Programme für spezielle Spiele wie Schach verwenden eine *heuristische Suche* durch einen Spielbaum bis zu einer begrenzten Tiefe. Die erreichbaren Spielpositionen werden dann durch eine *Bewertungsfunktion* evaluiert. Auf diese Weise entsteht ein Teilbaum, bei dem die Suche nach einer bestimmten Zahl von Zügen abgeschnitten wird und dessen Blätter dennoch alle einen numerischen Wert tragen. Diese Werte können auf die gleiche Weise nach oben propagiert werden, wie bei einer vollständigen Baumsuche, also beispielsweise durch Anwendung des Minimax-Verfahrens (vgl. Abschnitt 17.3).

Die Qualität des Ergebnisses einer solchen heuristischen Suche ist selbstverständlich maßgeblich von der Güte der Zwischenevaluierung abhängig. In Programmen für spezielle Spiele können auf das jeweilige Spiel zugeschnittene Bewertungsfunktionen verwendet werden, und so besteht in der Praxis ein Großteil der Entwicklungsarbeit etwa eines Schachcomputers darin, geeignete Kriterien für eine Stellungsbewertung zu finden und zu gewichten – beispielsweise Figurenwerte, Zentrumsbeherrschung, Königssicherheit, Bauernstruktur usw. Einem universellen Spielprogramm kann solches Expertenwissen nicht vorgegeben werden, sondern muss von dem System selbst anhand der Spielregeln generiert werden. Das selbstständige Konstruieren, d.h. Erlernen einer geeigneten Bewertungsfunktion, ist eine der größten und interessantesten Herausforderungen bei der Konstruktion von guten und vielseitig verwendbaren, universellen

Spielprogrammen. Im Folgenden betrachten wir zwei einfache Beispiele für vollautomatische Heuristiken, die prinzipiell auf jedes Spiel angewendet werden können.

17.5.1 Mobilitätsheuristik

Die sogenannte Mobilitätsheuristik basiert auf der Annahme, dass eine Spielsituation, in der dem eigene Spieler eine große Auswahl an Zügen zur Verfügung steht, im Allgemeinen besser ist als eine Position, in der seine Möglichkeiten stark beschränkt sind. In vielen Spielen kann dieses Kriterium ein erster Anhaltspunkt für die Güte einer Stellung sein. So führt etwa in Brettspielen wie Dame, Mühle oder Schach ein Übergewicht an Figuren, speziell an solchen mit höherer Mobilität wie eine Dame gegenüber einem Springer im Schach, in den meisten Stellungen zu einem Mehr an Zügen. Gleiches gilt für Bewertungsmaßstäbe wie Zentrumsbeherrschung oder Kontrolle über ein Spielfeld usw.

Die Mobilitätsheuristik ist sehr einfach zu implementieren. In einer zu bewertenden Position müssen dazu lediglich für jeden Spieler alle legalen Züge berechnet und dann die Zahl der gegnerischen Zugmöglichkeiten von der Zahl der eigenen subtrahiert werden. Abbildung 17.9 zeigt am Beispiel aus Abbildung 17.8, wie zwei mögliche Stellungen zwei Züge später anhand des Mobilitätskriteriums unterschiedlich bewertet werden.

Während Mobilität oft ein einfacher Indikator für den Wert einer Position sein kann, gibt es umgekehrt auch viele bekannte Spiele, in denen dieses Kriterium keine Aussagekraft hat. Ein offensichtliches Beispiel ist Tic Tac Toe, bei dem die Anzahl der freien Felder eindeutig durch die Zahl der gemachten Züge bestimmt und unabhängig von der Güte einer Stellung ist. Ein anderes Beispiel bildet das in Abbildung 17.10 gezeigte Schiebepuzzle, das als Eipersonenspiel einen Spezialfall für universelle Spielprogramme darstellt und in dem die Zahl der möglichen Züge, die immer zwischen 2 und 4 liegt, offensichtlich keinen Hinweis darauf gibt, wie weit ein Zwischenzustand vom Ziel entfernt ist. In einigen Spielen kann das Mobilitätskriterium sogar kontraproduktiv sein, wie etwa wenn beim Damespiel Schlagzwang besteht. Muss nämlich ein Spieler den Regeln zufolge schlagen, sobald ihm ein Stein angeboten wird, so schränkt dies seine Zugmöglichkeiten stark ein, was aber natürlich in den meisten Fällen von Vorteil ist, wenn

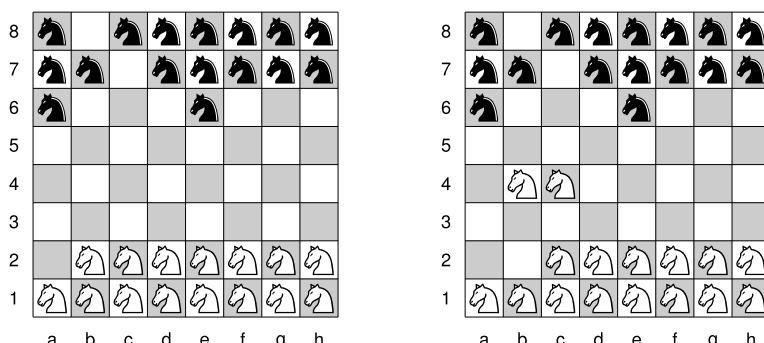


Abbildung 17.9: Links die Stellung aus Abbildung 17.8 nach der Zugfolge $\mathbb{Q}b4-a6, \mathbb{Q}b8xa6$; rechts die Stellung nach der alternativen Fortsetzung $\mathbb{Q}b2-c4, \mathbb{Q}b8-a6$. In der linken Stellung hat Weiß 38 legale Züge, in der rechten 46. Letztere ist daher aus seiner Sicht bei Anwendung der Mobilitätsheuristik deutlich besser bewertet.

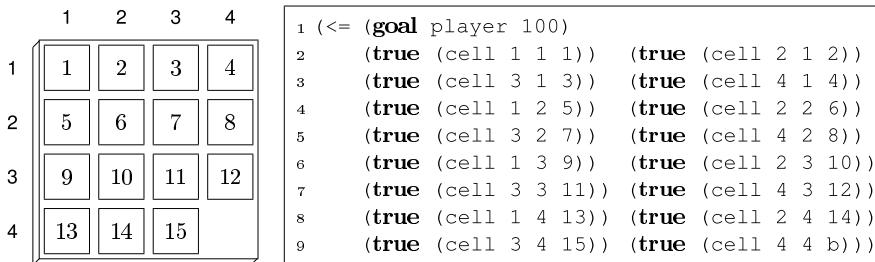


Abbildung 17.10: Das Schiebepuzzle (auch 15-Puzzle genannt) mit Zieldefinition.

dadurch ein Stein erobert werden kann. In solchen Spielen führt die Anwendung der Mobilitätsheuristik geradewegs zum Verlust, wenn der Gegner durch wiederholtes Opfern eigener Steine vermeintlich in die Ecke gedrängt wird.⁸ Ähnlich irreführend ist die Mobilitätsheuristik bei vielen Spielen, wenn sie in der sogenannten Misère-Variante gespielt werden, bei der Verlieren das Ziel ist (zum Beispiel Schlagschach). Die Entscheidung jedoch, ob die Mobilitätsheuristik in einem neuen Spiel sinnvoll ist, erfordert von einem universellen Spielprogramm die Erlangung domänenspezifischen Wissens.

17.5.2 Zielheuristiken

Die sogenannten Zielheuristiken basieren auf dem Prinzip, den Abstand von einem Zwischenzustand zu einem vorgegebenen Spielziel abzuschätzen. Im Folgenden betrachten wir ein konkretes Verfahren, ein solches Abstandsmaß anhand der rein logischen Beschreibung eines Spielziels zu berechnen. Hierzu wird mit Hilfe der *Fuzzy-Logik* der *Erfüllungsgrad* der Zielformel als ein numerischer Wahrheitswert zwischen 0 und 1 bestimmt (vgl. Kap. **Fuzzy**). Diese Grundidee kann leicht anhand des Schiebepuzzles aus Abbildung 17.10 veranschaulicht werden, bei dem das Spielziel durch Und-Verknüpfung der Zielpositionen der einzelnen Zahlen gegeben ist. Die Güte eines Zwischenzustands lässt sich gut daran abschätzen, wie viele dieser Teilziele bereits erfüllt sind. Dies bietet zwar keine perfekte Bewertung, da im Verlauf einer Lösung immer wieder einzelne Zahlen temporär aus ihren Zielpositionen verschoben werden müssen, aber in Verbindung mit einer tiefenbeschränkten Suche kann das Schiebepuzzle mit einer Zielheuristik vergleichsweise schnell gelöst werden.

Zur Implementierung einer auf der Fuzzy-Logik basierten Zielheuristik muss zunächst ein Grundwahrheitswert p mit $0,5 < p < 1$ festgelegt werden. Dieser wird an alle Stellungsmerkmale, die in einem Zwischenzustand erfüllt sind, vergeben. Entsprechend erhalten alle in dem Zustand nicht erfüllten Stellungsmerkmale den Wahrheitswert $1 - p$. Bei der in Abbildung 17.11(a) gezeigten Position gelten beispielsweise mit $p = 0,9$ folgende Wahrheitswerte:

```

(true (cell 1 1 x)) :0,9
(true (cell 2 2 x)) :0,1
(true (cell 3 3 x)) :0,9

```

⁸ Dazu eine kleine Anekdote aus der Weltmeisterschaft 2006, bei der ausgerechnet im Finale genau dies passierte: In einer Partie Dame auf einem zylindrisch geformten Brett bot Weiß (*Cluneplayer*, USA) in rascher Folge seine eigenen Steine zum Schlagen an, wohlweislich darauf achzend, nicht zum Zurückschlagen gezwungen zu sein. Schwarz (*Fluxplayer*, Deutschland) gewann die Partie leicht.

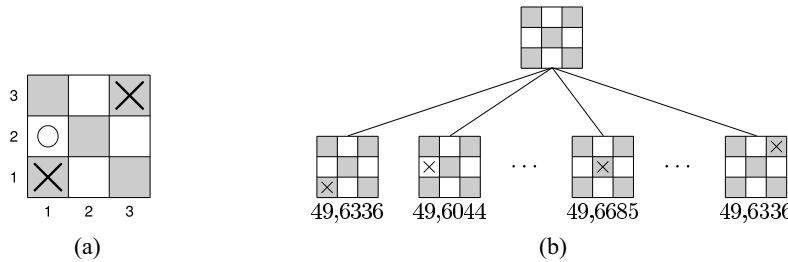


Abbildung 17.11: (a) Eine Beispielposition in Tic Tac Toe. (b) Die Bewertung der Zielheuristik für die verschiedenen Eröffnungsziege.

In der Fuzzy-Logik basiert die Bestimmung des Wahrheitsgehalts einer beliebigen logischen Formel auf einer Berechnungsvorschrift für die Konjunktion (auch „*t*-Norm“ genannt, vgl. Abschnitt Kap. **Fuzzy**), wie beispielsweise die einfache Multiplikation der Wahrheitswerte. Auf diese Weise ergibt sich mit den gerade genannten Beispielen für den Bedingungsteil der Regel

```
(<= (diagonal ?w)
    (true (cell 1 1 ?w)) (true (cell 2 2 ?w)) (true (cell 3 3 ?w)))
```

mit $?w = x$ der Wahrheitswert 0,081. Diese Berechnung kann für beliebige Formeln im Bedingungsteil einer GDL-Spielregel wie folgt verallgemeinert werden:

$$\begin{aligned} \text{eval}(\text{(not } A\text{)}) &= 1 - \text{eval}(A) \\ \text{eval}(\text{(or } A \text{ } B\text{)}) &= \text{eval}(A) + \text{eval}(B) - \text{eval}(A) \cdot \text{eval}(B) \end{aligned}$$

Die Disjunktion wird auch bei der Auswertung mehrerer Regeln für ein und dieselbe Konklusion verwendet; so ergibt sich beispielsweise bei der in Abbildung 17.11(a) gezeigten Position für die Bedingung der zweiten Regel zur Diagonale, also

```
(<= (diagonal ?w)
    (true (cell 1 3 ?w)) (true (cell 2 2 ?w)) (true (cell 3 1 ?w)))
```

für $?w = x$ der Wert 0,001, so dass sich für $(\text{diagonal } x)$ insgesamt (per Disjunktion) der Wahrheitsgehalt 0,081919 errechnet. Diese Berechnung kann fortgesetzt werden, um mit Hilfe der weiteren Spielregeln aus Abschnitt 17.1 den Wahrheitswert für das Hilfsprädikat $(\text{line } x)$ und damit auch für $(\text{goal xplayer } 100)$ zu bestimmen, wobei sich in unserer Beispielposition 0,116296 ergibt.

