

Dissertation

Modellierung syntaktischer Strukturen natürlicher Sprachen mit Hilfe von Graphgrammatiken

von
Frank Börncke

Fakultät für Mathematik und Informatik
der Universität Passau

26. Juli 2000

Danksagung

Die vorliegende Arbeit entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Theoretische Informatik von Prof. Brandenburg an der Universität Passau.

Mein besonderer Dank gilt Professor Brandenburg und Professor Felix, die mein Interesse für das interdisziplinäre Thema dieser Arbeit geweckt haben. Darüber hinaus boten sie mir die Möglichkeit, mich detailliert mit den damit verbundenen Fragestellungen zu beschäftigen. Mit ihrer Diskussionsbereitschaft und mit konstruktiver Kritik haben sie erheblich zum Gelingen dieser Arbeit beigetragen.

Insbesondere Konstantin Skodinis und Adi Palm standen mir bei zahlreichen fachlichen Diskussionen als wertvolle Gesprächspartner zur Verfügung. Ihre Ideen und Anregungen haben mir wiederholt neue Einsichten in das Thema eröffnet.

Unzählige Tips und Ratschläge von Sabine Bachl haben mir den Zugang zum Mysterium \LaTeX erleichtert. Mit ihrer Hilfe war es mir möglich, meine Inhalte auch in eine angemessene Form zu bringen.

Schließlich danke ich meinem Bruder Thomas, der viel Zeit und Mühe darin investiert hat, sich als Außenstehender in das Thema dieser Arbeit hineinzudenken. Seine kritischen Kommentare waren mir bei der Fertigstellung des Manuskripts eine große Hilfe.

Frank Börncke

Inhaltsverzeichnis

Danksagung	3
1 Einleitung	11
1.1 Überblick	11
1.2 Ausgangssituation	12
1.3 Motivation für die Formalisierung von GB	12
1.4 Syntaktische Strukturen und Graphgrammatiken	13
1.5 Parsing syntaktischer Strukturen	14
1.6 Erweiterte Wortgrammatiken für natürliche Sprachen	14
1.7 Einsatzmöglichkeiten der Ergebnisse	14
1.8 Abgrenzung zu anderen Arbeiten	15
1.9 Gliederung der Arbeit	16
1.10 Zusammenfassung und Ausblick	17
2 Formale Grundlagen und Terminologie	19
2.1 Graphentheorie und formale Sprachen	19
2.2 Graphgrammatiken	19
2.2.1 edNCE Graphgrammatiken	20
2.2.2 Darstellung von Produktionen	21
2.2.3 Produktionsanwendungen	21
2.2.4 Beispiele	23
2.2.5 Eigenschaften von Graphgrammatiken	23
2.3 Aufgabenstellung für diese Arbeit	25
3 Sprache als Forschungsgegenstand der Linguistik	27

3.1	Eine erste Abgrenzung	27
3.2	Syntax als eigenständige linguistische Disziplin	30
3.2.1	Schriftsprache besteht aus Sätzen	30
3.2.2	Sätze besitzen eine Struktur	31
3.2.3	Unterschiedliche Verwendungsweisen des Begriffs <i>Grammatik</i>	33
3.3	Natürliche Sprachen und die Chomsky Hierarchie	33
3.3.1	\mathcal{NL} ist nicht kontextfrei	34
3.3.2	Kontextsensitive Sprachen sind zu stark für \mathcal{NL}	36
3.3.3	Alternativen und Verfeinerungen der Chomsky Hierarchie	37
3.4	Modelle in der Linguistik	39
3.4.1	Sprachkompetenz und Performanz	40
3.5	Universalgrammatik für generische Algorithmen	41
3.5.1	Das logische Problem des Spracherwerbs	41
3.5.2	Eine angeborene Universalgrammatik liefert Basisinformationen . . .	42
3.5.3	Motivation: Universalgrammatik verspricht generische Lösungen . .	43
3.6	Government & Binding (GB)	43
3.6.1	GB als repräsentationelles Modell	44
3.6.2	Derivationelle vs. repräsentationelle Mechanismen	45
3.7	Zusammenfassung	47
4	Eine graphentheoretische Charakterisierung von GB	51
4.1	Grundstruktur für einen GB Syntaxbaum	53
4.1.1	Relationen zwischen Knotenpaaren	59
4.2	Kantenbeschriftungen im Syntaxbaum	60
4.2.1	Ordnungsrelation im Syntaxbaum	61
4.2.2	Exkurs: Syntaktische Struktur beeinflusst Perzeptionseigenschaften	63
4.3	Namen und Typen im Syntaxbaum	65
4.3.1	Namen im Syntaxbaum	66
4.3.2	Typen im Syntaxbaum	70

4.3.3	Übersicht: Attribute von Knoten und Kanten	71
4.4	Querbeziehungen zwischen Knoten im Syntaxbaum	72
4.4.1	Ketten überlagern die zugrundeliegende Baumstruktur	73
4.4.2	α -Ketten modellieren GB-Bewegungsprozesse	74
4.4.3	β -Ketten modellieren bindungstheoretische Beziehungen	76
4.4.4	Ein Prinzip für Ketten	78
4.5	Geordnete Nachbarschaften im Syntaxbaum	79
4.6	Übersicht: Die fünf Prinzipien in einer Kurzfassung	86
4.7	Einige Satzanalysen als Fallbeispiele	87
4.8	Ungrammatische Sätze	91
4.9	Zusammenfassung und Bewertung der Vorgehensweise	94
4.9.1	Selbstbeschränkung und Auswahlkriterien	94
4.9.2	Die fünf Prinzipien beschreiben nicht zu viel	95
4.9.3	Die Prinzipien sind konsistent formuliert	95
4.9.4	Redundanz im Modell kann nicht ausgeschlossen werden	96
4.9.5	Weiteres Vorgehen	96
5	Formale Analyse syntaktischer Strukturen	97
5.1	Vom Syntaxbaum zum Syntaxgraph	97
5.1.1	Problem – Der Baum hat zu viele Knotenbezeichner	97
5.1.2	Lösung – Mit zusätzlichen Kanten reicht ein endliches Alphabet	98
5.1.2.1	Elimination der Alphabete \mathcal{I} und \mathcal{J}	98
5.1.2.2	Elimination der Alphabets \mathcal{B}	99
5.1.2.3	Vereinfachung des Knotenalphabets	100
5.1.3	Weitere Begriffe und Definitionen	101
5.1.4	Beispiele für die Transformation Syntaxbaum \leftrightarrow Syntaxgraph	101
5.1.5	Zusammenfassung und Formulierung der weiteren Aufgabenstellung	102
5.2	Eigenschaften von Syntaxgraphen	104
5.2.1	Syntaxgraphen: gradbeschränkt, zusammenhängend und zyklensfrei	104

5.2.2	Syntaxgraphen sind nicht planar	106
5.2.3	Syntaxgraphen haben einen unbeschränkten Separator	107
5.2.4	Die Baumbreite von Syntaxgraphen ist unbeschränkt	114
5.2.5	Zusammenfassung	115
5.3	Syntaxbäume und konfluente Graphgrammatiken	115
5.3.1	Graphgrammatik, Konfluenz und Separatoren	116
5.3.2	Eine konfluente Graphgrammatik GG_{deriv} für Syntaxgraphen . . .	116
5.3.2.1	Startkonfiguration für die Graphgrammatik GG_{deriv} . . .	117
5.3.2.2	Nominalphrasen	118
5.3.2.3	Adjektivphrasen/Adverbialphrasen	119
5.3.2.4	Präpositionalphrasen	120
5.3.2.5	Die funktionale Kategorie C	121
5.3.2.6	Die funktionale Kategorie I	121
5.3.2.7	Verbalphrasen	122
5.4	Eigenschaften von GG_{deriv}	125
5.5	Zusätzliche Produktionen modellieren Querkanten	127
5.5.1	Erweiterung 1: Querkanten für bindungstheoretische Regularitäten	128
5.5.1.1	Neue Produktionen für die funktionale Kategorie I . . .	128
5.5.1.2	Neue Produktionen für die funktionale Kategorie C . . .	130
5.5.1.3	Neue Produktionen für Verbalphrasen	131
5.5.1.4	Eigenschaften der zusätzlichen Produktionen	135
5.5.2	Erweiterung 2: Querkanten für Bewegungen	136
5.5.2.1	Neue Produktionen für die funktionale Kategorie C . . .	136
5.5.2.2	Neue Produktionen für die funktionale Kategorie I . . .	137
5.5.2.3	Neue Produktionen für Verbalphrasen	138
5.5.2.4	Eigenschaften der zusätzlichen Produktionen	142
5.5.3	Erweiterung 3: α -Ketten und β -Ketten gleichzeitig	142
5.5.4	Erweiterung 4: Kontextsensitive Phänomene	143
5.6	Analyse von GG_{deriv*} bezüglich des Wortproblems	146

5.7	Zusammenfassung	147
6	Von Graphgrammatiken zu Wortgrammatiken	149
6.1	Ziel: Entwurf einer effizienten Wortgrammatik	149
6.2	Entwurf einer Graphgrammatik GG_{chain}	151
6.3	Analyse von GG_{chain}	155
6.4	Right Adjoining Grammars	156
6.5	Charakterisierungen der Klasse \mathcal{RAL}	158
6.6	Wortproblem und Parsing	161
6.7	Abgeschlossenheitseigenschaften von \mathcal{RAL}	164
6.8	Abgrenzung zu anderen Arbeiten	166
6.9	Zusammenfassung	169
7	Zusammenfassung der Arbeit	171
7.1	Überblick	171
7.2	Zum Umgang mit empirischen Problemen	172
7.3	GB besitzt ein generisches Potential	173
7.4	Formalisierung von GB und Analyse von Syntaxgraphen	173
7.5	Elementprobleme sind in Polynomialzeit lösbar	174
7.6	Wortgrammatiken	175
7.7	Die Formalisierung liefert auch linguistische Einsichten	176
7.8	Abgrenzung zu anderen Arbeiten	176
7.9	Ausblick	177
Anhang		179
	Verzeichnis der Definitionen	182
	Verzeichnis der Prinzipien von GB	185
	Verzeichnis der Abbildungen	186
	Literatur	190

Generative grammar and formal language theory have always been close, if frequently uncomfortable, siblings. Over the years, as the disciplines have developed independently, they have grown apart to the point that it is no longer entirely clear that either has anything to say to the other. More fundamentally, their universes of discourse have diverged to the extent that it is no longer entirely clear that they *can* say anything to each other. The connections between the concepts of interest in the two areas and the vocabularies in which they are expressed have been reduced to such a superficial level that it is not entirely clear that results in one have any significant consequences for the other. The fields no longer speak the same language.

JAMES ROGERS [ROG98]

1 Einleitung

1.1 Überblick

Die vorliegende Arbeit erschließt durch die Formalisierung einer linguistischen Theorie Möglichkeiten zum Entwurf generischer Verfahren zur Verarbeitung natürlicher Sprachen. Zu diesem Zweck setzen wir Graphsprachen für die Modellierung syntaktischer Strukturen ein. Damit lassen sich Ergebnisse der linguistischen Forschung mit Begriffen der Graphentheorie beschreiben und bewerten.

Zu diesem Ansatz motiviert der Umstand, daß in der Linguistik im Rahmen der Syntax jedem Satz einer natürlichen Sprache eine *nichtsequentielle* Struktur zugesprochen wird. Diese Struktur überlagert die lineare Wortfolge, die wir als *Satz* kennen. Eine Menge solcher syntaktischen Strukturen – die wir mit Graphen modellieren können – betrachten wir als *Graphsprache*. Die Arbeit zeigt, wie sich solche Graphsprachen mit Hilfe von Graphgrammatiken beschreiben lassen.