Schließlich können noch die verschiedenen erreichbaren Ziele entsprechend ihres Zielwerts gewichtet und dann mit der Berechnungsvorschrift für die Disjunktion zu einem Gesamtwert zusammengefasst werden; im Fall von Tic Tac Toe also nach folgender Formel:⁹

$$\begin{aligned} 100 \cdot [\text{eval}((\text{goal xplayer } 100)) \cdot 1 \\ + \text{eval}((\text{goal xplayer } 50)) \cdot 0,5 \\ - \text{eval}((\text{goal xplayer } 100)) \cdot 1 \cdot \text{eval}((\text{goal xplayer } 50)) \cdot 0,5] \end{aligned}$$

⁹ Um bei dieser Berechnung alle erreichbaren Zielwerte berücksichtigen zu können, muss im Allgemeinen zuvor der Wertebereich der Variablen in den jeweiligen Regeln für das Schlüsselwort `goal` ermittelt werden; dies kann mit Hilfe einer im nachfolgenden Abschnitt 17.6 beschriebenen Methode geschehen.

Für die Stellung in Abbildung 17.11(a) errechnet sich so eine Gesamtbewertung von 12,5969. Abbildung 17.11(b) zeigt die auf dieselbe Weise ermittelten Werte der Zielheuristik am Beispiel der möglichen ersten Züge. Ganz ohne Suche wird also nach der Zielheuristik das Mittelfeld gewählt werden. Zudem ist die Markierung eines Eckfelds stärker als die eines Randfelds.

Allgemein ist die Zielheuristik ein geeignetes Mittel für solche Spiele, in denen das Gesamtziel durch das schrittweise Erfüllen einzelner Teilziele erreicht werden kann. Dies kann auch zum Beispiel für solche Spiele gelten, die die vollständige Eliminierung gegnerischer Figuren oder die vollständige Besetzung eines Territoriums usw. zum Ziel haben, falls dies schrittweise geschehen kann und aus der Zielformel hervorgeht.

Dabei berücksichtigt die einfache Zielheuristik jedoch nicht, wie schwer es ist, von einem das Ziel in Teilen erfüllenden Zustand zu dem endgültigen Ziel zu gelangen, oder ob dies überhaupt möglich ist. So hat etwa in Tic Tac Toe eine einzelne Reihe, Spalte oder Diagonale, bei der genau zwei Felder mit der eigenen Markierung versehen sind, den oben berechneten Wert 0,081 unabhängig davon, ob das dritte Feld frei oder aber bereits vom Gegenspieler belegt ist, also gar nicht mehr vervollständigt werden kann. Darauf hinaus liefert die Zielheuristik keine sinnvollen Werte bei Spielen mit einem sehr speziellen Ziel wie beispielsweise Schachmatt, aus dessen reiner Definition in keiner Weise abgeleitet werden kann, zu welchem Grad das Ziel in einer Zwischenstellung erfüllt ist. Für solche Fälle lassen sich mehrere Heuristiken kombinieren, wie beispielsweise die Mobilitäts- und die Zielheuristik: Wenn eine dieser beiden keine Unterscheidung zwischen verschiedenen Zuständen vornimmt, die andere hingegen schon, dann wird durch eine Kombination automatisch nach derjenigen gespielt, die zu unterscheiden vermag. Im Allgemeinen erfordert jedoch eine Kombination immer eine geeignete Gewichtung der einzelnen Bewertungskriterien, was wiederum die Generierung von domänenspezifischem Wissen erfordert.

17.5.3 Optimierungen

Das Standardverfahren bei der Spielbaumsuche mit heuristischer Bewertungsfunktion verwendet die schon in Abschnitt 17.3 erwähnte, iterative Tiefensuche, bei der die maximale Suchtiefe schrittweise angehoben wird. Dies hat bei der heuristischen Baumsuche den zusätzlichen Vorteil, dass nach jeder durchsuchten Tiefe die Knoten anhand ihrer Zwischenbewertung sortiert werden können, so dass im nächsten Schritt, also in größerer Tiefe, zunächst die vielversprechenden Züge betrachtet werden. Mit dieser sogenannten „History Heuristics“ kann zum einen die Zahl an Alpha-Beta-Schnitten deutlich erhöht werden (vgl. Abschnitt 17.3). Zum anderen ist damit sichergestellt, dass die voraussichtlich guten Züge bereits analysiert wurden, wenn die Suche aus Zeitgründen auf der maximal erreichbaren Tiefe abgebrochen werden muss, ohne alle Züge betrachtet zu haben.

Das Prinzip der iterativen Tiefensuche kann darüber hinaus verallgemeinert werden, indem vielversprechende Züge tiefer analysiert werden, so dass also für verschiedene Züge je nach deren Zwischenbewertung ein unterschiedlicher Suchhorizont gewählt wird. Dies ist insbesondere dann hilfreich, wenn bei einem oder mehreren dieser Züge bis zum Ende gesucht werden kann, da die Bewertung von Terminalzuständen immer anhand der Spielregeln selbst erfolgt und nicht mehr auf die unsichere, heuristische Bewertungsfunktion zurückgegriffen werden muss. Auf diese Weise lässt sich dann auch feststellen, ob die Bewertungsfunktion ein guter Indikator ist oder ob die scheinbar vielversprechenden Züge in der Realität doch nicht so gut sind.

17.6 Wissen

Eine ebenso große Herausforderung wie das selbstständige Konstruieren einer guten Bewertungsfunktion ist für universelle Spielprogramme das Erlangen von Wissen über ein neues Spiel – Wissen, das zwar implizit bereits in den reinen Spielregeln steckt, aber erst gelernt, also explizit gemacht werden muss, bevor es für eine Strategie verwendet werden kann. Die Spanne reicht dabei von elementarem Wissen, wie etwa ob es sich um ein Nullsummen- oder ein kooperatives Spiel handelt, bis zu ganze Datenbanken füllendem Wissen, wie etwa das im Laufe der Jahrhunderte von Experten (und zunehmend auch Computern) gesammelte Schachwissen. Entsprechend vielfältig sind die Verwendungsmöglichkeiten in einem universellen Spielprogramm: Einfaches Wissen dient der Auswahl geeigneter Suchmethoden wie der Alpha-Beta-Heuristik für den Fall, dass ein Spiel als Zweipersonennullsummenspiel mit abwechselnden Zügen erkannt wurde. Strukturelles Wissen wie beispielsweise über Symmetrien in einem Spiel kann jede Suche beschleunigen. Wissen über den Wert unterschiedlicher Spielfiguren oder bestimmter Bereiche eines Spielfelds können die Grundlage für Bewertungsfunktionen und andere Suchheuristiken bilden usw.

Während es nahezu unbegrenzte Möglichkeiten gibt, Wissen über ein neues Spiel zu erlernen und zu verwenden, kann im Rahmen dieser Einführung lediglich auf zwei Verfahren eingegangen werden. Beide sind leicht zu implementieren und zugleich fast immer verwendbar.

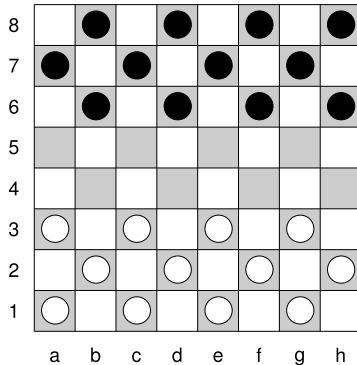
17.6.1 Domänenanalyse

Bei der Domänenanalyse werden Wertebereiche aller in einer Spielbeschreibung enthaltenen Prädikate und Funktionen ermittelt. Wie bereits in Abschnitt 17.5.2 erwähnt, ist dies zum Beispiel für die vollständige Berechnung der Zielheuristik vonnöten. Im Allgemeinen liefert die Kenntnis über die Wertebereiche einzelner Ausdrücke sehr grundlegendes Wissen über ein Spiel. So charakterisiert beispielsweise die Menge der möglichen Argumente für das Schlüsselwort `true` alle im Laufe eines Spiels potenziell auftretenden Stellungsmerkmale, womit die erreichbaren Zustände eingegrenzt werden können. Auf diese Weise lassen sich oft einzelne Elemente eines Spiels, wie etwa Spielfelder und Figuren identifizieren; siehe dazu Abbildung 17.12.

Für die Domänenanalyse werden die Abhängigkeiten der Argumente und Variablen in den einzelnen Spielregeln untersucht. Dies geschieht mittels der Konstruktion des sogenannten *Domänengraphen*, dessen Knotenmenge zum einen alle Prädikate, Funktionen und Konstanten beinhaltet, die in der Spielbeschreibung vorkommen, und zum anderen die einzelnen Argumentpositionen der Prädikate und Funktionen enthält, also zum Beispiel `c@1`, `c@1[1]`, `c@1[2]` und `c@1[3]` für die entsprechende dreistellige Funktion in der Tic-Tac-Toe-Beschreibung aus Abschnitt 17.1. Gerichtete Kanten zwischen den Knoten geben die aus den Spielregeln ersichtlichen Abhängigkeiten an.

Definition 17.6.1 (Domänengraph). *Der Domänengraph zu einer GDL-Beschreibung G ist der kleinste Graph (V, E) mit Knoten V und gerichteten Kanten E , für den gilt:*

1. $c \in V$ für jede Konstante c , die in G verwendet wird.
2. $p, p[1], \dots, p[n] \in V$ für jedes n -stellige Prädikats- oder Funktionszeichen p , das in G verwendet wird.



```

1 (init (cell a 1 white_man))
:
24 (init (cell h 8 black_man))
:
30 (next_file a b) ... (next_file g h)
31 (next_rank 1 2) ... (next_rank 7 8)
:
42 (<= (next (cell ?x 8 white_king)))
43   (does white (move ?u ?v ?x 8)))
44 (<= (next (cell ?x 1 black_king)))
45   (does black (move ?u ?v ?x 1)))
:

```

Abbildung 17.12: Angewendet auf die hier auszugsweise gezeigten Regeln des Damespiels kann die Domänenanalyse für das 3-stellige Stellungsmerkmal `cell` den Wertebereich $\{a, \dots, h\} \times \{1, \dots, 8\} \times \{\text{white_man}, \text{black_man}, \text{white_king}, \text{black_king}\}$ ermitteln. Darüber hinaus kann leicht festgestellt werden, dass auf den ersten beiden Argumenten durch die Regeln 30 und 31 jeweils eine Art Ordnungsrelation definiert ist. Dies könnte ein universelles Spielprogramm zu der Hypothese veranlassen, dass `cell` ein zweidimensionales Spielfeld mit insgesamt vier verschiedenen Figurtypen codiert.

3. $f \rightarrow p[i] \in E$ für jedes Vorkommen einer Funktion (bzw. Konstante) f im i -ten Argument eines Ausdrucks p im Kopf einer Regel aus G .
4. $p[j] \rightarrow q[i] \in E$ für jedes Vorkommen einer Variablen als i -tes Argument eines Ausdrucks q im Kopf und als j -tes Argument eines Ausdrucks p im Bedingungsteil ein und derselben Regel aus G .
5. $\text{init}[i] \rightarrow \text{true}[i]$, $\text{next}[i] \rightarrow \text{true}[i]$, $\text{legal}[i] \rightarrow \text{does}[i]$, $\text{legal}[2] \rightarrow \text{does}[2] \in E$.

Als Beispiel zeigt Abbildung 17.13(a) einen Ausschnitt aus dem Tic-Tac-Toe-Domänengraphen, der nach Punkt 1 und 2 aus der Regel

```

21 (<= (next (cell ?m ?n x))
22   (does xplayer (mark ?m ?n)))

```

entsteht. Abbildung 17.13(b) zeigt einen weiteren Ausschnitt aus diesem Domänengraphen, aus dem sich leicht die Wertebereiche für die aufgeführten Prädikate und Funktionen ablesen lassen:

```

cell:    {1, 2, 3} × {1, 2, 3} × {b, x, o}
row:     {1, 2, 3} × {b, x, o}
column:  {1, 2, 3} × {b, x, o}
diagonal:{b, x, o}
line:    {b, x, o}

```

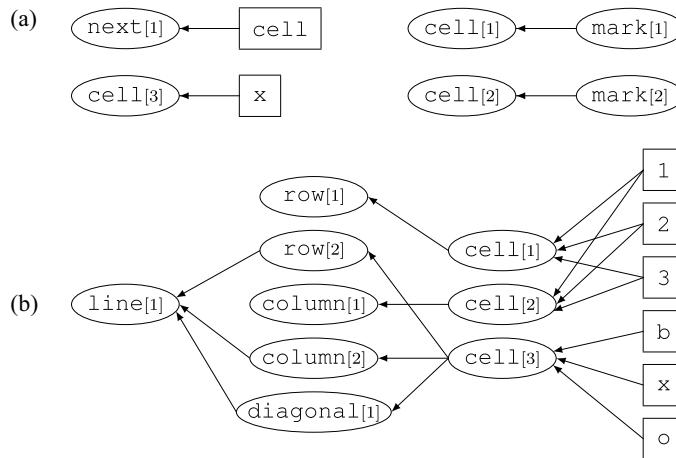


Abbildung 17.13: (a) Die Kanten im Domänengraphen für Tic Tac Toe, die sich aus Regel 21–22 ergeben; (b) Teilgraph, der sich aus den Regeln 37–58 nebst 3–11 und 21–25 (vgl. Abschnitt 17.1) ergibt.

17.6.2 Regelstrukturanalyse

Die Analyse der Struktur von Spielregeln hat das Ziel, von deren konkreter Syntax zu abstrahieren, um so stärker die Bedeutung einer Regel herauszuarbeiten. Damit können unterschiedliche Zwecke verbunden sein. Beispielsweise lassen sich auf diese Weise verschiedene Formulierungen einer und derselben Regel miteinander vergleichen, was ein universelles Spielprogramm in die Lage versetzen kann, ihm bekannte Strategien auf ähnliche Spiele, die lediglich in einem neuem Gewand daher kommen, anzuwenden. Die Regelstrukturanalyse bildet darüber hinaus die Grundlage für die automatische Erkennung von Symmetrien in einem Spiel.

Einige Symmetrien des Tic-Tac-Toe-Spiels sind in Abbildung 17.14 verdeutlicht: Zwei Spiel-situationen sind symmetrisch, wenn sie durch Drehung des Spiefelds ineinander überführt werden können oder durch diagonale Spiegelung entstehen. Um solche Symmetrien berechnen zu können, muss ein universelles Spielprogramm die gesamte Beschreibung eines Spiels daraufhin analysieren, ob bestimmte Elemente systematisch vertauscht werden können, ohne dass sich die Bedeutung der Spielregeln ändert. Dies schließt insbesondere auch das Spielziel mit ein, denn wenn beispielsweise die Standardregeln dahingehend abgewandelt werden, dass nur eine vollständig belegte Reihe und keine Spalte zum Gewinn führt, dann wird dadurch die Spiegelsymmetrie aus Abbildung 17.14 gebrochen (nicht jedoch die 180-Grad-Drehsymmetrie).

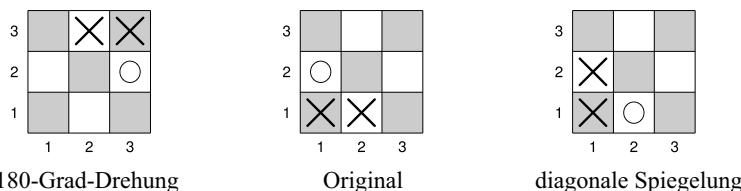


Abbildung 17.14: Symmetrische Stellungen in Tic Tac Toe.

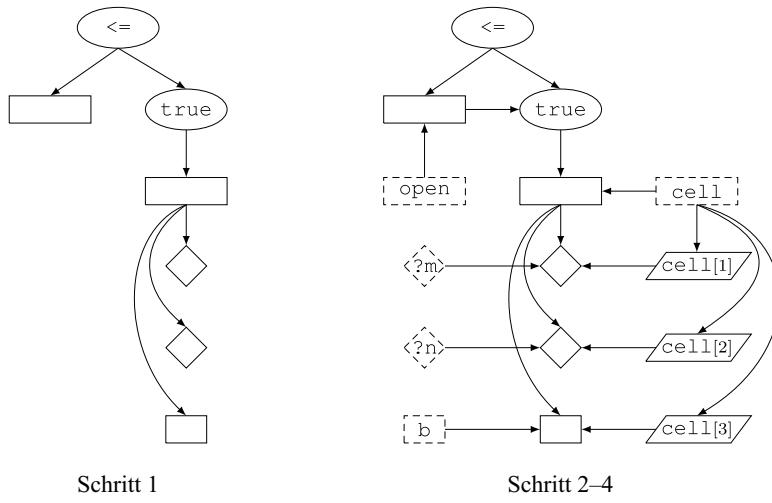


Abbildung 17.15: Die schrittweise Konstruktion eines Regelgraphen anhand des Beispiels `(<= open (true) (cell ?m ?n b))`. Unterschiedliche Formen der Knoten stehen für unterschiedliche Farben nach Schritt 5.

Die Regelstrukturanalyse basiert ähnlich wie die Domänenanalyse auf der Konstruktion eines Graphen aus den Spielregeln heraus. Dieser sogenannte *Regelgraph* wird schrittweise wie folgt gebildet (Abbildung 17.15 verdeutlicht die Konstruktion anhand einer Beispielregel):

1. Für jedes Vorkommen eines logischen Operators, eines Prädikats- und Funktionssymbols sowie von Variablen und Konstanten in einer Regel wird ein Knoten des entsprechenden Typs erzeugt, und es werden Kanten hinzugefügt, die die Operator-, Prädikats- und Funktionsknoten mit ihren jeweiligen Argumenten verbinden.
2. Für alle Prädikats- und Funktionssymbole, soweit sie nicht GDL-Schlüsselwort sind, werden Knoten für die einzelnen Argumentpositionen eingefügt und ebenfalls mit den Argumentknoten verbunden; für GDL-Schlüsselwörter und logische Operatoren werden hingegen zwischen den Argumenten selbst gerichtete Verbindungskanten entsprechend ihrer Reihenfolge eingefügt.
3. Für jede Variable bzw. Konstante wird ein weiterer Knoten hinzugefügt, der mit jedem ihrer Vorkommen verbunden wird.
4. Für jedes Prädikats- und Funktionssymbol, soweit es nicht GDL-Schlüsselwort ist, wird ein weiterer Knoten hinzugefügt, der mit jedem seiner Vorkommen und zusätzlich mit den Knoten für die Argumentpositionen verbunden wird.
5. Schließlich werden alle Knoten *gefärbt*, wobei für alle GDL-Schlüsselwörter und logischen Operatoren unterschiedliche Farben verwendet werden, während bei den weiteren Knoten farblich nur zwischen den fünf möglichen Typen (Prädikat, Funktion, Variable, Konstante, Argumentposition) unterschieden wird.

Auf diese Weise entsteht ein recht komplexer Graph, der alle notwendigen Informationen enthält, um eine Regel strukturell zu erfassen. Dadurch lassen sich beispielsweise zwei syntaktisch unterschiedliche Regelbeschreibungen miteinander vergleichen. Hierzu wird nach *Isomorphismen*

gesucht, also nach Transformationen der Knotenmenge eines Graphen in die Knotenmenge eines anderen, bei denen Kantenstruktur und Färbung der Knoten erhalten bleiben. Lassen sich auf diese Weise die einzelnen Graphen zweier unterschiedlicher Regelmengen aufeinander abbilden, so beschreiben sie das gleiche Spiel mit unterschiedlichen Symbolen, wie etwa wenn in Tic Tac Toe lediglich andere Koordinatenbezeichnungen für das Spielfeld, zum Beispiel (a,a), (a,b) an Stelle von (1,1), (1,2) usw., oder eine andere Markierung an Stelle des Kreuzes verwendet wird.

Die Regelgraphen erlauben darüber hinaus insbesondere auch die Erkennung von Symmetrien. Dazu werden *Automorphismen* berechnet, also 1:1-Abbildungen von einem Graphen in sich selbst. So lässt sich beispielsweise leicht berechnen, dass die Ersetzung

$$1 \rightarrow 3, 3 \rightarrow 1$$

einen Automorphismus auf dem Tic-Tac-Toe-Regelgraphen bildet, wie sich zum Beispiel anhand des Teilgraphen in Abbildung 17.16 überprüfen lässt, der aus den beiden Regeln

```
5 (init (cell 1 3 b))
9 (init (cell 3 1 b))
```

entstanden ist. Dieser Automorphismus entspricht der 180-Grad Drehung des Spielfelds (vgl. Abbildung 17.14). Die diagonale Spiegelsymmetrie ergibt sich entsprechend aus der Vertauschung der Spielfeldkoordinaten durch

$$\text{cell}[1] \rightarrow \text{cell}[2], \text{cell}[2] \rightarrow \text{cell}[1]$$

Dies ist ebenfalls ein Automorphismus für den Tic-Tac-Toe-Regelgraphen, wie sich wiederum am Beispiel des Teilgraphen aus Abbildung 17.16 ersehen lässt.¹⁰

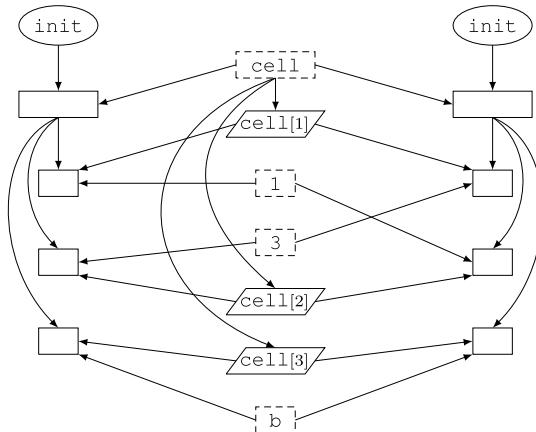


Abbildung 17.16: Regelgraph für Tic-Tac-Toe-Regeln 5 und 9. Werden die Knoten 1 und 3 aufeinander abgebildet, so ergibt sich der gleiche Graph, bei dem lediglich die linke und rechte abgebildete Seite den Platz tauschen.

¹⁰ Um genau zu sein, erfordert dieser Automorphismus noch einige weitere Vertauschungen von Knoten, wie beispielsweise $?m \rightarrow ?n, ?n \rightarrow ?m$ in dem Teilgraphen aus Abbildung 17.15.

Die Domänen- und Regelstrukturanalyse geben einen kleinen Einblick in die vielfältigen Möglichkeiten, ein universelles Spielprogramm dazu zu bringen, Wissen, das implizit in den reinen Regeln eines Spiels steckt, explizit zu machen und zum Spielen zu verwenden. Für dieses noch sehr jungen Gebiet gilt in besonderem Maße, dass es noch viele unerforschte Möglichkeiten zur Anwendung weiterer KI-Verfahren gibt.

17.7 Spiele mit unvollständiger Information

GDL als Eingabesprache für universelle Spielprogramme basiert auf der Annahme, dass nach jeder Runde alle Spieler über die Züge aller Anderen informiert werden. Da die Ausgangssituation vollständig bekannt ist und Züge in GDL immer deterministisch sind, also einen eindeutigen Nachfolgezustand ergeben, kann jeder Spieler nach jeder Runde den aktuellen Spielzustand berechnen. Für klassische Brettspielen wie Schach, Dame oder Go ist dies ebenso adäquat wie für jedes andere Spiel, in dem die Spieler perfekte Information haben. Viele interessante Spiele verwenden jedoch Züge wie zum Beispiel Würfeln, deren Effekt nichtdeterministisch ist, oder sind durch asymmetrische Information gekennzeichnet, wie beispielsweise die schwer zu spielende Schachvariante „Kriegspiel“, bei der die Spieler nur die Figuren der eigenen Farbe sehen. Kartenspiele wie Poker vereinen Zufallszüge (Mischen) mit Informationsasymmetrie (eigene Karten). Um auch solche Spiele für universelle Spielprogramme beschreiben zu können, wurde die Spracherweiterung GDL-II (für engl. „GDL for imperfect information games“) entwickelt.