Wie alle formalen Sprachen zeichnen sich auch Graphgrammatiken dadurch aus, daß sie mathematisch wohldefiniert sind. Dies stellt eine notwendige Voraussetzung dar, um Aussagen über eine Sprache zu beweisen. Von Interesse ist dabei vor allem die Untersuchung unendlicher Mengen. Das Ziel besteht dann darin, für sie eine endliche Beschreibung zu finden. Diese Aufgabe wird in der Regel von einer Grammatik erfüllt. Darüber hinaus ist man an erkennenden Algorithmen für Sprachen interessiert, die das Wortproblem effizient lösen. Bezüglich natürlicher Sprachen werden beide Aufgabenstellungen in dieser Arbeit mit Hilfe von Graphgrammatiken gelöst.

1.2 Ausgangssituation

Anders als bei den formalen Sprachen sind natürliche Sprachen von Haus aus *nicht* durch eine Grammatik gegeben, sondern vielmehr durch eine Sprachgemeinschaft, die sich dieser Sprache zum Zwecke der Kommunikation bedient. Dieser Umstand macht es notwendig, in einem ersten Arbeitsgang aus den gegebenen linguistischen Daten heraus eine formale Beschreibung zu *rekonstruieren*. Dazu werden in der Linguistik die Regularitäten von Sprachen nach verschiedenen Gesichtspunkten untersucht und entsprechend eine Reihe von Disziplinen unterschieden. In dieser Arbeit beschäftigen wir uns ausschließlich mit syntaktischen Phänomenen und gehen auf Themen wie Semantik, Phonetik, Phonologie oder Morphologie nicht ein.

Syntaktische Strukturen sind in der Linguistik von essentiellm Interesse. Mit ihrer Hilfe können Zusammenhänge und Ähnlichkeiten zwischen verschiedenen Sprachen erkannt und deutlich gemacht werden, die bei einer sequentiellen Sichtweise verborgen bleiben. Linguisten, die sich auf die Untersuchung unterschiedlicher Sprachfamilien spezialisiert haben, können sich so auf der Basis einer gemeinsamen Terminologie austauschen.

Für ein Modell syntaktischer Regularitäten greifen wir auf eine Theorie der Syntax von Noam Chomsky mit dem Namen „*Government and Binding*“ (GB) zurück. Diese vermag eine umfangreiche Menge von linguistischen Daten zu beschreiben und deren Zustandekommen zu erklären.

1.3 Motivation für die Formalisierung von GB

Aus algorithmischer Sicht ist GB vor allem deshalb interessant, weil hier mit dem Konzept einer Universalgrammatik ein *generischer Ansatz* verfolgt wird. Es wird angestrebt, für alle natürlichen Sprachen der Welt deren syntaktischen Strukturen auf der Grundlage einer einzigen und allen Sprachen gemeinsamen abstrakten Beschreibung zu charakterisieren. Viele Arbeiten in der linguistischen Literatur belegen, daß die universellen Gemeinsamkeiten zwischen verschiedenen Sprachen sehr viel umfangreicher sind, als es auf den ersten Blick den Anschein hat. Unterschiede zwischen den Sprachen werden in GB mit Hilfe von sprachspezifischen Parametern beschrieben. Mit diesen Parametern lassen sich zum Beispiel Wortstellungsphänomene elegant modellieren.

Die Ergebnisse dieser Arbeit zeigen durch eine Formalisierung der beteiligten Mechanismen Wege auf, dieses generische Potential von GB für praktische Anwendungen nutzbar zu machen. Für Algorithmen zur Sprachverarbeitung wie *Sprachgenerierung*, *Grammatikprüfung* oder *Übersetzung* ist es dann nicht mehr nötig, für die syntaktische Komponente einer jeden natürlichen Sprache von Grund auf ein eigenes Verfahren zu entwickeln.

Um für dieses Ziel eine Grundlage zu entwickeln, gehen wir in zwei Schritten vor: Erstens ist es nötig, die Regeln und Restriktionen von GB in eine graphentheoretische Terminologie zu übersetzen. Dabei bleibt der in der Linguistik zur Anwendung kommende Mechanismus erhalten. Die in der Literatur geläufigen Definitionen für GB werden in eine einheitliche Notation überführt. In einem zweiten Schritt wird diese formalisierte

Fassung von GB in Produktionen einer Graphgrammatik übersetzt. Dabei entwickeln wir den in dieser Arbeit zentralen Begriff des *Syntaxgraphen*, der den in der Literatur verbreiteten Begriff des *Syntaxbaums* erweitert. Diese Art und Weise der Charakterisierung von GB-Strukturen stellt einen neuartigen Ansatz dar.

1.4 Syntaktische Strukturen und Graphgrammatiken

Die Umsetzung beider Schritte erweist sich als nichttrivial und nicht automatisierbar. So reicht es einerseits nicht aus, zur Formalisierung von GB die entsprechenden Regeln und Prinzipien aus der linguistischen Literatur abzuschreiben. Da sie nicht als Regeln einer formalen Sprache konzipiert sind, ist es nötig, sie einzeln zu analysieren und dabei ihre Rolle für die Charakterisierung syntaktischer Strukturen herauszuarbeiten. Es wird deutlich, daß der von GB gewählte Mechanismus aufgrund seiner *repräsentativen* Beschreibungsweise nicht in die *derivationelle* Begriffswelt der formalen Sprachen paßt. Die nötige Umsetzung kann auch hier nicht automatisiert werden und wird deshalb konstruktiv erbracht.

Die vorliegende Arbeit demonstriert, daß und wie diese Ziele mit Graphgrammatiken erreicht werden können. Dabei gibt es mehrere Vorgehensmöglichkeiten. Wir machen in der Arbeit deutlich, wo alternative Herangehensweisen möglich sind und motivieren die von uns gesetzten Schwerpunkte. In diesem Sinne ist die Arbeit auch als *proof of concept* für die Anwendungsmöglichkeiten von Graphgrammatiken in der Linguistik zu verstehen.

Unabhängig von der Anzahl der modellierten Details können wir einige invariante Eigenschaften von Syntaxgraphen beweisen. Syntaxgraphen sind schlichte, azyklische, einfach zusammenhängende Graphen. Sie sind nicht planar und haben keinen beschränkten Separator. Letzteres impliziert, daß es Grenzen für die Anwendbarkeit von Graphgrammatiken geben muß. Als Ursache arbeiten wir universelle Phänomene heraus, die in allen bekannten natürlichen Sprachen vorkommen. Wir können sie jedoch auf einige esoterische Fälle von praktisch irrelevanten Satzkonstruktionen reduzieren. Dieses Ergebnis läßt sich auch so interpretieren, daß GB an der Umgangssprache gemessen zu mächtig ist, also zuviel beschreibt. Die kritischen Satzkonstruktionen, die wir in Beispielen vorstellen, kommen in der sprachlichen Praxis faktisch nicht vor.

Für praktische Anwendungsgebiete empfiehlt sich damit eine pragmatische Herangehensweise: Die überwiegende Menge syntaktischer Strukturen läßt sich adäquat mit boundary Graphgrammatiken beschreiben. Dazu zeigen wir, daß die in dieser Arbeit entwickelte Graphgrammatik alle Eigenschaften erfüllt, welche die GB-Theorie formuliert. Weitere Formalismen werden nicht benötigt. Unter diesem Gesichtspunkt ist unsere Modellierung 'kompakt', was für formale Betrachtungen gegenüber GB einen großen Vorteil darstellt.

1.5 Parsing syntaktischer Strukturen

Soweit bis jetzt dargestellt generieren Graphgrammatiken syntaktische Strukturen und beschreiben sie damit *konstruktiv*. Darüber hinaus ist auch das *Elementproblem* interessant, welches die Herangehensweise umkehrt: Hierbei ist einerseits zu überprüfen, ob ein gegebener Graph eine syntaktische Struktur darstellt und darüber hinaus, ob die dazugehörige Zeichenkette zur Menge der natürlichen Sprachen gehört.

Die Modellierbarkeit syntaktischer Strukturen mit Graphgrammatiken zeigt Wege auf, diese Fragen unter Anwendung von bereits bekannten Ergebnissen zur Theorie der formalen Sprachen effizient zu lösen. Dabei erweist es sich als hilfreich, vorab gezeigt zu haben, daß Graphgrammatiken mit der boundary Eigenschaft syntaktische Strukturen beschreiben. Zusammen mit anderen Graphinvarianten (Syntaxgraphen sind zusammenhängend und haben beschränkten Grad) stellt dies sicher, daß das Wortproblem in polynomialer Zeit lösbar ist. Bekannte Verfahren führen allerdings zu Algorithmen mit großen Exponenten. Die praktische Anwendbarkeit dieser Techniken bleibt aber unangetastet, solange n klein ist. Dies ist bei Sätzen natürlicher Sprachen typischerweise immer der Fall.

1.6 Erweiterte Wortgrammatiken für natürliche Sprachen

Für sprachspezifische Anwendungen – wenn also nur Verfahren für eine ausgewählte Sprache gesucht sind – reicht es unter Umständen aus, mit einer Wortgrammatik zu arbeiten, welche die terminalen Zeichenketten direkt beschreibt. Es ist dann nicht nötig, die syntaktische Struktur in einem eigenen Zwischenschritt aufzubauen.

Ergebnisse der Literatur zeigen, daß kontextfreie Grammatiken für diesen Zweck nicht mächtig genug sind. Gleichzeitig verbietet es sich unter dem Gesichtspunkt der Effizienz, als Alternative auf kontextsensitive Grammatiken zurückzugreifen. Wir stellen die entsprechenden Argumente aus der Literatur zusammen und entwickeln aus den Mechanismen der Graphgrammatik für Syntaxgraphen heraus einen neuen Formalismus für eine Wortgrammatik, die wir als *Right Adjunct Grammar* bezeichnen. Die so charakterisierte Klasse ist mächtiger als die der kontextfreien Sprachen. Wir beweisen einige Eigenschaften dieses Grammatikformalismus und demonstrieren, daß das dazugehörige Wortproblem in $O(n^4)$ lösbar ist. Das vorgestellte Verfahren stellt eine Erweiterung des CYK-Algorithmus für kontextfreie Sprachen dar.

1.7 Einsatzmöglichkeiten der Ergebnisse

Syntaxgraphen und Graphgrammatiken erweisen sich zusammen als ein geeignetes Hilfsmittel, um die vielfältigen syntaktischen Phänomene zu beschreiben, die die Linguistik in natürlichen Sprachen beobachtet.

Während Syntaxgraphen sich als zugrundeliegende Datenstruktur anbieten, folgt aus dem

Zugang über Graphgrammatiken die Existenz effizienter Verfahren für die Verwaltung und den Aufbau syntaktischen Strukturen.

Die gezeigten Parsingeigenschaften von boundary Graphgrammatiken sind nützlich für die Implementation von Systemen, die den Prozeß des automatisierten ‘Verstehens’ natürlicher Sprache modellieren. Hier stellt die syntaktische Analyse der Eingabe eine wesentliche Komponente in der Abarbeitung dar. Umgekehrt kann die Möglichkeit, Syntaxgraphen systematisch über Graphgrammatiken zu erzeugen, dabei helfen, Systeme für automatische Sprachgenerierung zu entwerfen.

Beide Module zusammengenommen stellen eine zentrale Komponente für die Konstruktion von Übersetzern natürlicher Sprachen dar. Neue Ansätze eröffnen sich dabei durch den Zugang über eine Universalgrammatik: Dies ermöglicht den Entwurf generischer Algorithmen. Durch sprachspezifische Verfeinerungen können sie individuell an die jeweiligen Anforderungen angepaßt werden. Die Arbeit demonstriert, an welchen Stellen sich mit wenigen Parametern eine Reihe von sprachspezifischen Regularitäten steuern lassen.