17.7.1 GDL-II

GDL-II erweitert GDL um zwei Schlüsselwörter mit fester Bedeutung: Die zusätzliche Konstante `random` ist ein vordefinierter Rollenname für einen Spieler, der stets eine rein zufällige Auswahl aus der für ihn definierten Menge legaler Züge trifft. Das zusätzliche Prädikat (`sees r p`) besagt, dass Spieler r eine bestimmte Information p erhält.

Definition 17.7.1 (Semantik einer GDL-II-Beschreibung). *Eine Regelmenge G in GDL-II beschreibt ein Spiel, dessen grundlegenden Komponenten nach Definition 17.2.1 gebildet werden und für das zusätzlich gilt:*

1. *In jeder Stellung S wählt der `random`-Spieler zufällig und mit gleicher Wahrscheinlichkeit einen seiner legalen Züge.*
2. *Die Beobachtungen, die ein Spieler r macht, wenn in einer Stellung S die Züge Z erfolgen, sind genau die Ausdrücke p , für die $(\text{sees } r \ p)$ aus $G \cup S^{\text{true}} \cup Z^{\text{does}}$ ableitbar ist.*

Je nach Spielregel kann eine Beobachtung ein bestimmter Zug eines anderen Spielers, aber auch ein bestimmtes Stellungsmerkmal wie beispielsweise eine zugeteilte Spielkarte sein. Selbst Kommunikation der Spieler untereinander lässt sich mit Hilfe des `sees`-Prädikats beschreiben.

Zur Illustration der beiden zusätzlichen Schlüsselwörter in GDL-II betrachten wir die Beschreibung einer Variante des klassischen Tic Tac Toe, bei der die beiden Spieler die Züge ihres Gegners nicht zu sehen bekommen. Natürlich kann es dabei passieren, dass ein Spieler ein bereits belegtes Feld markieren möchte. Ist dies der Fall, bleibt der Zug ergebnislos, aber der Spieler wird darüber informiert. Weiterhin soll das Spiel dadurch fairer gemacht werden, dass beide

Spieler stets gleichzeitig Felder markieren. Versuchen die Spieler dabei ein und dasselbe Feld zu markieren, so wird durch Münzwurf bestimmt, wer von beiden erfolgreich ist.

Zunächst können die beteiligten Rollen und die Ausgangsstellung in dieser Tic-Tac-Toe-Variante wie folgt definiert werden, wobei die zusätzliche Rolle des Zufallsspielers (Regel 9) für den Münzwurf benötigt wird:

```

1 (role xplayer)
2 (init (cell 1 1 b))
3 (init (cell 1 2 b))
4 (init (cell 1 3 b))
```



```

5 (role oplayer)
6 (init (cell 2 1 b))
7 (init (cell 2 2 b))
8 (init (cell 2 3 b))
```



```

9 (role random)
10 (init (cell 3 1 b))
11 (init (cell 3 2 b))
12 (init (cell 3 3 b))
```

In jedem Zug können beide Spieler ein beliebiges Feld auswählen, das sie nicht zuvor schon selbst markiert haben. Gleichzeitig findet in jeder Runde ein Münzwurf statt, dessen Ergebnis zur Anwendung kommt, wenn beide Spieler dasselbe Feld markieren wollen, und der seinem Zweck entsprechend `tiebreak` genannt wird:

```

13 (<= (legal xplayer
        (mark ?m ?n))
14     (true (cell ?m ?n ?w))
15     (distinct ?w x))
16
17
18 (legal random (tiebreak x)))
```



```

19 (<= (legal oplayer
        (mark ?m ?n))
20     (true (cell ?m ?n ?w))
21     (distinct ?w o))
22
23
24 (legal random (tiebreak o)))
```

Ein Spieler ist mit seinem Versuch, ein Feld zu markieren, erfolgreich, wenn es zuvor leer war und sein Gegenspieler ein anderes Feld gewählt hat:

```

25 (<= (next (cell ?m ?n x))
26     (does xplayer
27         (mark ?m ?n))
28     (true (cell ?m ?n b))
29     (does oplayer
30         (mark ?j ?k)))
31     (or (distinct ?m ?j)
32         (distinct ?n ?k))))
33 (<= (next (cell ?m ?n o))
34     (does oplayer
35         (mark ?m ?n))
36     (true (cell ?m ?n b))
37     (does xplayer
38         (mark ?j ?k)))
39     (or (distinct ?m ?j)
40         (distinct ?n ?k))))
```

Ebenso erfolgreich ist derjenige, für den per Münzwurf entschieden wurde, falls beide Spieler dasselbe Feld markieren wollen. Schließlich muss noch spezifiziert werden, dass alle bereits existierenden Markierungen sowie alle leeren und von keinem Spieler gewählten Felder bleiben, wie sie sind:

```

41 (<= (next (cell ?m ?n ?w))
42   (true (cell ?m ?n b))
43   (does xplayer (mark ?m ?n))
44   (does oplayer (mark ?m ?n))
45   (does random (tiebreak ?w)))
46
47 (<= (next (cell ?m ?n x))
48   (true (cell ?m ?n x)))
49 (<= (next (cell ?m ?n b))
50   (true (cell ?m ?n b))
51   (not (marked ?m ?n)))
52 (<= (marked ?m ?n)
53   (does ?r (mark ?m ?n)))
54
55 (<= (next (cell ?m ?n o))
56   (true (cell ?m ?n o)))

```

Die Regel, nach der ein Spiel beendet ist, sobald ein Spieler eine Reihe, Spalte oder Diagonale vollständig gefüllt hat oder das Spielfeld kein leeres Feld mehr enthält, kann ohne Änderung aus der ursprünglichen Beschreibung von Tic Tac Toe aus Abschnitt 17.1 übernommen werden. Die Regeln für das Ziel des Spiels müssen hingegen um den Fall, dass beide Spieler gleichzeitig eine Reihe gebildet haben, erweitert werden. Der Kürze halber sei hier auf eine Wiedergabe der konkreten Codierung verzichtet.

Da in GDL-II kein Spieler mehr automatisch über die Züge der anderen Spieler informiert wird, würden beide Seiten in unserem Beispiel ohne zusätzliche Information völlig blind spielen, da sie nie wissen können, ob einer ihrer Markierungsversuche erfolgreich gewesen ist. Mit Hilfe des zusätzlichen Schlüsselworts in GDL-II kann jedoch jede gewünschte Form der Informati ons bereitstellung definiert werden. Die folgenden Regeln besagen beispielsweise, dass es ein Spieler stets erfährt, wenn sein Zug erfolgreich war:

```

57 (<= (sees ?r ok)
58   (does ?r (mark ?m ?n))
59   (true (cell ?m ?n b))
60   (does ?s (mark ?j ?k))
61   (or (distinct ?m ?j)
62     (distinct ?n ?k)))
63
64
65 (<= (sees xplayer ok)
66   (does xplayer (mark ?m ?n))
67   (true (cell ?m ?n b))
68   (does random (tiebreak x)))
69 (<= (sees oplayer ok)
70   (does oplayer (mark ?m ?n))
71   (true (cell ?m ?n b))
72   (does random (tiebreak o)))

```

Laut Regel 57–62 erhält ein Spieler immer dann ein „ok“, wenn er ein leeres Feld markiert, das zudem nicht im selben Zug vom Gegner markiert wird. Die gleiche Beobachtung macht ein Spieler laut den Regeln 65–68 bzw. 69–72, wenn er ein leeres Feld markiert und der Münzwurf zu seinen Gunsten ausfällt. Da die gemachten Beobachtungen in beiden Fällen identisch sind, kann sie ein Spieler nicht voneinander unterscheiden und daher nicht darauf schließen, ob beispielsweise sein Gegenspieler das gleiche Feld belegen wollte. Die Beobachtung bleibt selbst dann genau dieselbe, wenn beide Bedingungen gleichzeitig erfüllt sind. In jedem Fall erhalten die Spieler genug Information, um genau zu wissen, welche Felder sie selbst erfolgreich markiert haben. Dies wiederum ermöglicht es ihnen, ihre legalen Züge nach Regel 14–16 bzw. 20–22 zu berechnen. Generell sollte eine Spielbeschreibung in GDL-II stets so gestaltet sein, dass die Spieler ausreichend Information erhalten, um in jeder Situation mindestens einen legalen Zug für sich selbst bestimmen zu können.

In unserem Beispiel kann ein Spieler aus dem Ausbleiben der Beobachtung „ok“ schließen, dass sein Versuch keinen Erfolg hatte. Ein kluger Spieler wird daraus sogar logisch folgern können, dass das entsprechende Feld in der neuen Spielsituation in jedem Fall mit dem gegnerischen

Symbol belegt sein muss – entweder, weil es dies vorher schon war, oder weil der Münzwurf für den Gegner entschieden hat.

Kommunikationsprotokoll für GDL-II Für GDL-II wird ein gegenüber GDL leicht modifiziertes Kommunikationsprotokoll benötigt, bei dem die Spieler nicht mehr nach jedem Zug über die Züge aller Spieler informiert werden (vgl. Abschnitt 17.1.4). Vielmehr werden jedem einzelnen Spieler dessen individuellen Beobachtungen nach den Regeln für das Schlüsselwort `sees` übermittelt. Je nach konkreter Ausgestaltung des Protokolls kann der Spielleiter zusätzlich noch jedem Spieler dessen eigenen Zug bestätigen. Dies hilft dabei, einen eventuellen Kommunikationsfehler zu erkennen, wenn dieser zu einem anderen als den geplanten Zug geführt haben.

17.7.2 Hypothetische Spielstellungen

GDL-II-Spiele stellen ein universelles Spielprogramm vor die zusätzliche Aufgabe, auch ohne vollständige Kenntnis der Spielposition nach guten Zügen zu suchen sowie die richtigen Schlüsse aus den Beobachtungen zu ziehen. Schon das einfache, oben erwähnte Beispiel, bei dem ein Spieler aus dem Ausbleiben eines „ok“ schlussfolgern kann, dass ein bestimmtes Feld spätestens mit dem letzten Zug vom Gegner besetzt wurde, deutet an, was ein gutes Spielprogramm können muss.

Zur Verarbeitung unvollständigen Wissens kann ein universelles Spielprogramm eine Tabelle verwalten, in der zu jedem Zeitpunkt alle aus Sicht des Spielers möglichen Spielstellungen codiert sind – in der mathematischen Spieltheorie auch als *Informationsmenge* bezeichnet. Diese Tabelle wird erzeugt, indem zunächst von der vollständig bekannten Ausgangsposition analog der Konstruktion des Spielbaums (vgl. Abschnitt 17.3) für jede Kombination der Züge aller anderen Spieler die entsprechende Nachfolgestellung berechnet wird. Im weiteren Verlauf wird diese Berechnung dann stets auf die Gesamtheit der möglichen Zustände angewendet. Die nach einem Zug gemachten Beobachtungen wirken dabei als Filter, durch den alle Hypothesen ausgeschlossen werden, in denen der Spieler laut den Spielregeln andere Informationen erhalten hätte. Abbildung 17.17 illustriert dies aus Sicht des Kreuz-Spielers nach seinem ersten Zug in unserer Variante von Tic Tac Toe.

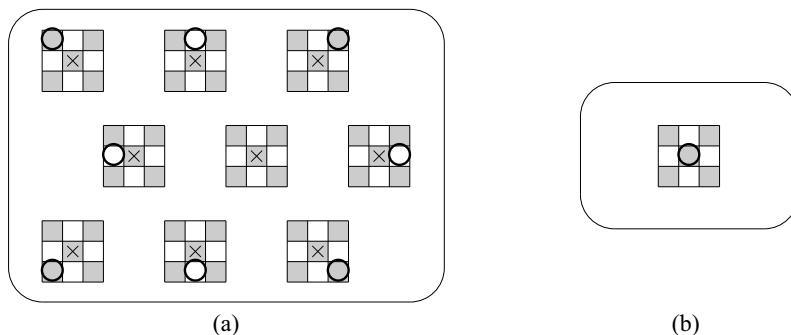


Abbildung 17.17: Die Positionen, die der Kreuz-Spieler nach seinem ersten Zug, mit dem er das Mittelfeld markieren möchte, für möglich hält, wenn (a) er die Beobachtung `ok` macht oder (b) diese Beobachtung ausbleibt.

Die beiden dargestellten Informationsmengen verdeutlichen, wie ein universelles Spielprogramm durch die Berechnung der möglichen Spielstellungen stets die richtigen logischen Schlussfolgerungen aus Beobachtungen ziehen kann: Da Feld (2,2) in jedem Element der Informationsmenge in Abbildung 17.17(a) vom Kreuz-Spieler belegt ist, folgt daraus, dass der Zug erfolgreich gewesen sein muss. Hingegen ist in Abbildung 17.17(b) dieses Feld in dem einzigen möglichen Zustand durch den Gegner markiert, so dass der Kreuz-Spieler auch in diesem Fall die richtigen Schlüsse zieht.¹¹ Ein Spieler kann also im Prinzip stets die Gesamtheit der Informationsmenge für seine Berechnungen zum Beispiel der legalen Züge verwenden. Auch die Bewertung von Zügen kann für jede mögliche Position einzeln erfolgen, um dann zum Beispiel nach dem Minimax-Prinzip den Zug mit maximalem Minimalwert über alle Elemente der Informationsmenge zu wählen.

Im Fall von Tic Tac Toe ohne Ansicht der Züge bleibt die Informationsmenge im Verlauf eines Spiels handhabbar, da einerseits zwar die gegnerischen Zugmöglichkeiten in jeder Runde zu einer weiteren Verzweigung führen (einschließlich der Züge der Sonderrolle `random`), andererseits aber auch mit jedem Zug Information gewonnen wird. In anderen Spielen, wie zum Beispiel Schach ohne Ansicht der Züge oder Spiele, bei denen am Anfang Karten gemischt werden, ist jedoch offensichtlich, dass alleine das Generieren aller denkbaren Spielsituationen praktisch unmöglich ist. Eine Lösung bietet auch hier wieder die Verwendung stochastischer Methoden an (vgl. Abschnitt 17.4). Dabei werden nur für einige, zufällig ausgewählte gegnerische Züge Nachfolgepositionen erzeugt, d.h., es wird nur mit wenigen Elementen der vollständigen Informationsmenge gerechnet. Im Verlauf des Spiels wird jede dieser Hypothesen auf Konsistenz mit den Beobachtungen getestet und, falls inkonsistent, verworfen bzw. durch ein anderes, wiederum zufällig ausgewähltes Element der Informationsmenge ersetzt. Mit dieser Methode kann jedes universelle Spielprogramm für GDL leicht auf GDL-II-Spiele erweitert werden, da jede Hypothese eine vollständige Spielsituation abbildet, mit der so gespielt werden kann, als ob es sich um ein Spiel mit vollständiger Information handelt. Die aus der Analyse der einzelnen, hypothetischen Stellungen resultierenden Zugvorschläge und ihre berechneten Werte können dann zum Beispiel zu einem wahrscheinlichkeitstheoretischen Erwartungswert zusammengefasst werden, aufgrund dessen die Zugauswahl erfolgt. Dieses Vorgehen ist leicht parallelisierbar, da die Hypothesen voneinander unabhängig sind.

Die reine Simulation zufällig ausgewählter, vollständiger Spielstellungen hat jedoch ihre Grenzen. Zum einen bildet sie oft nur einen kleinen Ausschnitt aus dem Hypothesenraum, was leicht zu falschen Schlussfolgerungen führen kann. Alleine die Legalität eines Zugs, aber auch seine Güte können nicht immer schon dadurch gewährleistet werden, dass dieser in einigen ausgewählten Stellungen legal bzw. vielversprechend ist (vgl. dazu auch Abschnitt 17.4.4 über die Grenzen der stochastischen Spielbaumsuche). Ein weiterer und sehr fundamentaler Nachteil steckt in der impliziten Annahme vollständiger Information in jeder einzelnen Hypothese. Auf diese Weise wird der Unterschied zwischen Wissen und Nichtwissen völlig ignoriert, so dass zum Beispiel ein Zug, dessen einziger Zweck die Gewinnung wertvoller Information ist, in keiner der hypothetischen Stellungen als sinnvoll erachtet wird. Aus dem gleichen Grund würde auch in jedem einzelnen hypothetischen Spiel nie ein Nachteil in einem Zug gesehen, mit dem freiwillig wertvolle Information preisgegeben wird. Insgesamt bildet daher auch der große Be-

¹¹ Sein Gegenspieler weiß in dieser Situation natürlich ebenfalls, dass er das Feld belegt hat; im Unterschied zum Kreuz-Spieler kann er allerdings nicht wissen, dass jener im ersten Zug erfolglos war.

reich der Spiele mit unvollständiger Information ein interessantes und ergiebiges Betätigungsfeld auf dem Gebiet der universellen Spielprogramme.

17.8 Weiterführende Literatur

Eine gute Quelle für den Einstieg in die aktuelle Forschung auf dem Gebiet der universellen Spielprogramme bietet das Webportal www.general-game-playing.de. Hier finden sich neben einer ausführlichen Literatursammlung auch Links zu einem aktiven Gameserver mit Hunderten von GDL-Spielbeschreibungen, zu den Quellcodes diverser universeller Spielprogramme, zu weiterer Software wie beispielsweise für einen Spielleiter sowie zu Lehrmaterialien zum Thema.

Abschließend seien einige Hinweise auf weiterführende Literatur zu den einzelnen Themen der vorangegangenen Abschnitte gegeben. Aus historischer Sicht und in Bezug auf spezielle Spiele sind die Arbeiten von Alan Turing [21] und Feng-Hsiung Hsu [7] zum Thema Schachprogrammierung sowie von Arthur Samuel [15] und Jonathan Schaeffer *et al.* [16] zum Damespiel von Bedeutung.

Für die effiziente Verarbeitung von GDL-Regeln in einem universellen Spielprogramm haben Stefan Edelkamp und Peter Kissmann [4] eine auf binären Entscheidungsdiagrammen basierende Methode entwickelt, während Evan Cox *et al.* [3] ein Verfahren zur Erzeugung von Graphen für die Inferenz mit GDL-Regeln entworfen haben. Die stochastische Baumsuche wurde insbesondere von Yngvi Björnsson und Hilmar Finsson [1] untersucht und um viele Aspekte erweitert [5]. Einige grundlegende Methoden zur heuristischen Suche und zum vollautomatischen Wissenserwerb wurden von Gregory Kuhlmann und Peter Stone [10], Jim Clune [2], David Kaiser [8] sowie Stephan Schiffel und Michael Thielscher [18] entwickelt. Die Anwendung eines stochastischen Verfahrens in universellen Spielprogrammen für Spiele mit unvollständiger Information geht auf Michael Schofield *et al.* [19] zurück.

Den Stand der Forschung nach fünf Jahren internationaler Wettbewerbe für universelle Spielprogramme hat Michael Thielscher [20] zusammengefasst. Von historischem Interesse sind die Arbeiten Jacques Pitrats, der als Erster das Konzept eines universellen Spielprogramms erdacht und implementiert hat [14]. Schließlich seien noch die Dissertationen von Barney Pell [12], Jim Clune [13], Stephan Schiffel [17], Hilmar Finsson [6] und Peter Kissmann [9] empfohlen, die jeweils bestimmte Aspekte der universellen Spielprogrammierung in aller Ausführlichkeit behandelt haben.

Literaturverzeichnis

- [1] Björnsson, Y. und Finsson, H. (2009). CADIAPLAYER: A simulation-based general game player. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(1):4–15.
- [2] Clune, J. (2007). Heuristic evaluation functions for general game playing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1134–1139, Vancouver. AAAI Press.

- [3] Cox, E., Schkufza, E., Madsen, R., und Genesereth, M. (2009). Factoring general games using propositional automata. In *Proceedings of the IJCAI Workshop on General Game Playing (GIGA'09)*, pages 13–20, Pasadena. Erhältlich auf: www.general-game-playing.de.
- [4] Edelkamp, S. und Kissmann, P. (2008). Symbolic classification of general two-player games. In Dengel, A., Berns, K., Breuel, T., Bomarius, F., und Roth-Berghofer, T., editors, *Proceedings of the German Annual Conference on Artificial Intelligence (KI)*, volume 5243 of *LNCS*, pages 185–192, Kaiserslautern. Springer.
- [5] Finnsson, H. und Björnsson, Y. (2010). Learning simulation control in general game-playing agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 954–959, Atlanta. AAAI Press.
- [6] Finsson, H. (2012). *Simulation-Based General Game Playing*. PhD thesis, Reykjavík University.
- [7] Hsu, F.-H. (2002). *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion*. Princeton University Press.
- [8] Kaiser, D. (2007). Automatic feature extraction for autonomous general game playing agents. In Durfee, E., Yokoo, M., Huhns, M., und Shehory, O., editors, *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, page 92, Honolulu. IFAAMAS Press.
- [9] Kissmann, P. (2012). *Symbolic Search in Planning and General Game Playing*. PhD thesis, Universität Bremen.
- [10] Kuhlmann, G., Dresner, K., und Stone, P. (2006). Automatic heuristic construction in a complete general game player. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1457–1462, Boston. AAAI Press.
- [11] Love, N., Hinrichs, T., Haley, D., Schkufza, E., und Genesereth, M. (2006). General Game Playing: Game Description Language Specification. Technical Report LG–2006–01, Stanford Logic Group, Computer Science Department, Stanford University. Erhältlich auf: www.general-game-playing.de.
- [12] Pell, B. (1993). *Strategy Generation and Evaluation for Meta-Game Playing*. PhD thesis, Trinity College, University of Cambridge.
- [13] Pell, B. (2009). *Heuristic Evaluation Functions for General Game Playing*. PhD thesis, University of California, Los Angeles.
- [14] Pitrat, J. (1971). A general game playing program. In Findler, N. und Meltzer, B., editors, *Artificial Intelligence and Heuristic Programming*, pages 125–155, Edinburgh University Press.
- [15] Samuel, A. (1959). Some studies in machine learning using the game of checkers. *IBM Journal*, 3:211–229.
- [16] Schaeffer, J., Björnsson, Y., Burch, N., Kishimoto, A., Müller, M., Lake, R., Lu, P., und Sutphen, S. (2005). Solving checkers. In Kaelbling, L. und Saffiotti, A., editors, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 292–297, Edinburgh.
- [17] Schiffel, S. (2011). *Knowledge-Based General Game Playing*. PhD thesis, Technische Universität Dresden.
- [18] Schiffel, S. und Thielscher, M. (2007). Fluxplayer: A successful general game player. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1191–1196, Vancouver. AAAI Press.

- [19] Schofield, M., Cerezhe, T., und Thielscher, M. (2012). HyperPlay: A solution to general game playing with imperfect information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1606–1612, Toronto. AAAI Press.
- [20] Thielscher, M. (2011). General game playing in AI research and education. In Bach, J. und Edelkamp, S., editors, *Proceedings of the German Annual Conference on Artificial Intelligence (KI)*, volume 7006 of *LNAI*, pages 26–37, Berlin. Springer-Verlag.
- [21] Turing, A. (1953). Chess. In Bowden, B., editor, *Faster than Thought*, pages 286–295. Pitman Publishing, London.