Auch aus linguistischer Sicht eröffnet der formale Zugang neue Ansätze. Wir beschäftigen uns mit dem Umstand, daß Sätze nicht notwendig schlechter zu verstehen sind, wenn sie lang werden. Dabei machen wir den Zusammenhang deutlich, daß bei Syntaxgraphen mit wachsendem Separator auch die dazugehörenden Sätze schwerer zu verarbeiten sind. Darüber hinaus beobachten wir weitere Zusammenhänge zwischen strukturellen Eigenschaften des Syntaxgraphen und empirisch feststellbaren Verarbeitungsaspekten.

Formale Ergebnisse dieser Art können empirische Untersuchungen in der Linguistik motivieren und neue Einsichten in syntaktische Zusammenhänge eröffnen.

1.8 Abgrenzung zu anderen Arbeiten

Da in der Literatur üblicherweise mit Syntaxbäumen, nicht aber mit Syntaxgraphen gearbeitet wird, ist der Ansatz, syntaktische Strukturen über Graphgrammatiken zu definieren, bis jetzt noch nicht systematisch untersucht worden.

Die Idee, syntaktische Strukturen über Produktionen zu beschreiben, ist – unter ganz anderen Vorzeichen – im Zusammenhang mit *Tree Adjoining Grammars* (TAGs) untersucht worden. Dieser Grammatikformalismus beschreibt keine Zeichenketten, sondern deren zugrundeliegende Baumstrukturen. Die Erweiterung von Bäumen zu Graphenstrukturen findet aber auch hier nicht statt. Primär werden dort formalsprachliche Eigenschaften untersucht, und daher wird auf universalgrammatische Aspekte nicht eingegangen. Auch werden graphentheoretische Eigenschaften von syntaktischen Strukturen in der TAG-Literatur nicht thematisiert.

Erweiterungen von kontextfreien Grammatiken zur Modellierung natürlicher Sprachen sind in der Literatur mehrfach vorgestellt worden. Hervorzuheben sind dabei eine Reihe von Arbeiten, welche auf der Basis ganz verschiedener Ansätze ein und dieselbe Klasse definiert haben und die als *schwach kontextsensitive Sprachen* bekannt sind. Typische

nicht kontextfreie Phänomene aus den natürlichen Sprachen bekommt man mit diesen Grammatiken in den Griff. Zur Lösung des Wortproblems sind Verfahren in $O(n^6)$ bekannt.

Die in dieser Arbeit entwickelten *Right Adjunct Grammars* sind weniger mächtig als die schwach kontextsensitiven Sprachen, erkennen aber alle die Konstruktionen, die in der Literatur immer wieder als Standardbeispiele für nicht kontextfreie Sprachen zitiert werden. Zugleich kann das Wortproblem schneller gelöst werden.

Ein Automatenmodell, welches die Sprachen einer *Right Adjunct Grammar* erkennt, benötigt zusätzlich zu einem Keller noch ein weiteres Speichermedium. Wir gehen kurz auf einer Reihe von Erweiterungen ein, die in der Literatur diskutiert worden sind und erörtern die Verwandtschaft mit unserem Verfahren, was es ermöglicht, den Ansatz zu klassifizieren.

1.9 Gliederung der Arbeit

Die Arbeit gliedert sich in folgende Abschnitte: Die formalen und terminologischen Grundlagen werden im *zweiten Kapitel* zusammengefaßt, wobei wir auf Graphgrammatiken und Graphsprachen ausführlicher eingehen.

Im *dritten Kapitel* wird in die Begriffswelt der Linguistik eingeführt, insoweit sie für diese Arbeit relevant ist. Dabei gehen wir insbesondere auf Fragestellungen ein, die für ein Verständnis von GB wichtig sind. Damit wird deutlich, warum wir erst Eigenarbeit investieren müssen, um die Ergebnisse von GB als Graphsprache beschreiben zu können.

Mit der Formalisierung von GB beschäftigt sich ausführlich das *vierte Kapitel*, in dem ausgewählte Prinzipien zu einer Menge von Eigenschaften umformuliert werden, die syntaktisch korrekte Baumstrukturen charakterisieren. Die resultierenden Graphen werden als *Syntaxbäume* bezeichnet. Als Ergebnis erhalten wir ein formales Syntaxmodell für ein Fragment der englischen Sprache, welches sich auf graphentheoretische Terminologie stützt.

Im *fünften Kapitel* überführen wir die formal spezifizierten Syntaxbäume in *Syntaxgraphen*. Diese Modifikation ist nötig, weil Syntaxbäume von GB unendliche Knotenalphabete verwenden. Diese Eigenschaft ist mit dem Graphenmodell unverträglich. Die Einführung von Syntaxgraphen löst das Problem. Auf dieser Grundlage können wir eine Graphgrammatik entwickeln, welche ausschließlich GB-kompatible Graphen erzeugt.

Das *sechste Kapitel* zeigt, wie sich mit Hilfe von Graphgrammatiken nicht nur Syntaxgraphen, sondern auch Kettengraphen beschreiben lassen, die der linearen Wortreihenfolge entsprechen. Die Ergebnisse dieser Modifikation münden in die Entwicklung einer Wortgrammatik (*Right Adjunct Grammar*), welche nützliche Eigenschaften für die Modellierung natürlicher Sprachen besitzt.

1.10 Zusammenfassung und Ausblick

Die Arbeit zeigt, wie sich die Resultate der linguistischen Forschung unter formalsprachlichen Gesichtspunkten klassifizieren und für effiziente Algorithmen nutzbar machen lassen.

Die Ergebnisse lassen sich in verschiedener Hinsicht praktisch einsetzen: Mit Hilfe von Graphgrammatiken können syntaktische Strukturen, die den GB-Restriktionen genügen, auch über eine formale Sprache beschrieben werden, was die Klassifizierung und nachträgliche Erweiterung um zusätzliche Regularitäten erleichtert. Da das Wortproblem in Polynomialzeit lösbar ist, kann die Korrektheit von syntaktischen Strukturen effizient überprüft werden.

Bezüglich der Beschreibung linearer Zeichenketten wird deutlich, daß die Erweiterung um notwendige kontextsensitive Mechanismen nicht notwendig mit einer Verschlechterung der Komplexität erkaufte werden muß.

Schließlich eröffnet und weist der Ansatz über eine universalgrammatische Beschreibung syntaktischer Regularitäten den Weg zum generischen Entwurf linguistischer Algorithmen: Wo mit klassischen Ansätzen für jede Sprache ein eigenes Verfahren entwickelt werden muß, wird in Zukunft die Herangehensweise über parametrisierte Techniken viel Arbeit und Zeit im Entwurfsprozeß ersparen.

Note for the reader: due to inadequacies of my word processor, Γ is to be taken as lower-case gamma, λ as lower-case lambda.

NOAM CHOMSKY IN [CHO94]

2 Formale Grundlagen und Terminologie

2.1 Graphentheorie und formale Sprachen

Für das Verständnis dieser Arbeit werden zentrale Begriffe der Graphentheorie und der Theorie der formalen Sprachen als bekannt vorausgesetzt.¹ Eine Wortgrammatik ist ein Viertupel $WG = (N, T, P, S)$, wobei unterschiedliche Typen von Grammatiken im Sinne der Chomsky Hierarchie klassifiziert werden. Für eine Wortgrammatik WG bezeichnet $L(WG)$ die von WG beschriebene Sprache, also die Menge der vom Startsymbol aus ableitbaren terminalen Wörter. Ein Graph $G = (V(G), E(G))$ ist durch eine Menge von Knoten V und Kanten E beschrieben. Diese können markiert sein. Dann beschreiben Funktionen $m_V : V \rightarrow \Sigma$ und $m_E : E \rightarrow \Delta$ die Markierungen mit einem Knotenalphabet Σ und einem Kantenalphabet Δ .

Wegen der zentralen Bedeutung für diese Arbeit folgt eine kurze Einführung in den Graphgrammatik-Formalismus. Der mit diesem Konzept vertraute Leser kann direkt mit Kapitel 3 fortfahren.

2.2 Graphgrammatiken

Graphgrammatiken ermöglichen es, mit Hilfe einer endlichen Beschreibung eine unendliche Menge von Graphen zu generieren. Im Rahmen einer Produktionsanwendung werden ausgewählte Knoten oder Kanten eines Graphen durch einen neuen Graphen ersetzt und dieser dann in den alten Kontext eingebettet. In der Regel besteht die linke Seite einer Produktion dabei nur aus einem Symbol. In diesem Sinne können Graphgrammatiken als Erweiterung von Wortgrammatiken angesehen werden, wobei die Produktionen hier Graphen statt Zeichenketten manipulieren. Der Ersetzungsschritt und die Details der Einbettung werden durch Produktionsregeln eindeutig und vollständig beschrieben.

Die Literatur untersucht eine Reihe verwandter Formalismen mit zum Teil sehr un-

¹ Als Grundlage verweisen wir auf [Har78] und [HU79] oder auf [RS97]

terschiedlichen Konzepten. Manche Formalismen erweitern das Graphenmodell zu Hypergraphen, was zu Hyperkantenersetzungssystemen (Hyperedge-replacement grammars, HR-Grammars) führt.² Grundlegende Formalismen arbeiten ohne Kantenbeschriftungen und sind unter den Bezeichnungen NCE (*neighbourhood controlled embedding*)³ oder NLC (*node label controlled*)⁴ bekannt. Abhängig vom Typ der Graphgrammatik bestimmt der Kontext die Anwendbarkeit einer Produktion. Dabei werden verschiedene Ansätze unterschieden, wie weit der Kontext im Rahmen eines Ersetzungsschrittes modifiziert werden kann. Dies hat unmittelbar Folgen für die Lokalitätseigenschaften und damit auch für die Mächtigkeit des generierenden Formalismus.⁵ Mit dem Wortproblem verhält es sich naheliegenderweise umgekehrt: Je mächtiger die beschreibbaren Sprachen sind, um so schwieriger ist das dazugehörige Wortproblem zu lösen.⁶

Gemeinsam ist den unterschiedlichen Ansätzen, daß sie als Konstrukt zur Beschreibung einer formalen Sprache ohne Ausnahme *derivationeller Natur* sind in dem Sinne, daß sie ausgehend von einem Startsymbol mit Hilfe einer endlichen Menge von Produktionen eine potentiell unendlich große Menge von Graphen beschreiben. Diese Eigenschaft wird uns in Abschnitt 3.6.1 noch ausführlich beschäftigen, weil sie zu einer zentralen Aufgabenstellung dieser Arbeit hinleitet.

2.2.1 edNCE Graphgrammatiken

In dieser Arbeit werden edNCE-Graphgrammatiken verwendet.

DEFINITION 1 (*edNCE-GRAPHGRAMMATIK*): Eine *edNCE* (*edge directed label neighbourhood controlled embedding*) Graphgrammatik (nachfolgend kurz: Graphgrammatik oder *GG*) ist ein Quadrupel $GG = (N, T, P, S)$ mit

- $N = N_V \cup N_E$: Alphabet der nichtterminalen Zeichen mit nichtterminalen Knotenmarkierungen N_V und nichtterminalen Kantenmarkierungen N_E .
- $T = T_V \cup T_E$: Alphabet der terminalen Zeichen mit terminalen Knotenmarkierungen T_V und terminalen Kantenmarkierungen T_E .
- $P = \{(A_i, R_i, C_i) | 1 \leq i \leq t\}$ ist eine endliche Menge von t Produktionen mit
 - $A_i \in N_V$ Knotenmarkierung der linken Seite
 - R_i Graph der rechten Seite
 - C_i Einbettungsrelation mit

$$C_i \subseteq (N_V \cup T_V) \times (N_E \cup T_E) \times \{in, out\} \times V(R_i) \times (N_E \cup T_E) \times \{in, out\}$$

² [HK87], [Hab92]

³ [JR82]

⁴ [JR80a], [JR80b]

⁵ [Nag79]

⁶ In [MR90] wird gezeigt, daß allgemeine ELC-Sprachen so mächtig sind, daß jede nichttriviale Fragestellung unentscheidbar ist. Dies gilt entsprechend für das Wortproblem. Erst durch Beschränkungen erhält man Resultate, die effiziente Algorithmen in Aussicht stellen. So zeigt Vogler in [Vog90], wie man das Wortproblem für eingeschränkte Kantenersetzungssysteme in $O(n^3)$ löst.

- $S \in N_V$ ist das Startsymbol oder das Axiom.

Die Mengen N_V, N_E, T_V, T_E sind paarweise disjunkt. Die Einbettungsrelation erlaubt beim Ersetzungsschritt eine Reihe lokaler Modifikationen im Graphen. Kanten können *vervielfacht*, mit ausgewählten Knoten *verbunden*, *umbenannt*, in ihrer Richtung *umgedreht* oder ganz *gelöscht* werden.

2.2.2 Darstellung von Produktionen

Um Graphgrammatiken einprägsam und anschaulich darstellen zu können, wird eine Produktion und deren Einbettungsrelation jeweils in einem Kasten wiedergeben:

1. Die *linke Seite der Produktion* besteht aus einem Nichtterminalsymbol und ist jeweils oben links im Kasten angegeben.
2. Die *rechte Seite der Produktion* besteht aus den übrigen Knoten und Kanten *innerhalb* des Kastens. Diese werden im Rahmen einer Produktionsanwendung an Stelle der linken Seite in den zugrundeliegenden Graphen eingesetzt.
3. Die Einbettungsrelation kommt durch Kanten zum Ausdruck, welche die Begrenzung des Kastens überschreiten.

BEISPIEL 1: Wenn von der Möglichkeit, Kantenrichtungen und Kantenbezeichner zu vergeben, nicht Gebrauch gemacht wird, dann können die entsprechenden Bezeichner auch weggelassen werden, wie bei den beiden Produktionen in Abbildung 2.1. Hier handelt es sich nicht um eine edNCE-Graphgrammatik, sondern um eine eNCE-Graphgrammatik. Die Kanten sind ungerichtet. Die Produktionen dieser Graphgrammatik erzeugen die Menge aller vollständigen Graphen.

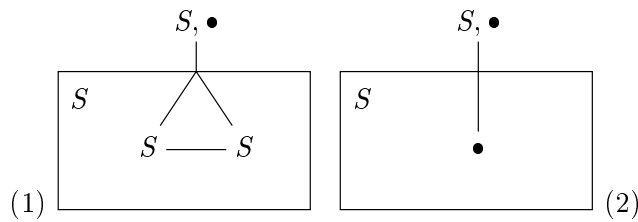


Abbildung 2.1: Zwei Produktionen für vollständige Graphen

2.2.3 Produktionsanwendungen

DEFINITION 2 (KONKRETER ERSETZUNGSSCHRITT): *Ein konkreter Ersetzungsschritt lässt durch Anwendung einer Produktion $p = (A, R, C)$ auf einen Knoten $v \in V(G)$ aus dem Graphen G den Graphen G' entstehen, kurz $G \Rightarrow_{(v,p)} G'$ genau dann, wenn:*

- Der Graph auf der rechten Seite der Produktion enthält zusätzliche Knoten, es gilt:
 $V(G) \cap V(R) = \emptyset$

- Die alte Knotenmenge wird durch die Knoten des eingebetteten Graphen ergänzt. Der Knoten v geht verloren: $V(G') = (V(G) \setminus \{v\}) \cup V(R)$
- Außerhalb der Einbettung bleibt der Graph unverändert. Der Ersetzungsschritt bewirkt nur eine lokale Modifikation⁷: $G_{|V(G) \setminus \{v\}} = G'_{|V(G') \setminus V(R)}$
- Die Knotenmarkierung für alte Knoten ändert sich nicht, für die neu hinzugekommenen Knoten entspricht sie der des eingebetteten Graphen:
 $m_{V_{G'}}(v) := m_{V_G}(v)$, falls $v \in V(G)$, $m_{V_{G'}}(v) := m_{V_R}(v)$, falls $v \in V(R)$.
- Die neuen Kanten entsprechen den alten Kanten ohne diejenigen, die zum eingebetteten Knoten hin oder davon wegführen. Die Knoten und Kanten des eingebetteten Graphen kommen neu hinzu. Schließlich werden abhängig von der Einbettungsrelation einer Produktion neue Kanten zwischen dem alten und dem neuen Graph hinzugefügt. Dieser Vorgang wird mit Knoten- und Kantenmarkierungen gesteuert. Die Wirkungsweise der Einbettungsrelation C ist festgelegt wie folgt:

$$\begin{aligned}
E(G') = & (E(G) \setminus \{(v, d_1, w), (w, d_1, v) \in E(G), w \in V(G)\}) \\
& \cup E(R) \\
& \cup \{(z, d_2, w) | w \in V(R), z \in V(G) \setminus \{v\}, m(z) = B, \\
& \quad (z, d_1, v) \in E, (B, d_1, in, w, d_2, in) \in C\} \\
& \cup \{(z, d_2, w) | w \in V(R), z \in V(G) \setminus \{v\}, m(z) = B, \\
& \quad (v, d_1, z) \in E, (B, d_1, out, w, d_2, in) \in C\} \\
& \cup \{(w, d_2, z) | w \in V(R), z \in V(G) \setminus \{v\}, m(z) = B, \\
& \quad (z, d_1, v) \in E, (B, d_1, in, w, d_2, out) \in C\} \\
& \cup \{(w, d_2, z) | w \in V(R), z \in V(G) \setminus \{v\}, m(z) = B, \\
& \quad (v, d_1, v) \in E, (B, d_1, out, w, d_2, out) \in C\}
\end{aligned}$$

Definition 2 beschreibt formal, wie die alten Kanten mit dem neu eingefügten Graphen verbunden werden müssen. Dabei können nur bereits bestehende inzidente Kanten modifiziert werden. Verbindungen zu Knoten, die über keine Kante erreichbar sind, können nachträglich nicht mehr aufgebaut werden. Sie sind im Verlauf weiterer Ersetzungsschritte unerreichbar geworden.

DEFINITION 3 (ABLEITUNGSSCHRITT): Ein Ableitungsschritt ist ein konkreter Ersetzungsschritt modulo Isomorphie. Dieser wird notiert als $G \Rightarrow_{(v,p)} G'$ oder in kurzer Schreibweise $G \Rightarrow G'$. Wenn G'' zu G' isomorph ist ($G'' \cong G'$), dann gilt durch Anwendung der Produktion p auf einen Knoten $v \in V(G)$ auch $G \Rightarrow_{(v,p)} G''$.

DEFINITION 4 (ABLEITBAR): Ein Graph G' ist aus einem Graph G ableitbar (in n Schritten), wenn es eine konkrete Ableitung (der Länge n) von G nach G'' gibt und $G' \cong G''$. Als Notation steht $G \Rightarrow^* G'$ für ableitbar und analog $G \Rightarrow^n G'$ für ableitbar in n Schritten.

DEFINITION 5 (SPRACHE EINER GRAPHGRAMMATIK): Für eine Graphgrammatik $GG = (N, T, P, S)$ wird das Axiom S identifiziert mit einem Knoten mit der Markierung S . Der Ausdruck $L(GG) = \{G | S \Rightarrow^* G, \text{ terminaler Graph über den Alphabeten } T_V, T_E\}$ bezeichnet die von GG erzeugte Menge von Graphen, die aus S abgeleitet werden kann. $L(GG)$ heißt auch die von GG erzeugte Sprache.

⁷ Die Notation $G_{|W}$ bezeichnet den induzierten Subgraphen von G , der entsteht, wenn die Menge $V(G)$ auf die Elemente von W beschränkt wird.

2.2.4 Beispiele

BEISPIEL 2: Abbildung 2.2 zeigt eine mögliche Ableitungsfolge für den vollständigen Graphen⁸ K_4 auf der Basis der Produktionen in Abbildung 2.1. Der Index am Implikationspfeil gibt an, welche der beiden Produktionen zur Anwendung kommt. Das in folgenden Schritten ersetzte Nichtterminalsymbol ist jeweils mit einem Kasten markiert.

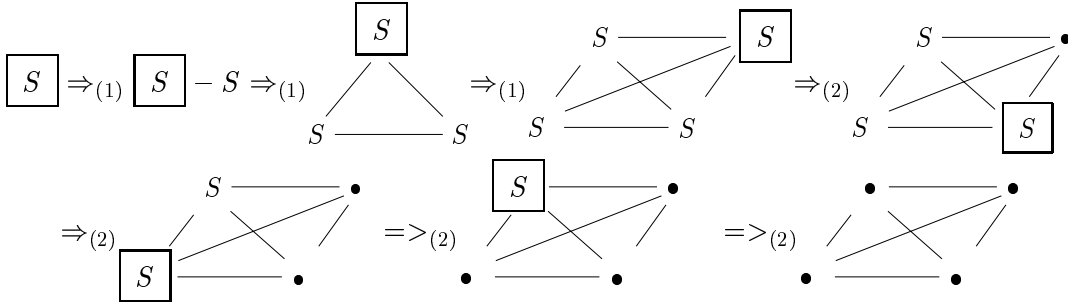
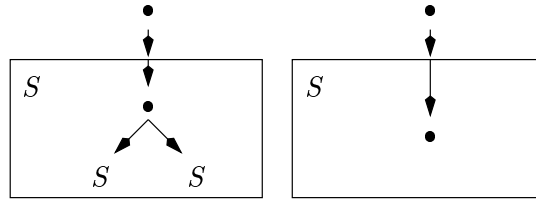
Abbildung 2.2: Ableitungsschritte für den K_4

Abbildung 2.3 zeigt zwei Produktionen einer Graphgrammatik für *extended binary trees*⁹, bei denen alle inneren Knoten verzweigen.

Abbildung 2.3: Produktionen für *extended binary trees*

2.2.5 Eigenschaften von Graphgrammatiken

Bei der Definition von Graphsprachen werden isomorphe Graphen als ‘gleich’ betrachtet.

DEFINITION 6 (ÄQUIVALENZ VON GRAPHGRAMMATIKEN): Zwei Graphgrammatiken GG_1 und GG_2 heißen äquivalent, wenn sie dieselbe Sprache erzeugen, also $L(GG_1) = L(GG_2)$. GG_1 und GG_2 erzeugen dann bis auf Isomorphie dieselben Graphen.

Unterschiedliche Typen von Graphgrammatiken werden klassifiziert. Für unsere Arbeit sind die Eigenschaften *konfluent* und *boundary* interessant, weil sie es ermöglichen, Aussagen über die Komplexität des Wortproblems zu formulieren:

⁸ K_n bezeichnet einen vollständigen Graphen über n Knoten. Jeder Knoten ist mit jedem anderen Knoten über eine Kante verbunden.