Index

- A**
 - A*-Suche 95
 - auf Graphen 97
 - speicherbeschränkte 102
 - A-Algorithmus 95
 - A-posteriori-Chancen 257
 - A-posteriori-Vertrauen 257
 - A-posteriori-Wahrscheinlichkeit 245, 251
 - A-priori-Chancen 257
 - A-priori-Vertrauen 257
 - A-priori-Wahrscheinlichkeit 251
 - A-priori-Wissen 259
 - Abduktion 132
 - Abgeschlossenheit 178
 - Abhängigkeitsgraph
 - bedingter 272
 - Abhängigkeitskarte 272
 - Abhängigkeitsstruktur 487
 - Ableitung 133, 146, 147, 161, 181, 620, 623
 - unscharfe 393
 - Ableitungsschritt 622
 - Abnormalitätsprädikat 186
 - Abschluss
 - deduktiver 148
 - absolute Häufigkeit 241
 - Abstandsfunktion 425
 - Abstraktion 338
 - ACT-R 32
 - Adaption
 - von Fällen 321
 - Additionsaxiom 242
 - erweitertes 242
 - Agenda 78
 - Agent 9–10, 12, 13, 107, 109, 171, 329, 461, 527–551, 614
 - intelligenter 6, 529
 - kognitiver 33, 559
 - kommunizierender 559
 - konversationaler 567
 - multipler 527
 - nützlichkeitsbasierter 530
 - rationaler 183, 199
 - virtueller 567
 - zielbasiert 530
- Agenten
 - künstliche 13
- Agentenarchitektur 529
- Agentenkommunikationssprache 535
- Agentenorientiertes Software Engineering 538
- Agentspeak(L) 531
- AgentUML 537
- AGR 539
- Ähnlichkeit 287, 307
- Ähnlichkeitsfunktion 425
- Ähnlichkeitsmaß 303, 309
 - Semantik 310
- Ähnlichkeitsrelation 287
- Aktion 330–333
- Aktionspotential 360
- aktiver Pfad 270
- Aktivierungsfunktion 365, 366
- Akzeptanzgrad 284
- Algorithmisierung 119
- Allgemeingültigkeit 132
- Analyseverfahren
 - probabilistisches 507
- Ansatz von Mamdani 283
- answer set programming 126
- Antizipation 42
- Antwortmengen 197
- Antwortmengen-Programmierung 126, 197
- Aplysia 363
- Arbeitsgedächtnis 40, 62
- Architektur
 - kognitive 566, 568
- Architekturen
 - kognitive 31

- Argumentation
 formale 198
- Assertionen 182
- Assistent
 sprachgesteuerter 477
- Assoziativität 248
- Assoziator 369
- Attraktor 370, 372
- Attribut-Wert Darstellung 305
- Äußerung
 komplexe 577
 sprachliche 474
- Äußerungsplanung 575
- Aufmerksamkeit 35, 37
- Aufwands-Prognose 322
- Ausdrucksfähigkeit
 eines Formalismus 123
- Ausnahmen 171
- Aussagenlogik
 Beweisverfahren 142
- Autoepistemische Logik 183
- autoepistemisches Schließen 172
- Automat
 endlicher 491
- automatisches Schlussfolgern 235
- Autonomie 529
- B**
- Back-Propagation 378
- Backtracking 212, 335, 348
 dependency-directed 87
- backtracking 87
- Bagging 419
- Bayes 33, 507
- Bayes-Netz 235, 266, 277
 kausales 33
- Bayesscher Satz 245
- BDI 547, 567, 571
- BDI Agent 530
- Bedeutung
 kompositionale 488
 sprachlicher Ausdrücke 478
- bedingt unabhängig 259, 267
- bedingte Unabhängigkeit 266
- bedingte Verteilung 239
- bedingte Wahrscheinlichkeit 243
- bedingter Abhängigkeitsgraph 272
- bedingter Unabhängigkeitsgraph 267, 272
- Begriffsausdruck
 inkonsistenter 121
- Begriffstheorie 39
- Behaviorale Messungen 26
- Behaviors 570
- Benutzerschnittstelle 603
- Berechenbarkeitseigenschaft 117
 eines Formalismus 123
- Bergsteigen
 mit Zurücksetzen 91
 optimistisches 91, 93
- Beschreibungslogik 9, 115, 125, 132, 586, 589
- Bestensuche 94
- bevorzugte Modelle 177
- bevorzugte Teiltheorie 187
- Beweisprozedur 134, 148
- Bewertungsfunktion 51, 463, 465, 466, 546, 632, 636, 637
- Bewusstsein 35
- Bilderkennen 174
- Bildverarbeitung 374
- Binärprotokoll 543
- Blackboard Systeme 536
- blockierter Pfad 270
- Blum 345
- BOB-Test 319
- Boosting 419
- Bordprotokoll 543
- branch and bound 84
- Breitensuche 82
- BWB-Test 319
- C**
- CAPS 32
- Case-Based Reasoning (CBR) 297
- Case-Retrieval-Netz 320
- CASEC-Architektur 568
- CBR
 Charakteristika 299
 Methodologie 301
- CBR-Cycle 301
- CBR-System 324
- CDL 182
- center of gravity 286
- certainty factor 252
- Chancen 257
- Chart 500

- Chart-Parsing 499–503
Clique 280
Cliquesbaumpropagation 279
closed list 80, 97
Closed World Assumption 175, 332
CLP 221
COG 286
cognition
 embodied 22
 situated 22
Comet 222
conditional dependence graph 272
conditional independence graph 272
Constraint Satisfaction 572
Constraint-Löser 206
Constraint-Lösungsmethode 121
Constraint-Satisfaction 207
Constraint-Satisfaction-Problem 503
Constraint-System 122
 Erfüllbarkeit eines 122
Constraints 205–230
 globale 215
 temporale 227
Contract Net Protocol 537
Cortex 359, 391, 392
Cortexareal 392
Cowan, J.D. 368
CSP 207
- D**
- d-Trennung 270
Dartmouth-Konferenz 2
Data Mining 409
Daten
 große Mengen 587
 verknüpfte 590, 593
 Visualisierung 605
Datenbanktheorie 260
Deduktion 22
Deduktionstheorem 137
Deep Computing 400
Default 171, 177
 benannter 190
 normaler 182
 offener 177
Default-Logik 177
 Spezialfälle 182
Default-Logik-Modifikationen 182
- Defuzzifizierung 285
Defuzzifizierungsschnittstelle 282
Denken 44
 abduktives 45
 analoges 45
 deduktives 44
 induktives 45
 menschliches 49
 problemlösendes 50
dependence graph 272
dependence map 272
Dependenzstruktur 487, 501
Diagnose 174
Dialog 560
Dialogmanagement 563
Dialogsystem 478, 563
Dichteschätzung 380
Diktiersystem 483
Disjunktion 141, 197, 624
Disjunktion von Hypothesen 253
Diskontierungsfaktor 631
Diskriminanzfunktion
 verallgemeinerte 373
Diskurs 474
Diskurskontext 572
Diskursplanung 565
Diskursrepräsentationsstruktur 489
Diskursrepräsentationstheorie 489
Diskursverarbeitung 58
Distributivgesetze 250
Domänenanalyse 637
Domänengraph 637
Dominante Strategie 541
Doyle, J. 177
DPLL 142
Dreiecksfunktion 283
Dublin Core 586
dynamische Programmierung 463
- E**
- E-commerce 323
einfach verkettet 279
Elementarereignis 240
Elementarwahrscheinlichkeit 241
Elternknoten 274
Emergenz 536
Emotion 55, 566, 577
EMYCIN 255

- Endzustand 75
 Engagement 562
 Ensemble-Methoden 419
 Entitätenauflösung 601
 Entscheidbarkeit 124
 Entscheidungsalgorithmus 119
 Entscheidungslogik 282
 EPIC 32
 Ereignis 236
 - Chance des Eintretens 240
 - Elementar- 240
 - paarweise unvereinbare 242
 - sicheres 241
 - unmögliches 241
 - unvereinbare 242
 - zufälliges 240
 Ereignisalgebra 241
 Ereignisdiskjunktion, vollständige 245
 Ereignisraum 240
 Erfüllbarkeit 132, 137
 Erfüllungsgrad 284
 Erinnern 42
 Erklärung 189
 Erregungsdynamik 360, 365
 erweitertes Additionsaxiom 242
 Erweiterung
 - zylindrische 263
 Evidenzfaktor 277
 Evidenzkombination
 - Nichtkommutativität der 256
 Evidenzpropagation 260, 263, 266, 275
 Existenz von Extensionen 182
 Expansion 183
 Expansion von Begriffsausdrücken 120
 Explorationsbaum 97
 Explorationsgraph 97, 101
 extensionale Fuzzy-Menge 288
 extensionale Hülle 288
 Extensionalität 288
 Extensionen 176
- F**
- faktorisierbar
 - bzgl. eines gerichteten Graphen 274
 - bzgl. eines ungerichteten Graphen 273
 Fall 299
 Fallbasis 300
 - Probleme 321
- Fallunterscheidung 180
 Feedback 562
 Fehler
 - Trainingsfehler 412
 - wahrer Fehler 412
 Feld
 - topologisches 490
 Fertigkeiten
 - sensomotorische 43
 FIPA 537
 FIPA-ACL 536
 Fixierung
 - funktionale 51
 Fixpunkt 178, 372
 Fixpunktoperator 177
 foothill problem 93
 Formalismus
 - ALC* 115
 Formelschema 185
 Fragebeantwortung
 - domänenunabhängige 484
 Frame 174
 Frame-Axiome 173
 Frame-Problem 173, 330
 Frame-Sprachen 115
 Frame-System 174
 Fukushima, K. 399
 Funktion
 - plenoptisch 393
 Furst 345
 Fuzzifizierungsschnittstelle 282
 Fuzzy-Logik 235, 246, 634
 Fuzzy-Menge 235, 246, 247, 281
 - extensionale 288
 Fuzzy-Mengentheorie 235, 246
 Fuzzy-Regel 281
 Fuzzy-Regelsystem 235, 281, 282
 Fuzzy-Regelung 282
 - Ansatz von Mamdani 283
 - mit Gleichheitsrelationen 287
 - mit Relationalgleichungen 290

G

Gödel-Implikation 291
 Gödel-Relation 291
 Gabbay, D. 180
 GDL 614, 615, 618, 640, 642

 - Kommunikationsprotokoll 619

- Gedächtnis 39
 Taxonomie 40
- Gegenspieler 627
- Gegenspielermodell 627
- Gehirn 357
- General Game Playing (GGP) 613
- General Problem Solver 352
- Generierung
 multimodales Verhalten 565
 natürlicher Sprache 514
 von Handlungsplänen 350
- Gesetz 174
- Gesprächsdolmetschen 478
- Gestenerkennung 564
- Gewichtsdynamik 365
- Glättung 235, 248
- Gleiche-Kosten-Suche 84
 verbesserte 85
- Gleichheitsrelation 287
- gleichmöglich 240
- Grad der Möglichkeit 246
- Grad der Wahrheit 246, 247
- Grad der Zugehörigkeit 246, 247
- Grammatik
 constraint-basierte 496
 kontextfreie 492
- Graph
 azyklischer 270
 gerichteter 270
 ungerichteter 270
- Graphentheorie 260
- graphisches Modell 239, 259
- Graphoid-Axiome 268
- Graphplan 345
- greedy search 94
- Großmutterzelle 400
- Grundinstanzierung 622
- Gruppe 539
- H**
- Häufigkeit
 absolute 241
 relative 241
- Hülle
 extensionale 288
- Habituation 363
- HACKER 344
- Hammingmaß 314
- handhabbare Probleme 124
- Handlungskontrolle 35–37
- Handlungsplanung 6, 34, *siehe* Planen, 572
- Haufenparadoxon 246
- Hauptachsentransformation 381, 386
- Hayes, P. 330
- Head-driven Phrase Structure Grammar 497
- Hebb
 -Regel 363
 -Synapse 364
 verallg. Regel 386
- Hebb, D.O. 363
- Herbrand-Theorie 155
- Heuristik 48, 90, 213, 336–338
- Hierarchical Task Network 338
- Hindernisvermeidung 396
- Hintergrundwissen 8, 49, 314, 439, 478, 597
- Hippocampus 364
- Hodgkin, A.L. 360
- HPSG 497
- Hubel, D.H. 399
- Huxley, A.F. 360
- Hyperebene
 optimal 428
 Separationsweite 429, 437
 VC-Dimension 436
- Hyperparameter 389
- Hypothese 133, 146, 188, 413, 434, 439, 646
- Hypothesen
 Disjunktion von 253
 Konjunktion von 253
- Hypothesenraum 434, 435, 450, 646
- I**
- Identität 600
- Imitation 43
- Implikation
 Gödel- 291
- Impräzision 236
- independence graph 272
- independence map 272
- Independent Component Analysis 382
- Indikatorfunktion 247
- induktive logische Programmierung 440
- Infektionskrankheit 250