⁹ Bei *extended binary trees* besitzt jeder Knoten entweder den Ausgangsgrad 0 (Blatt) oder 2 (innerer Knoten). Siehe auch Knuth: [Knu73, S.399ff]

DEFINITION 7 (KONFLUENZ): Eine Graphgrammatik heißt *konfluent*, (C-edNCE), falls für jeden ableitbaren Graphen G und für jedes Paar von Produktionen $p = (u, R, C)$ und $p' = (u', R', C')$ mit $u, u' \in G, u \neq u'$ gilt: $G'' \cong G'''$ mit $G \Rightarrow_{u,p} G' \Rightarrow_{u',p'} G''$ und $G \Rightarrow_{u',p'} G'' \Rightarrow_{u,p} G'''$.

Eine Graphgrammatik ist dann konfluent¹⁰, wenn es auf die Reihenfolge der Anwendungen von unabhängigen Produktionen nicht ankommt. Produktionsanwendungen führen dann in beliebiger Reihenfolge zu demselben Ergebnis. Eine kontextfreie Grammatik ist konfluent in genau diesem Sinne, bei Graphgrammatiken ist dies nicht notwendig der Fall. Konfluenz ist eine Eigenschaft, die es erlaubt, eine Reihe von Begriffen und (Beweis)-Techniken von den gut erforschten kontextfreien Wortgrammatiken auf Graphgrammatiken zu übertragen. Um eine Graphgrammatik auf Konfluenz zu testen, ist es nötig, Nachbarschaftsbeziehungen zu betrachten. Kritische Fälle treten dann auf, wenn eine Produktion die Struktur nicht nur lokal durch Verbindungen zu neu eingeführten Knoten erweitert, sondern wenn zusätzlich Kanten zu bereits bestehenden Nachbarknoten gelöscht, hinzugefügt oder modifiziert werden. Das entsprechende Entscheidungsproblem ist in Polynomzeit entscheidbar.

Ergebnisse aus der Literatur zeigen, daß manche Graphsprachen - zum Beispiel quadratische Gittergraphen (Abbildung 2.4) - nicht mit Hilfe von konfluenten Graphgrammatiken beschrieben werden können. Auf entsprechende Ergebnisse und Anwendungen wird in Kapitel 5 eingegangen.

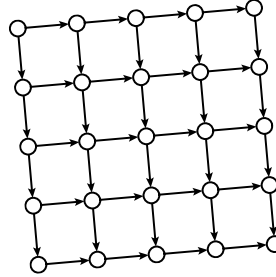


Abbildung 2.4: Ein 5×5 Gittergraph

DEFINITION 8 (BOUNDARY): Eine Graphgrammatik heißt *boundary*, (B-edNCE), falls die rechte Seite jeder Produktion keine adjazenten nichtterminalen Knoten enthält.

Jede boundary Graphgrammatik ist auch konfluent.¹¹ Falls die Sprache gradbeschränkt ist, gilt B-edNCE=C-edNCE.¹²

Allgemein gilt für das Wortproblem einer Graphgrammatik GG , daß es selbst dann **NP**-vollständig ist, falls GG boundary ist. Wenn eine Reihe von Zusatzbedingungen erfüllt sind, ergeben sich bessere Resultate.¹³ Für eine konfluente oder boundary Graphgram-

¹⁰ Eigenschaften von konfluenten Graphgrammatiken faßt Engelfriet in [Eng97, Kapitel 4] zusammen.

¹¹ [EL89] und [CER93]. Eigenschaften von boundary Graphgrammatiken untersucht [ELW88]

¹² [Bra91]

¹³ [Bra88a]

matik GG ist das Wortproblem dann in \mathbf{P} , wenn alle Graphen aus $L(GG)$ gradbeschränkt und zusammenhängend sind. Das Polynom kann allerdings sehr groß werden.¹⁴

2.3 Aufgabenstellung für diese Arbeit

Um vernünftige Ergebnisse zur Beschreibung syntaktischer Strukturen natürlicher Sprachen zu erhalten, ist deshalb eine Graphgrammatik GG gesucht, welche die folgenden formalen Eigenschaften besitzt:

1. GG ist konfluent oder boundary
2. $L(GG)$ ist gradbeschränkt
3. $L(GG)$ ist zusammenhängend.

Die Arbeit wird zeigen, daß eine Graphgrammatik mit diesen Eigenschaften existiert: Syntaktische Strukturen lassen sich mit Hilfe von Produktionen beschreiben und das Wortproblem effizient lösen.

¹⁴ Die beeinflussenden Parameter werden in [RW86] untersucht.

In transformational linguistics (more perhaps than in older, more established sciences), a familiarity with the anecdotes and personalities is extremely useful in understanding what has been written. Papers have at times been organized not to put forth a particular idea but to attack the complex of beliefs held by some rival. Arguments that appear mixed and chaotic can often be decoded by the realization ‘Oh, he’s attacking X.’ Students planning to study transformational grammar in depth should learn the history and politics of the field.

WINOGRAD IN [WIN83, S. 557]

3 Sprache als Forschungsgegenstand der Linguistik

Das folgende Kapitel erläutert die für diese Arbeit relevante linguistische Terminologie und Methodik. Die Auswahl orientiert sich an dem Ziel, einen formalen Zugang zum Begriff *natürliche Sprache* zu entwickeln. Entsprechend erfolgt die Beschreibung aus der Perspektive der Informatik und beschäftigt sich schwerpunktmäßig damit, wie die linguistischen Methoden aus der Sicht der Theorie der formalen Sprachen zu bewerten sind.

3.1 Eine erste Abgrenzung

Unter einer natürlichen Sprache verstehen wir in einer ersten Näherung *eine Sprache, die ein Mensch als Muttersprache erlernen kann*. Gemessen an formalen Ansprüchen erscheint diese Charakterisierung zunächst unbrauchbar aus mindestens vier Gründen, mit denen wir uns auseinandersetzen müssen:

1. **Unklare Terminologie:** Schwer wiegt der an die Definition für natürliche Sprache zu richtende Vorwurf, daß das Konzept *Muttersprache* formal nicht faßbar ist und sich Begriffe wie *Mensch* und *erlernen* der mathematischen Beschreibbarkeit entziehen. Auch bezieht die Formulierung ausgestorbene oder noch nicht existierende Sprachen ausdrücklich mit in die Definition ein, was dahingehend problematisch ist, als daß der empirische Zugang zu diesen Sprachen zusätzlich erschwert oder aus logischen Gründen prinzipiell ausgeschlossen ist. Von Gemeinsamkeiten ausge-

storbener und zukünftiger Sprachen zu sprechen, ist nur dann möglich, wenn zeit-invariante Aspekte bei natürlichen Sprachen unterstellt werden. Damit sind solche Regularitäten gemeint, die sowohl in historischen als auch entwicklungsgeschichtlichen Dimensionen hinweg keine Veränderung erfahren haben. Tatsächlich liefert die Linguistik Argumente dafür, daß es solche Aspekte geben muß.¹

2. **Äußeres Erscheinungsbild:** Es gibt sehr viele Sprachen auf der Welt², die sich darin deutlich unterscheiden, daß völlig unterschiedliche Alphabete und Schreibweisen zur Anwendung kommen. Dieser Umstand bringt zwar Schwierigkeiten bei der Suche nach einer gemeinsamen Beschreibung mit sich, stellt aber keineswegs ein grundsätzliches Hindernis dar. Selbstverständlich folgt aus der Verwendung unterschiedlicher Symbole nicht, daß ein unterschiedlicher struktureller Aufbau vorliegt. Für unsere Zwecke reicht es aus, unter einem *universellen Alphabet* Σ die Menge aller Symbole zu verstehen, die in irgendeiner Schriftsprache Verwendung finden.³ Diese Definition ist ungenau, da nicht festgelegt ist, wieviele Symbole Σ enthält. Stattdessen unterstellen wir, daß jedes benötigte Zeichen in Σ vorhanden ist. Dabei reicht es aus, Σ als *endlich* anzunehmen.⁴
3. **Inkonsistenzen und Ausnahmen:** Vielfältige grammatische Regularitäten lassen Sprachen oft als komplexe und chaotische Phänomene erscheinen, die jeweils ganz eigene Gesetzmäßigkeiten mit sich bringen. Entsprechend werden die Unterschiede zwischen Sprachen in der Regel deutlicher wahrgenommen, als die Gemeinsamkeiten. Unregelmäßigkeiten im Regelapparat einer Sprache verleiten leicht zu dem Fehlschluß, daß sich zwischen verschiedenen Sprachen prinzipiell kein tiefergehender Zusammenhang feststellen läßt. Unvoreingenommen betrachtet schließt aber auch diese Beobachtung nicht aus, daß sich hinter der Vielfalt an Phänomenen nicht doch eine gemeinsame – möglicherweise sehr abstrakte – Systematik verbirgt.
4. **Individualsprachen:** Tatsächlich ist es im Einzelfall kaum nachzuweisen, daß zwei Menschen genau dieselbe Muttersprache besitzen. Der Umstand, daß sie in der Lage sind, miteinander zu kommunizieren, stellt keinen Beleg für eine Übereinstimmung in jeder Hinsicht dar. Wenn aber jeder Mensch eine ganz persönliche ihm eigene (Individual)-Sprache spricht, dann verliert der Terminus *natürliche Sprache* als Klassifikationsbegriff seinen Sinn. Eine formalisierter Sprachbegriff muß eine Strategie beinhalten, mit dieser unvermeidbaren Unschärfe umzugehen.

¹ In Abschnitt 3.6 werden die entsprechenden Gedankengänge erläutert. Unsere Charakterisierung zielt darauf ab, Sprache als einen Zustand anzusehen, und nicht als einen Prozeß. Diese Sicht auf Sprache wird in der Linguistik auch als *synchron* bezeichnet und von einer *diachronen* Herangehensweise unterschieden, welche die Entwicklung von Sprachen über mehr oder weniger große Zeiträume hinweg betrachtet.

² Derzeit existieren ungefähr 6000 Sprachen auf der Welt. Von diesen werden im nächsten Jahrhundert voraussichtlich die Hälfte aussterben, wenn sich die zu beobachtende Entwicklung in der Zukunft weiter fortsetzt. In der Vergangenheit hat es weitaus mehr Sprachen gegeben als heute, von denen oft aber nur noch rudimentäre Zeugnisse vorliegen. Siehe [MS97]

³ Warum die Beschränkung auf Schriftsprache sinnvoll ist, wird nachfolgend erläutert.

⁴ Eine gute Approximation von Σ stellt die UNICODE Konvention dar. Diese als Erweiterung des ASCII-Codes gedachte Codetabelle beansprucht, „genügend“ Symbole für die Sprachen der Welt zur Verfügung zu stellen.

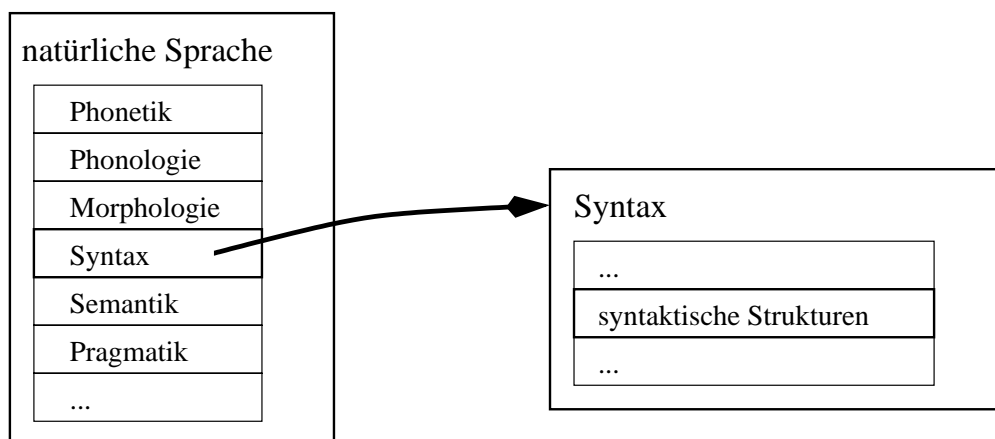


Abbildung 3.1: Linguistische Disziplinen betrachten verschiedene Aspekte von Sprache

Die Linguistik hat Antworten formuliert, um die vier oben skizzierten Kritikpunkte handhaben zu können. Vor einer Klärung ist es nötig, eine Systematik in das reichhaltig vorliegende Datenmaterial zu bringen, mit dem der Linguist sich konfrontiert sieht. Dies kann unter verschiedenen Gesichtspunkten geschehen. Die unterschiedlichen Herangehensweisen spiegeln sich in einer großen Zahl an Disziplinen wieder, welche die Linguistik unterscheidet. Zu nennen sind neben anderen *Syntax*, *Semantik*, *Phonologie*, *Phonetik*, *Morphologie* oder *Pragmatik*, zu denen wiederum zahlreiche Unterdisziplinen hinzukommen. Diese differenzierte Herangehensweise ermöglicht es, Gemeinsamkeiten zwischen verschiedenen Sprachen aufzuzeigen, die bei einer ganzheitlichen Betrachtung verborgen bleiben. Wir beschränken uns in dieser Arbeit auf ausgewählte Aspekte der *Syntax* und untersuchen die Möglichkeiten, Ergebnisse aus der linguistischen Forschung bezüglich syntaktischer Strukturen mit Methoden aus der Theorie der formalen Sprachen zu beschreiben (Abbildung 3.1). Die folgende Definition macht deutlich, daß es dabei ausschließlich um Schriftsprache geht:

DEFINITION 9 (NL , NATÜRLICHE SPRACHE): Eine natürliche Sprache $NL \subseteq \Sigma^*$ umfaßt die Menge aller wohlgeformten niedergeschriebenen Sätze, die in einer natürlichen Sprache vorkommen dürfen. Um mehrere natürliche Sprachen voneinander zu unterscheiden, schreiben wir NL_1 , NL_2 , usw. Die Gesamtheit aller natürlichen Sprachen $\bigcup NL_i$ wird mit \mathcal{NL} notiert. Es erleichtert die Formalisierung, wenn wir die Disjunktheit von unterschiedlichen Sprachen unterstellen: $NL_i \cap NL_j = \emptyset, i \neq j$.⁵

In der Praxis ist es nicht möglich, alle sprachlichen Phänomene erschöpfend auf Schriftzeichen abzubilden. Die Restriktion auf Schriftsprache stellt also eine Vereinfachung dar.⁶ Ein vollständiger und umfassender Sprachbegriff müßte zusätzlich die ganze Bandbreite

⁵ Diese Annahme stellt eine Vereinfachung dar. Tatsächlich gibt es Fälle der Übereinstimmung. So gibt es beispielsweise im Finnischen und Ungarischen gleichlautende Sätze mit der Bedeutung „Der Zug kommt.“. Allerdings entspricht der finnische Ausdruck für „Der Zug“ dem ungarischen Ausdruck für „kommt“ und umgekehrt. [MS95]

⁶ Allerdings ist diese Restriktion nicht ganz so scharf, wie es zunächst scheinen mag: Auch für gesprochene Sprache existieren Notationen wie das IPA (*international phonetic alphabet*), die eine Codierung als Lautschrift ermöglichen.

nonverbaler Kommunikation mit umfassen. Solche Ansätze spielen in dieser Arbeit keine Rolle.

In der Definition für NL ist außerdem der Begriff „wohlgeformt“ problematisch. Für eine formale Betrachtung ist dieser Zugang ungenügend, denn er liefert kein nachprüfbares Kriterium dafür, ob ein beliebiges Element aus Σ^* der Menge NL angehört oder nicht. Die Menge NL ist damit nur vage beschrieben. Für eine Formalisierung syntaktischer Regularitäten müssen wir deshalb selber eine Lösung erarbeiten. Vor der Entwicklung eines solchen formalen Wohlgeformtheitsbegriffs wollen wir untersuchen, welche Rolle er in der Linguistik spielt. Dabei hilft eine Betrachtung der der Syntax zugrundeliegenden linguistischen Terminologie.

3.2 Syntax als eigenständige linguistische Disziplin

3.2.1 Schriftsprache besteht aus Sätzen

Der Satzbegriff wird in der Literatur uneinheitlich verwendet.⁷ Als erste Arbeitshypothese erscheint es naheliegend, einen Satz S als unstrukturierte *Sequenz von Symbolen* ($S \in \Sigma^*$) anzusehen, welche von Satzzeichen wie $\{„“, „!“, „?“, \dots\}$ begrenzt werden. Die Symbole stehen dann unstrukturiert gleichberechtigt nebeneinander wie in „J-O-H-N- \sqcup -L-O-V-E-S- \sqcup -M-A-R-Y“. Bei dieser Charakterisierung gilt die Beziehung $|NL| \ll |\Sigma^*|$. Nur eine sehr kleine Teilmenge aller Buchstabenkombinationen repräsentiert einen vernünftigen Satz. Mit einer zusätzlichen einschränkenden Definition werden wir der Beobachtung gerecht, daß die Kombinationen nicht zufällig sind, sondern der Satz sich vielmehr aus einem Vorrat vordefinierter Zeichenketten bedient, nämlich den Wörtern einer Sprache. Auf der Basis eines Lexikons mit $\text{LEXIKON} \subset \Sigma^*$ gilt für einen Satz die Beziehung $S \in \text{LEXIKON}^*$, der Satz ist dann eine *Sequenz von Zeichenketten* wie in „John-loves-Mary.“ Auch diese Charakterisierung ist offensichtlich noch zu grob wegen $|NL| \ll |\text{LEXIKON}^*|$. In keiner Sprache dürfen wir die Bestandteile eines Satzes beliebig permutieren. Weitere Restriktionen lassen sich formulieren, wenn die Wörter als typisiert angenommen werden. Der Satz ist dann eine *Sequenz von typisierten Zeichenketten* wie in „John_{Nomen}-loves_{Verb}-Mary_{Nomen}“. Das Lexikon übernimmt dann die Rolle eines Alphabets und der linguistische Begriff vom *Wort* fällt mit dem Begriff *Symbol eines Alphabets* im Sinne der formalen Sprachen zusammen.⁸ Diese Ansätze berücksichtigen noch nicht, daß es sprachspezifische Gesetzmäßigkeiten und Beschränkungen gibt, die zulässige Zeichenketten in einer Sprache erfüllen müssen. Die dabei relevanten Regula-

⁷ „Den Begriff »Satz« zu definieren, ist problematisch; es gibt eine ganze Anzahl von Satzdefinitionen, die in verschiedener Weise auf inhaltlicher (logischer, philosophischer, kommunikationswissenschaftlicher, psychologischer) oder auf rein rhythmischer Grundlage formuliert wurden und sich kaum miteinander in Einklang bringen lassen.“ [Pel84, S. 135]

⁸ Wörter werden dann behandelt wie unveränderliche Symbole. Aus der Sicht der Syntax ist dieses Vorgehen zulässig. Will man aber eine umfassendere Beschreibung von Sprache entwerfen, muß man zusätzliche Regularitäten beschreiben: „Als strukturalistischer Klassifikationsbegriff jedoch ist das Wort nicht brauchbar: Hierfür ist der Wortbegriff, über den in der Sprachwissenschaft alles andere als Konsens herrscht, zu heterogen. Ist z.B. dt. »Waschmaschine« ein Wort, sind frz. »Machine à laver« drei Wörter, engl. »Washing-machine« eines oder zwei? [...] Die Einheit des Wortes ist problematisch.“ [Pel84, S. 115]

ritäten werden in der *Morphologie* untersucht, welche sich mit der Gestalt von Wörtern beschäftigt und den Einheiten, aus denen sie sich zusammensetzen⁹, den sogenannten *Morphemen*.¹⁰ Aus syntaktischer Sicht macht es keinen wesentlichen Unterschied, ob wir unter Σ ein Alphabet, eine Menge von Morphemen oder Wörtern verstehen, solange wir diese Komponente als endlich betrachten.¹¹ Dies ist sinnvoll, weil der innere Aufbau von Wörtern syntaktische Regularitäten nur am Rande berührt. Die Formalisierung solcher morphologischen Regularitäten stellt eine eigene Aufgabenstellung dar.¹² Die tieferen Einsichten in die inneren Regularitäten eines Satzes, die uns in dieser Arbeit interessieren, eröffnen sich erst dann, wenn man sich von der sequentiellen Sichtweise löst.

3.2.2 Sätze besitzen eine Struktur

Betrachtungen der wechselseitigen Abhängigkeiten zwischen den Bestandteilen eines Satzes führen in der Linguistik zu dem Ergebnis, daß jedem grammatischen Satz, der dem Betrachter als sequentielle Wortfolge erscheint, eine *hierarchische Struktur* zugesprochen werden kann, wie in Abbildung 3.2 in einem einfachen Beispiel dargestellt.¹³ Die Syntax beschäftigt sich mit der Beschreibung und Analyse solcher Strukturen. Die Ergebnisse dieser Betrachtungen ermöglichen es, Kriterien zu finden, um ungrammatische Sätze von grammatischen zu unterscheiden.

Die strukturelle Sichtweise ist nützlich, um zum Beispiel das Zustandekommen von Mehrdeutigkeiten zu erklären wie in „*Der Mann sieht das Kind mit dem Fernrohr*“. Beide möglichen Lesarten sind durch verschiedene zugrundeliegende syntaktische Strukturen erklärbar, wie Abbildung 3.3 illustriert.

Davon zu unterscheiden sind Mehrdeutigkeiten, wie sie bei einem Wort wie „*Schloß*“ oder „*Bank*“ auftreten. Diese zu untersuchen, ist Aufgabe der *Semantik*.

⁹ „Offensichtlich ist es angebracht, den morphologischen Kategorien ein gewisses »Eigenleben« zuzugestehen. M. a. W., morphologische Regularitäten bleiben in den Regeln des Strukturaufbaus zunächst unberücksichtigt ...“ [GHS87, S. 188/189] Soweit es sich um morphologische Erscheinungen handelt, können diese also ignoriert werden, obwohl diese die Grundbausteine für Satzkonstruktionen darstellen: „Das Funktionieren des Sprachsystems wird vorgestellt als ein Kombinieren von kleineren Einheiten zu nächstgrößeren: Aus 25-30 Phonemen wird eine potentiell unbegrenzte Menge von Morphemen zusammengefügt. Die Möglichkeit der Sprache, in dieser Weise »aus wenig viel zu machen«, macht nicht halt auf der Morphemebene; jedem ist geläufig, daß Sprache in ihrer alltäglichen Verwendung in größere Einheiten realisiert auftritt, als die Morpheme es sind. Diese größeren Einheiten sind das, was wir als Sätze bezeichnen. Sie bilden den Untersuchungsgegenstand der Syntax.“ [Mar60, S. 311]

¹⁰ Unterschieden werden dabei „solche Morpheme, die selbständig als Wort auftreten können (= freie Morpheme), und solche, die nur als Teil eines Wortes auftreten können (=gebundene Morpheme)“ [Pel84, S. 116]. Offen bleibt dann, ob unter „Wort“ auch flektierte Formen zu verstehen sind, oder nur unflektierte Grundformen, wie sie als Einträge im Lexikon zu finden sind. Anders gesagt: Ist es sinnvoll, zu sagen, daß die Zeichenketten „Häuser“, „gelaufen“, „schönsten“ auch Wörter sind, oder gilt dies nur für die Grundformen „Haus“, „laufen“, „schön“? Wir vertreten in dieser Arbeit die letzte Position.