- Inferenz 181, 599, 619
 automatische 129–164
- Inferenzalgorithmen 117
- Inferenzregel 133
- Inferenzrelationen
 Eigenschaften von 180
- Information
 inkonsistente 171
 unvollständige 642
- Informationsextraktion 484
- Informationsklassifikation 483
- Informationsmanagement 483
- Informationsmenge 645
- Informationspräsentation
 nutzeradaptive 478
- Informationsrekonstruktion 479
- Informationsstruktur 489
- Informationsstrukturierung 484
- Informationswechsel 358
- Informiertheit 90
- Inhaltserschließung 487
- Initiative 562
- Inkonsistenzbehandlung 187
- inkonsistenztolerantes Schließen 173
- Inkrementalität
 temporale 518
- Intelligenz 3
- Intentionalität 21
- Interaktion 529, 534
- Interaktionsprotokoll 537
- Interaktionssteuerung 571
- Interaktionsverhalten 571
- Interpolation 235, 248
- Interpretation 135
- InterRAP 532, 533
- introspektive Fähigkeiten 183
- iterative broadening 87
- iterative deepening 88, 102
- K**
- Körperlichkeit 560
- Kalkül 133
- Karte
 Abhängigkeits- 272
 Disparitäts- 394
 Merkmals- 375, 385
 neuronale 394
 perfekte 272
- retinotope 392
- topographische 395
- Unabhängigkeits- 272
- kd-Baum 317, 318
- Kernontologie 597, 598
- KI
 situierte 12
 symbolische 11
 Teilbereiche 13
 verkörperte 12
- KI-These 4
- KL-ONE 115
- Klassifikator 369, 372, 397
- klassische Logik 238
- klassischer Wahrscheinlichkeitsbegriff 240
- Klausel 140
- Klauselform 140
- Knoten 78
 erschlossener 80, 97
 offener 78, 97
- Knotenexpansion 78
- Knowledge Graph 604
- Kognition 21–64
 über andere 56
 in der Gruppe 57
 menschliche 34
 soziale 53
- Kognitionswissenschaft 22
- Kohärenz 490
- Kohäsion 490
- Kohonen, T. 385
- Kolmogorow-Axiome 242
- Kombination
 Nichtkommutativität der Evidenz- 256
 parallele 254
 in EMYCIN 255
 sequentielle 253
- Kombinationsfunktion 257
- Kombinatorik 240
- Kommunikation 534, 559
 multimodale 519, 562
- Kommunikationsfähigkeit
 sprachlichen 476
- Kommutativität 248
- Kompetenz 7
- Komplement 248

- Komplementmenge 248
Komplexität 52
Komplexitätsklasse PSPACE 125
Komplexitätstheorie 124
Komponenten
 langsam variierende 383
 maximaler Varianz 381
 unabhängige 382
Konditionierung 42, 363
Konjunktion von Hypothesen 253
Konnektionismus 22
Konolige, K. 184
Konsequenz 177
Konsistenz 132
 globale 212
 lokale 209
Konsistenzannahme 177
Konstituentenstruktur 487
Konstruktorannahme 306
Kontraposition 179
Konzept 473
Koordination 539
 interpersonale 563
Koreferenz 601
 sprachliche 489
Korrespondenzproblem (des Stereosehens) 394
Kostenfunktion
 monotone 84
KQML 535
Kreuzvalidierung 387, 426, 431
Kumulativität 181
- L**
- Lösung
 Güte der 81
 optimale 95
Lakemeyer, G. 184
Langsamkeit
 Prinzip der 383
Langzeitgedächtnis 42
Laut 474
Lernaufgabe 407
 Begriffslernen aus Beispielen 413, 440
 Cluster-Analyse 458
 deskriptive 449
 Entscheidungsbaumlernen 414
Finden optimaler Hyperebenen 429
Finden von Assoziationsregeln 442, 444, 456
Finden von Subgruppen 450, 456
Funktionslernen aus Beispielen 411
Lernen aus Beispielen 408
lineare Klassifikation 428
prädiktive 449
Regression 413, 421
Verstärkungslernen 461
Lernbarkeit
 PAC 434
 von k-KNF 435
 von k-Term-DNF 435
Lernen 39, 41, 363, 378
 assoziatives 42
 von Schlussfolgerungsnetzen 280
 wahrscheinlich annähernd korrektes 434
Lernregel 365, 375
Lernverfahren 513
 Apriori-Algorithmus 444
 hierarchisches Clustering 460
 Instanzbasiertes Lernen 422
 k-means 459
 k-NN diskret 424
 k-NN kontinuierlich 424
 Q-Lernen 465
 Stützvektormethode 427
 top-down induction of decision trees 415
Levesque, H. 184
Lexical Functional Grammar 494
Lifschitz, V. 177, 184
likelihood ratio 258
Linguistik
 theoretische 24
 linguistische Regel 281
 linguistische Variable 235
 linguistischer Term 283
Linked Data 590, 607
Linked Open Data 601
Literal 140
Logik 7, 22, 24
 Beschreibungs- 115
 formale 113, 488
 klassische 238
 mathematische 131

- Modal- 237
 terminologisch 115
- Logikprogrammierung 193, 498
- logisches Neuron 366
- Long-Term Potentiation 364
- M**
- möglich 237
- mögliche Welten 185
- Möglichkeit 290
- Möglichkeitsgrad 246
- Maß 242
 - des Misstrauens 251
 - des Vertrauens 251
- Maßtheorie 242
- Makinson, D. 180, 181
- Makrooperator 93
- Markov-Modell 508
- Markow-Netz 235, 266
- Marr, D. 357, 394
- Maschinelles Lernen 298, 353, 405–467, 513, 564
- Max 560, 567
- Max-Kriterium-Methode 285
- maximal konsistente Teilmengen 187
- McCarthy, J. 177, 330, 352
- McCullough, W.S. 366
- McDermott, D. 177
- MCT-Suche 629
- mean of maxima 285
- Mean-of-Maxima-Methode 285
- measure of belief 251
- measure of disbelief 251
- Mechanism Design 540, 542
- mehrdimensionaler Raum 260
- Membranpotential 359
- Menge
 - Fuzzy- 246, 247
 - unscharfe 235
- Mengenfunktion 242
- Mensch-Maschine-Kommunikation 477, 485
- Merkmalsdetektor 374, 386, 399
- Merkmalsstruktur 493
- Messgröße 282
- minimale Modelle 185
- Minimax-Suche 626
- Minimax-Verfahren 624
- Minimum
 - lokales 93
- Minsky, M. 377
- Misstrauen
 - Maß des 251
- Mittel-Ziel-Analyse 51, 336
- MKNF 184
- Mobilitätsheuristik 633
- Modalität 237
- Modallogik 118, 237
- Modaloperator 177
- Modell 137
 - graphisches 239, 259
- Modellierung
 - kognitive 1, 23, 24
- Modellierungssprachen
 - Constraint-basierte 222
- Modellselektion 387
 - Bayes'sche 389
- Modelltheorie
 - mentale 47
- Modus Ponens 133
- MOISE+ 539
- MOM 285
- Monotoniebeschränkung 101
- Monotonieeigenschaft 172
- Moore, B. 177
- Morphologie 486
- Multiagentenplanen 540
- Multiagentensysteme 527–551
- multimodale Kommunikation 519
- Multimodalität 561
- Multiplikationssatz 244
- Mustererkennung 396
- mutex 346
- MYCIN 250
 - Regelauswertung in 254
- N**
- Nachbedingung 332
- Nachricht 534
- Nash-Gleichgewicht 541
- Negation 184, 197, 249, 346, 623
- Negation as Failure 175, 184, 193
- Negentropie 382
- Nervensystem 359
- Nervenzelle 359
- NETtalk 400

- Netz
 neuronales 22
 rekurrentes 63
Netzwerkdarstellung 263
Neuroinformatik 357
Neurokognition 27
Neuronale Netze 357–401
Neuronales Netz 11
 künstliches 32, 365
 natürliches 359
 Typen 369
Neurone
 abgestimmte 366
Neurotransmitter 361
Neurowissenschaft
 kognitive 27
Newell, A. 352
nicht-handhabbare Probleme 124
Nichtkommutativität 256
nichtmonotones Schließen 171
NOAH 339
NONLIN 340
Normalform
 konjunktive 139
 logische 137
Normalform eines Begriffsausdrucks 121
Normalisierung 601
Normative Systeme 539
notwendig 237
- O**
Okulardominanzstreifen 391, 395
Ontologie 585, 595
 Arten von 596
 vernetzte 598
 verteilte 597
open list 78, 97
Operator 75, 332, 333
 invertierbarer 79
OPL 222
Optimierung 368
 quadratische 430
Optischer Fluss 396
OR 180
Organisation 534, 538
Orientierungskolumnen 391
OWL 115, 586, 589, 600
- P**
paarweise unvereinbar 242
PAC-Lernen 434
Papert, S. 377
parallele Kombination 254
 in EMYCIN 255
Pareto-Effizienz 541
Parsing
 als Constraint Satisfaction 501
Partial Global Planing 540
Penumbra 237
perfekte Karte 272
Persistenz-Default 173
Persistenzproblem 330
Perzeptron 372, 376, 399
Pfad
 aktiver 270
 blockierter 270
Phon 474
Phonmodell 508
Phrasenstrukturgrammatik 491
Physis 577
Pitts, W. 366
Plan
 bedingter 351
 partiell geordneter 339–340
Planen 329–353
 graphbasiertes 345–349
 hierarchisches 338
 im Planraum 339–345
 im Zustandsraum 334–339
 least-commitment 339
 mit partiell geordneten Plänen 340–344
 reaktives 351
 Transformationsplanen 344–345
 unter Unsicherheit 350
Planer 570
Planungsaufgabe 333
Planungsgraph 345
Plastizität 363
Plateau 93
Pluralitätsprotokoll 542
Poggio, T. 394
polynomiales Laufzeitverhalten 124
Poole, D. 181, 187, 189
POP 340–343
Possibilitätsgrad 236

- Possibilitätstheorie 247, 290
 Possibilitätsverteilung 290
 Prädikaten-Zirkumskription 185
 Prädikatenlogik 113, 131, 171
 - erster Stufe 134
 - Kalküle 154
 Präferenz 237
 Präferenzrelation 185, 186
 Prämissen, partiell geordnete 192
 Pragmatik 474
 Prioritäten 179, 183
 Proaktivität 529
 probabilistische Analyseverfahren 507
 probabilistisches Schlussfolgerungsnetz 235, 239, 259, 274
Problem
 - komplexes 52
 Problemlösen 44
 - als Suche 75
 Procedural Reasoning System (PRS) 531
 Produktsatz 244
 Programmierung
 - constraint-basierte 220
 - Constraint-logische 221
 Programmverifikation 129
 Progression 334
 Projektion 262
 PROLOG 175
 Prosodie 517
 Pseudo-Inverse Matrix 370, 379
 PSPACE 125
 Psychologie 24
 Puffer
 - visueller 40
Q
 Q-Learning 465
 Qualifikationsproblem 174, 330
 Qualitätsmaß 417
 - Informationsgewinn 417
 Quantor
 - sprachlicher 506
 - quasi-induktiv 178
R
 Radialbasis
 - funktionen 366
 - netzwerke 378
 Randomisierte Verfeinerungen 93
 Randverteilung 239, 265
 Rationalitätsprinzip 7
 RDF 588
 RDF-Graph 588
 RDF-Tripel 591, 593, 599
 RDFS 589
 Reaktivität 529
 Rechtschreibprüfung 478
 Referenz
 - sprachinterne 479
 Referenzauflösung 480
 Reflexagent 530
 - modellbasierter 530
 Regel
 - Fuzzy- 281
 - linguistische 281
 - vage 239, 248
 Regelauswertung
 - in einem Mamdani-Regler 284
 - in MYCIN 254
 Regelgraph 640
 Regelstrukturanalyse 639
 Regelsystem
 - Fuzzy- 235, 281
 Regression 335
 Regularisierungsnetzwerke 388
 Reinforcement Learning 461
 Reiter 177, 183
 Reiz
 - sozial-emotionaler 54
 Reizverarbeitung
 - neuronale 35
 Relation
 - Gödel- 291
 - rhetorische 490
 relationales Schlussfolgerungsnetz 260
 Relationalgleichung 290
 relative Häufigkeit 241
 Repräsentation
 - mentale 38
 - symbolische 6
 - von Wissen 109
 Repräsentationssprache 305
 Resolution 120, 134, 269
 - A-geordnete 149
 - aussagenlogische 145
 - prädikatenlogische 157