¹¹ Die Eigenschaft der Endlichkeit eines Alphabets stellt bei formalen Sprachen eine übliche Voraussetzung dar. Für Wörter gilt das in der Regel nicht - man betrachte nur, wieviele Bezeichnungen es für Zahlen gibt. Wenn wir also in dieser Arbeit stets ein endliches Lexikon zugrundelegen, so stellt dies eine Vereinfachung dar.

¹² [Spr92]

¹³ Die Bedeutung der internen Knotenbezeichner wie *NP* und *VP* wird später erläutert werden.

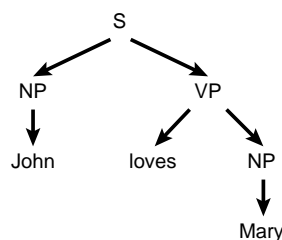


Abbildung 3.2: Ein einfacher Syntaxbaum für „John loves Mary“

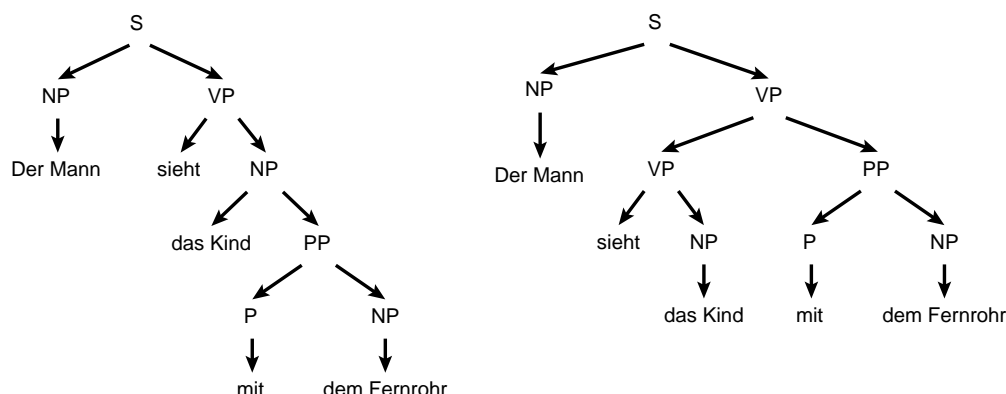


Abbildung 3.3: Manchmal sind mehrere Satzanalysen möglich

Auf der Basis der Begriffe Alphabet/Wort ist die adäquate Beschreibung des Strukturaufbaus von Sätzen der natürlichen Sprache der zentrale Gegenstand der Syntax. Die Reichweite des Syntaxbegriffes wird dabei unterschiedlich gehandhabt. Hier behandeln wir ausschließlich *satzimmanente Phänomene* und ignorieren satzübergreifende Zusammenhänge. Da Sätze beliebig lang sein können, sind auch innerhalb eines Satzes Strukturbedingungen zwischen Wörtern nicht notwendig beschränkt.

Da uns vor allem die Beschreibung von syntaktischen Strukturen aus der Sicht der formalen Sprachen interessiert, können die Wörter als Blätter der Struktur gleichsam als unveränderliche atomare Objekte betrachtet werden, die einen Typ besitzen.¹⁴ Von der inneren Struktur eines Wortes wird bei uns abstrahiert.

¹⁴ Wörter sind hier einfach Zeichenketten, die eine bestimmte Funktionalität im Satzkontext besitzen. Diese Funktionalitäten sind für unsere Zwecke mit den Typbezeichnungen *Nomen*, *Verb*, *Adjektiv*, etc. ausreichend beschrieben, obwohl auch dies eine Vereinfachung darstellt: „Die klassische Einteilung der Wortkategorien in Substantive, Verben, Adjektive, Artikel, Pronomen, Numeralien, Adverbien, Präpositionen, Konjunktionen und Interjektionen, wie sie in der traditionellen Grammatik verwendet werden, eignen sich nicht für den exakten Aufbau einer Grammatik. Die Begriffe sind zu vage definiert, es fehlen einheitliche Gesichtspunkte, die Einteilung ist nicht disjunkt.“ [Kut74, S. 211] oder auch: Es liegt die Situation vor, „... daß in der traditionellen Grammatik sowohl Bezeichnung als auch Einteilung der Wortarten mehr oder weniger intuitiv und unsystematisch vollzogen wurden. [...] Die vorstrukturalistische Einteilung der Wortarten (auch Wortklassen genannt) beruhte auf willkürlichen, uneinheitlichen Kriterien und war daher undurchsichtig und nicht intersubjektiv nachprüfbar.“ [Pel84, S. 126]

3.2.3 Unterschiedliche Verwendungsweisen des Begriffs *Grammatik*

Die Begriffe *Grammatik* und *Syntax* werden in der Literatur uneinheitlich verwendet. Während in der Theorie der formalen Sprachen der Grammatikbegriff ein wohldefinierter endlich beschriebenes Regelsystem zur Generierung von Mengen beschreibt, ist die Abgrenzung dieses Begriffs in der Linguistik nicht so klar. Oft wird er dort umfassender verwendet, beschreibt dann ein sehr umfangreiches System, welches alle Aspekte von natürlichen Sprachen zu modellieren beansprucht.¹⁵

In der Linguistik wird unter Grammatik in einem eingeschränkten Sinn auch ein Regelsystem verstanden, welches in der Lage ist, diejenigen Gesetzmäßigkeiten von Sprache zu beschreiben, mit denen sich die Syntaxtheorie beschäftigt. So gesehen darf eine grammatische Theorie dann nicht als universelles Modell zur Erklärung aller sprachlichen Erscheinungen verstanden werden. Sie kann allerdings sehr wohl Teil eines solchen umfassenden Modells sein.

In dieser Arbeit wird der Begriff *Grammatik* im Sinne der formalen Sprachen verwendet, wie sie zum Beispiel bei Wortgrammatiken zur Anwendung kommen. Dann beschränkt sich die Aufgabenstellung darauf, eine Grammatik G für eine bestimmte Sprache zu finden mit der Eigenschaft $NL = L(G)$. Entsprechend ist ein Grammatikformalismus gesucht, der die Klasse \mathcal{NL} möglichst gut charakterisiert. Dies leitet über zum nächsten Abschnitt: Wie ist die Menge \mathcal{NL} aus der Sicht der Chomsky Hierarchie zu klassifizieren?

3.3 Natürliche Sprachen und die Chomsky Hierarchie

Daß natürliche Sprachen nicht regulär sind, hat Chomsky bereits 1957 gezeigt.¹⁶ Die Frage, ob \mathcal{NL} kontextfrei ist, wurde in der Folge in der Literatur viel diskutiert. Zahlreiche Argumente wurden für die Position vorgetragen, daß kontextfreie Grammatiken nicht mächtig genug sind, um alle syntaktischen Phänomene zu beschreiben, die in natürlichen Sprachen eine Rolle spielen. In jedem Fall lassen sich aber mit Hilfe von kontextfreien Grammatiken viele Regularitäten beschreiben.¹⁷ Der für die Entdeckung kontextsensiti-

¹⁵ „Die Aufgabe einer Grammatik ist es [...], die Regelmäßigkeiten des Sprachgebrauchs durch ein System von Regeln explizit zu machen.“ [Kut74, S. 206]

¹⁶ [Cho57]

¹⁷ Zur Modellierung syntaktischer Phänomene wurden von Chomsky bereits früh Phrasenstrukturgrammatiken (PSG) vorgeschlagen ([Cho55], [Cho57], [Cho65]), bei denen es sich letztlich um kontextfreie Grammatiken handelt. Phrasenstrukturgrammatiken sind in dem Sinne einfach zu handhaben, als daß kontextfreie Mechanismen gut erforscht und deren Möglichkeiten und Grenzen umfassend dokumentiert sind. Zur automatischen Generierung von Parsern zu einer vorgegebenen CFG stehen Softwarewerkzeuge zur Verfügung, welche gemäß einer vorgegebenen Spezifikation den Quellcode für einen entsprechenden Parser generieren [MB91]. Zur Lösung des Wortproblems sind $O(n^3)$ -Algorithmen bekannt. Für linguistische Anwendungen ist diese Komplexität gut genug, weil die Instanzen von n in der Regel klein sind, denn typischerweise sind Sätze aus \mathcal{NL} nicht sehr lang. Dafür hat die Größe einer Grammatik einen entscheidenden Einfluß auf die Laufzeit. In ungünstigen Fällen kann der CYK-Algorithmus für kontextfreie Sprachen exponentiell viele Symbole in den Tabellenfeldern generieren, was sich abhängig von der Implementation negativ auf die Laufzeit auswirken kann. Dazu Tomita in [Tom86]: „First the length of a sentence (the number of words) is usually between 10 and 20, and seldom exceeds 30. Second, the number of rules in a natural language gets large if one tries to cover the language fairly comprehensively. No accurate estimation has been made on how many context-free

ver Phänomene betriebene Aufwand und die im Vergleich dazu wenigen überzeugenden Daten machen aber deutlich, daß solche Regularitäten nicht häufig auftauchen.

Pullum und Gazdar liefern in [PG87] eine umfassende kritische Übersicht zu häufig zitierten Arbeiten, die sich mit diesem Thema beschäftigen.¹⁸ Sie kommen zu dem Schluß, daß *keiner* der vorgebrachten Beweise einer kritischen Betrachtung standhält, die Frage nach der Kontextfreiheit von \mathcal{NL} also weiterhin offen ist. Erst neuere Arbeiten liefern überzeugende Argumente dafür, daß \mathcal{NL} nicht kontextfrei sein kann:

3.3.1 \mathcal{NL} ist nicht kontextfrei

Kontextfreie Sprachen sind abgeschlossen unter Durchschnitt mit regulären Mengen. Unter Anwendung dieser Eigenschaft zeigt Shieber in [Shi88], daß die schweizerdeutsche Sprache nicht kontextfrei ist: Zunächst wird L definiert als Menge der grammatischen Sätze der schweizerdeutschen Sprache. Mit Hilfe einer geeigneten regulären Menge R setzt Shieber dann $L' := L \cap R$. Da L' sich als nicht kontextfrei erweist, gilt selbiges auch für L und daraus folgt die Behauptung. Da - wie in Abschnitt 3.5 noch ausführlicher erläutert wird - nach einem universellen Formalismus gesucht wird, der in der Lage ist, *jede* natürliche Sprache zu beschreiben, wird in der Linguistik infolge dieser Klassifizierung des Schweizerdeutschen auch \mathcal{NL} als nicht kontextfrei angesehen.

Von Interesse sind im Schweizerdeutschen insbesondere Sätze der folgenden Form (mit deutscher Wort-für-Wort Transkription):

„Jan säit das mer₁ d'chind₂ em Hans₃ es huus haend wele₁ laa hälfe₂ aastriche₃.“

Jan sagte, daß wir die Kinder dem Hans das Haus wollten lassen helfen anstreichen.

Die Indizes geben an, welche Paare an Substantiven und Verben zusammengehören. Dem Beispielsatz entspricht von der Bedeutung her der Satz mit der für die deutsche Sprache typische Wortstellung:

„Jan sagte, daß wir₁ die Kinder₂ dem Hans₃ das Haus anstreichen₃ helfen₂ lassen wollten₁.“

Die Kasusmarkierungen der Substantive im Schweizerdeutschen machen eine Art von Querbeziehung bezüglich der dazugehörenden Verben deutlich, die als „*kreuzweise Abhängigkeit*“ (*cross-serial dependency*) bezeichnet wird. Für ein Wort der Länge n stimmt ein Zeichen an der Position $i < n/2$ mit dem Zeichen an der Position $i + n/2$ überein.¹⁹

rules are needed to cover English almost completely. We know, however, that it is much larger than for programming language grammars. Therefore, a natural language parser must be efficient with respect not only to the sentence length, but also to the number of rules in its grammar.“

¹⁸ Weitere Argumente finden sich in [Kin83].