- Resolutionskalkül 145, 146
Korrektheit 147
Verfeinerungen 154
- Retrieval 317
- Rezeptives Feld 374, 392
Modell 392
- RIF 589
- Rolle 115, 538
- Rücksetzen
chronologisches 87
- Rückwärtssuche 79
- S**
- Sacerdoti 339
- Sackgasse 76, 87
- Sandhaufen 246
- Satz 474
- Satzerkennung 58
- Satzplanung 575
- Schätzfunktion
faire 95
heuristische 89
optimistische 96
zulässige 96
- Schachprogramm 613
- scharfe Grenze 237
- Schattenwurf 262
- Schema 138, 185, 190, 589, 603, 607
der Constraint-Programmierung 222
motorisches 43
- Schema-Matching 601
- Schichtenarchitektur 532
- Schleife
deliberative 570
- Schließen 48
automatisches 129
fallbasiertes 297
logisches 131
- Schluss
abduktiver 114
deduktiver 114
induktiver 114
- Schlussfolgern 44, 238
automatisches 235
- Schlussfolgerungsnetz
Lernen eines 280
probabilistisches 235, 239, 259, 274
relationales 260
- Schlussregel 238
- Schnitt 248
- Schnittaxiom 268
- Schnittmenge 248
- schwache Beweisbarkeit 188
- Schwache Vereinigung 268
- Schwerpunktmethode 286
- Sehbahn 390
- Sejnowski, T.J. 400
- Semantic Web 9, 585–611
- Semantic Web Layer Cake 587
- Semantik 474
der Prädikatenlogik 134
formale 117
GDL-Beschreibung 620
- Semantikkonstruktion 495
- SemaPlorer 603, 605
- Semi-Graphoid-Axiome 268
- semi-monoton 183
- semi-normal 182
- Separatormenge 277
- sequentielle Kombination 253
- sicheres Ereignis 241
- Sicherheit
Quantifizierung von 237
- Sicherheitsfaktor 235, 239, 250, 252
-axiome 257
Inkonsistenz der Originaldefinition 255
probabilistische Interpretation 251, 257
Rechenregeln 253
- Sicherheitsgrad 237
- σ -Algebra 241
- Silbe 474
- Silbentrennung 478, 483
- Simon, H. 352
- simplified memory-bounded A* 103
- simulated annealing 93
- Simulation
stochastische 631
- Simulierte Ausglühen 93
- SIPE 350, 353
- Situation 330–334
- Situationskalkül 173, 330
- situierter Aktivität 329, 338
- Situertheit 10, 529
- Skizzen 323
- Skolemform 139

- SKOS 601
 Slot 174
 Slow Feature Analysis 383
 SNLP 340, 345
 SNOMED 587, 597
 soft constraints 292
 Soft-Constraints 225
 Sorites-Paradoxon 246
 Soziale Gesetze 539
 Sozialpsychologie 54
 Sozionik 542
 SPARQL 589, 591
 Spielbaumsuche 624
 Spielbeschreibung 614
 Spielen
 universelles 613
 Spieler 615
 Spielleiter 619
 Spielprogramme
 universelle 613
 Spielregel 614, 615, 619
 Spieltheorie 540, 542, 645
 Spielzustand 614
 Sprache 57, 473, 560
 Spracherkennung 564
 Sprachgenerierung 514
 Sprachgenerierungssystem 515
 Sprachmodell 508
 Sprachtechnologie
 Anwendungen 480
 Sprachverarbeitung 58, 473–520, 572
 inkrementelle 518
 Sprachverarbeitungssystem 60
 Sprachverstehen 174, 478
 Sprechakt 476, 490, 534, 560
 Sprecherintention 490
 Stützvektoren 429
 Stützvektormethode 427
 Kernfunktionen 432
 Optimierungsproblem 430
 stabiles Modell 194
 Startzustand 75
 statistischen Lerntheorie 427
 Stellgröße 282
 Stellung 614, 619, 626
 Stereopsis 394
 Stichprobenkomplexität 435
 von endlichen Räumen 435
 von unendlichen Räumen 438
 Stigmergie 536
 stochastisch unabhängig 244
 strenge Beweisbarkeit 188
 STRIPS 331–333
 Strukturbeschreibung 485, 487
 Strukturdynamik 376
 Strukturrepräsentation
 flache 504
 Substitution 159
 Subsumption 154
 Subsumptionsalgorithmus 121
 Subsumptionsarchitektur 532
 Subsumptionshierarchie 116
 Suche 4, 22, 75–103, 143, 212–214, 216,
 317, 334, 453, 499, 585, 614, 629
 absteigende 444
 als Optimierungsproblem 92
 blinde 79
 erschöpfende 81
 facettierte 606
 generische 78
 gierige 94
 heuristische 79, 89, 632
 in Graphen 80
 informierte 79
 lokale 223
 mit schrittweiser lokaler
 Verbesserung 91
 schrittweise verbreitende 87
 schrittweise vertiefende 88
 semantische 604
 stochastische 628
 uninformierte 79, 82, 626
 zielorientierte 79
 Suchgraph 76
 Suchmaschine 585
 Suchstrategie 79
 unfaire 81
 Suchtechniken 212
 Suchverfahren
 optimales 81
 vollständiges 81
 Support Vector Machine 427
 Sussman, G. 344
 Symbolverarbeitung 22

- Symbolverarbeitungsmodell 30
Symmetrie 241, 639
Symmetri axiom 268
Synapse 359, 361, 363
 Übertragungsgewicht 365, 366
 Hebb- 364
Syntax 474
Syntaxabhängigkeit 188
System
 übergeordnetes 40
 kognitives 21
 offenes 540
 situiertes 15
 verteiltes 587
Systemarchitektur
 für die Sprachverarbeitung 516
Systeme
 intelligente 1
Szenario 189
- T**
t-Conorm 249
t-Norm 248
Tagging 504
Term
 linguistischer 283
Textbedeutung 475
Textgenerierung
 automatische 483
Textzusammenfassung 484
Theorembeweisen 130
THEORIST 190
Theory-of-Mind 55
Tiefensuche 85
 iterative 636
Tippfehlersuche 483
Toleranzrelation 287
Trennflächenklassifikator 510
Trennung 267
 d- 270
 u- 270
triangular conorm 249
triangular norm 248
Triangulierung 280
Turing, A. 30, 476, 613
Turn-Taking 561, 573
- U**
u-Trennung 270
Überanpassung 387
Übergangsfunktion 76
Übersetzung
 maschinelle 482
Übersetzungssystem 482
UCPOP 344, 345
UCT-Bonus 629
Umwelt 529
unabhängig 244
 bedingt 267
Unabhängigkeit
 bedingte 266
Unabhängigkeitsaxiom 257, 259
Unabhängigkeitsgraph
 bedingter 267, 272
Unabhängigkeitskarte 272
Unentscheidbarkeit 124
Unifikation 145, 158, 160, 161, 221, 334, 494, 622
Unifikationsalgorithmus 160
Unifikationsgrammatik 492
Unifikator 159
uniform cost search 84
unmögliches Ereignis 241
unscharfe Menge 235
unsicheres Wissen 235
Unsicherheit 237, 300
unvereinbar 240, 242
URI 587, 590, 600
uvw:fig.decomp.poss 264
- V**
vage Regel 239, 248
vages Wissen 235
Vagheit 237, 246
Vapnik-Chervonenkis-Dimension 436
Variable
 linguistische 235
VC-Dimension 436
 Hyperebenen 436
 von endlichen Räumen 436
Verbundbaum 277
Vereinigung 248
Vereinigungsmenge 248

- Verfahren
 bildgebende 28
- Verhaltensgenerierung 565, 575
- Verhaltensverarbeitung 564
- Verkörperte Kommunikation 559–579
- Verkörperung 12, 559
- Verstärkungslernen 461
 dynamische Programmierung 463
 Q-Learning 465
- Verteilte Künstliche Intelligenz 527
- Verteilung 260
 bedingte 239
 Possibilitäts- 290
 Rand- 239
 Wahrscheinlichkeits- 235
- Vertrauen
 A-posteriori- 257
 A-priori- 257
 Maß des 251
- Vertrauensgrad 237, 251
- Vertrauensgradänderung 251
- Vertrauenswürdigkeit 602
- Verzweigungsproblem 330
- Vokabular 116, 303, 603
- vollständige Ereignisdisjunktion 245
- vollständige Wahrscheinlichkeit 245
- Vorbedingung 177, 332
- Vorhersageproblem 330
- Vorwärtssuche 79
- Voting 542
- W**
- Wahrheitsgrad 246, 247
- Wahrheitswert 236
- Wahrnehmung 35, 36
- Wahrscheinlichkeit 236, 240, 242
 A-posteriori- 245, 251
 A-priori- 251
 bedingte 243
 Elementar- 241
 empirische Deutung 243
 frequentistische Deutung 243
 klassische Definition 240
 logische Deutung 243
 personalistische Deutung 243
 subjektive 257
 subjektive Deutung 243
- vollständige 245
 von Hypothesen 245
- Wahrscheinlichkeitsraum 243
- Wahrscheinlichkeitsverhältnis 258
- Wahrscheinlichkeitsverteilung 235
- Watson 604
- Wettbewerbslernen 384
- Widerlegungsvollständigkeit 147
- widersprüchliche Defaults 176
- Widrow, B. 366, 377
- Wiedererkennen 42
- Wiesel, T.N. 399
- Wissen 7, 236, 637
 A-priori- 259
 begriffliches 108
 deskriptives 106
 explizites 107
 gemeinsames 108
 generisches 259
 grammatisches 513
 implizites 107
 kausales 108
 konditionales 108
 lexikalisches 513
 probabilistisches 239
 propositionales 106
 prozedurales 107
 terminologisches 108
 unsicheres 235
 vages 235
 verteiltes 108
- Wissensbasiertes System 235
- Wissensbasis 7, 109, 282, 599
- Wissenscontainer 303
- Wissensebene 7
- Wissensentdeckung 409
- Wissenserwerb 41
- Wissensmodellierung 7
- Wissensrepräsentation 7, 105, 110, 113, 130, 306, 479, 491–499, 533, 599, 614
 deklarative 111
- wohlfundierte Semantik 195
- Wortartentagging 510
- Worterkenneung 58
- Wortform 474, 486
- Worthypothesengraph 517

- X**
- XML 588
 - XOR-Problem 358, 373
 - Lösungbeispiele 398
- Z**
- zerlegbar
 - bzgl. eines gerichteten Graphen 274
 - bzgl. eines ungerichteten Graphen 273
 - Zerlegung 260
 - einer Relation 261
 - einer Wahrscheinlichkeitsverteilung 264
 - Zerlegungsaxiom 268
 - Zielheuristik 634, 636
 - Zinc 222
 - Zirkuläre Begründungen 184
 - Zirkumskription 177, 185
 - zufälliges Ereignis 240
- Zug 614
- legaler 620
- Zugehörigkeitsgrad 246, 247, 284
- Zusammenziehungsaxiom 268
- Zustand 75, 236, *siehe* Situation, 334, 461, 530
- Zustandsübergangssystem 333
- Zustandsmenge
 - explizit vorgegebene 77
 - implizit vorgegebene 77
- Zustandsraumrepräsentation 75
- Zuverlässigkeit, Stufen der 191
- Zuverlässigkeitsgrad 187
- Zyklen einer Terminologie 120
- zylindrische Erweiterung 263