¹⁹ Solche kreuzweisen Querbeziehungen können bei unbeschränktem n nicht mit kontextfreien Grammatiken beschrieben werden. Dieser Umstand ist in der Literatur mehrfach als Argument: „ L enthält Phänomen mit kreuzweiser Abhängigkeit $\Rightarrow L$ nicht kontextfrei“ mißbraucht worden. Dieser Ansatz funktioniert natürlich nicht: Wenn eine Menge eine nicht kontextfreie Teilmenge enthält, kann sie trotzdem noch kontextfrei sein. Die Sprache $\{x|x \in \{a,b\}^*\}$ enthält beispielsweise die Copy-Language ww , ist aber regulär (und damit auch kontextfrei).

Dies ist bezüglich der zugrundeliegenden Struktur eine grundsätzlich andere Konstruktion, als eine „verschachtelte Abhängigkeit“ (*nested dependency*), welche charakteristisch für kontextfreie Sprachen ist (korrekt geklammerte Terme). Nach diesem Schema lassen sich im Schweizerdeutschen unendlich lange Sätze mit unbeschränkt hoher Verschachtelungstiefe konstruieren.²⁰

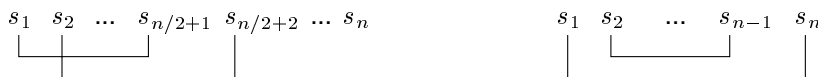


Abbildung 3.4: kreuzweise und verschachtelte Abhängigkeit

In der Syntax sucht man nach formalsprachlichen Beschreibungsmitteln, die nicht nur die korrekte Abfolge der Terminalsymbole zu generieren in der Lage sind (*schwache Äquivalenz*), sondern mit Hilfe der Ableitungsbäume auch die ihnen zugrundeliegende Struktur (*starke Äquivalenz*). Da ein in Sinne von $a_1 a_2 \dots a_n b_1 b_2 \dots b_n$ verschachtelter Terminalstring von einem String mit Verschachtelung $a_1 a_2 \dots a_n b_n b_{n-1} \dots b_1$ allein von der Oberfläche her nicht unterschieden werden kann,²¹ ist empirisch nicht immer eindeutig, daß eine kreuzweise Relation tatsächlich vorliegt. Im Schweizerdeutschen ist dies aber der Fall, die kreuzweisen Querbeziehungen kommen durch die Morphologie zum Ausdruck.

Auf dieser Grundlage gründet Shieber seinen Beweis. Die reguläre Menge R für die Schnittbildung definiert er als

$$R = \text{„Jan säit das mer (d'chind)* (em Hans)* es huus haend wele (laa)* (hülfe)* aastriche“}$$

Als Schnittmenge $L' := L \cap R$ ergibt sich dann:

$$L' = \text{„Jan säit das mer (d'chind)^n (em Hans)^m es huus haend wele (laa)^n (hülfe)^m aastriche“}$$

Der entsprechende Satz läßt sich im Deutschen sinngemäß umschreiben als folgender Bandwurmsatz:

„Jan sagte, daß wir wollten, daß die Kinder zulassen, . . . , daß die Kinder zulassen, dem Hans zu helfen, dem Hans zu helfen, . . . , dem Hans zu helfen, das Haus anzustreichen.“

Daß der Satz unter semantischen Gesichtspunkten höchst fragwürdig ist, schwächt Shiebers Argument nicht. Unter syntaktischen Gesichtspunkten ist der Satz korrekt konstruiert. Die so erzeugte Teilsprache L' ist von der Form $xa^n b^m y c^n d^m z$ und damit nicht kontextfrei.²²

²⁰ Daß sich dieses Beispiel tatsächlich so fortsetzen läßt, ist wesentlich. Sonst wäre die Konstruktion mit einem regulären Ausdruck beschreibbar. Shieber dazu: „One could argue that the phenomenon of cross-serial clause structure is bounded by, say, five embeddings or, to be more generous, one hundred. In either case, the language with bounded cross-seriality would be context-free, regardless of case-marking properties. Down this path lies tyranny. Acceptance of this argument opens the way to proofs of natural languages as regular, say finite. The linguist proposing this counterargument to salvage the context-freeness of natural language may have won the battle, but has certainly lost the war.“ [Shi88, S. 329]

²¹ Die Indizes in den beiden Strings sind kein Bestandteil der Terminalsymbolmenge. Sie geben als Metasybole lediglich die Querbeziehungen an, die in einer geeigneten Grammatik auch in der Ableitungsstruktur zum Ausdruck kommen sollen.

²² Dies ergibt sich leicht, wenn man die Sprache unter Anwendung eines Homomorphismus weiter zu

3.3.2 Kontextsensitive Sprachen sind zu stark für \mathcal{NL}

Auf der Grundlage der Chomsky Hierarchie kommen als weitere Kandidaten zur Beschreibung von \mathcal{NL} nur noch Grammatiken vom Typ-0 oder Typ-1 in Frage.²³ Savitch erklärt beide Klassen als ungeeignet für die Charakterisierung von \mathcal{NL} , weil sie viel zu mächtig sind.

Bei unbeschränkten Typ-0-Grammatiken scheint dies offensichtlich, bei Typ-1 ist eine genauere Betrachtung nötig: Savitch führt an, daß rekursiv aufzählbare Sprachen mit Hilfe eines *Padding* genannten Mechanismus gleichsam in kontextsensitiven Sprachen enthalten sind. Padding bezeichnet die Möglichkeit, ein zusätzliches Alphabetsymbol beliebig oft wiederholt an einen Terminalstring hintenanzustellen. Es gilt die Beziehung: $L \in RE \Rightarrow \exists L' \subseteq \{xd^t | x \in L, d \notin \Sigma\}$, L ist CSL . Mit kontextsensitiven Sprachen kann also beinahe jede Sprache approximiert werden. In diesem Sinne sind kontextsensitive Sprachen gegenüber den Typ-0-Sprachen strukturell kaum eingeschränkt.²⁴

Aus dieser Perspektive erscheinen kontextsensitive Sprachen als mächtig genug zur Beschreibung von \mathcal{NL} . Das allein ist jedoch nicht zufriedenstellend, wie Savitch betont: „*Such models are certainly adequate to describe any natural language string set. However [...] they do not tell us anything about which string sets are possible human languages. They do not tell us anything that can distinguish natural language syntax from any other classification task carried out by humans or other animals or machines. [...] All models that are this powerful witness no properties of human language other than the fact (or assumption) that it lends to some sort of algorithmic analysis. Hence, it makes more sense to look for weaker models that more exactly capture the string sets produced by real and potential human languages.*“²⁵ Tatsächlich lassen sich mit Hilfe von kontextsensitiven Grammatiken Regularitäten modellieren, die in keiner bekannten natürlichen Sprache eine Entsprechung finden. Dies gilt für Phänomene wie zum Beispiel Generierung von Spiegelungen auf der Satzebene als syntaktischer Prozeß, die unbeschränkte Vervielfachung von Wörtern (w^n) oder vielfältige arithmetische Operationen.²⁶ Alle diese Möglichkeiten kommen in \mathcal{NL} nicht zur Anwendung, was in der Literatur als Indiz verstanden wird, daß im Sinne von $\mathcal{NL} \subset CSL$ eine echte Teilmengenbeziehung vorliegt.

$a^n b^m c^n d^m$ reduziert. Von dieser Sprache ist bekannt, daß sie nicht kontextfrei sind. Da außerdem kontextfreie Sprachen abgeschlossen sind unter Homomorphismen, gilt das behauptete. Siehe auch [HU79]

²³ Produktionen einer Typ-0-Grammatik unterliegen keiner Beschränkung. Typ-1-Grammatiken generieren exakt die Sprachen, die von nichtdeterministischen Turing-Maschinen mit einer linearen Beschränkung auf den Speicherplatz ($NSPACE(n)$) erkannt werden. Grammatiken dieses Typs werden auch als *kontextsensitiv* bezeichnet. Die Produktionen einer kontextsensitiven Grammatik sind von der Form $\alpha A \beta \rightarrow \alpha w \beta$ mit: α, β, γ beliebige Strings, A ein Nichtterminalsymbol. In [Sav87] kritisiert Savitch den Begriff „kontextsensitiv“ und schlägt statt dessen den Terminus „nonerasing“ vor.

²⁴ Savitch: „*Perhaps a more telling reason to reject the class of context-sensitive grammars as a model for natural languages is that the languages that they generate are in a sense just as structurally complex as those of the unrestricted (type 0) grammars. They do not generate all the recursively enumerable languages, but for each recursively enumerable language, they do generate a related language which intuitively is just as complex.*“ [Sav87, S. 361]

²⁵ [Sav87, S. 359]

²⁶ Mit unbeschränkten Grammatiken können diverse arithmetische Operationen wie $a + b$, $a * b$, Primzahltest, Umwandlung Unär \rightarrow Binärdarstellung von Zahlen beschrieben werden.

Auch in anderen Zusammenhängen scheint es - ganz anders als bei den regulären Mengen und kontextfreien Sprachen - keine vielfältigen Anwendungsgebiete für kontextsensitive Sprachen zu geben. Dazu Salomaa: „*Indeed it has been apparent for a long time that this family [the family of context-sensitive languages] is not interesting from the point of view of applications. And I think that very few interesting results have been obtained for this family or its relation to the theory as a whole.*“²⁷

Mit der Klassifizierung von \mathcal{NL} als kontextsensitiv würde bezüglich Berechenbarkeit und Komplexität ein viel zu mächtiges Instrument gewählt. Außerdem bieten im Hinblick auf das Wortproblem kontextsensitive Grammatiken keine Eigenschaften, die effiziente Algorithmen zur Sprachverarbeitung in Aussicht stellen. Dieser zweite Aspekt widerlegt nicht, daß \mathcal{NL} kontextsensitiv ist, aber er motiviert die Suche nach Alternativen.

3.3.3 Alternativen und Verfeinerungen der Chomsky Hierarchie

In der Literatur sind eine Reihe von Grammatikformalismen vorgestellt worden, welche die grobe Einteilung der Chomsky Hierarchie insbesondere im Bereich zwischen CFL und CSL verfeinern. Andere verwerfen die Hierarchie ganz und setzen stattdessen alternative Konzepte dagegen.

Auf einen Verfeinerungsansatz gehen wir genauer ein, nämlich den der schwach kontextsensitiven Grammatiken (*mildly Context-sensitive Grammar* oder kurz *mCSG*). Diese werden in der Linguistik häufig als Referenz herangezogen. Uns eröffnet dies die Möglichkeit, am Ende der Arbeit einen Vergleich zu unseren Ergebnissen vorzunehmen.

Zur Menge der *mCSGs* zählen *Tree Adjoining Grammars* (*TAG*)²⁸. *TAGs* wurden von Joshi, Levy und Takahashi in [JLT75] eingeführt. Bei dieser Grammatik werden Produktionen durch eine Menge von geordneten Bäumen beschrieben. Es handelt sich also nicht um eine Wortgrammatik im herkömmlichen Sinne. Die generierten Wörter der Sprache sind Baumstrukturen. Die repräsentierten Satzsequenzen werden am Ende der Ableitung entlang der Blätter von links nach rechts abgelesen. Formale Eigenschaften faßt [YJ83] zusammen. Die Relevanz dieses Grammatiktypus für die Beschreibung von \mathcal{NL} untersucht Joshi in [Jos85], [Jos87]. Ausgewählte linguistische Phänomene werden betrachtet in [Kro89] und [KJ87]. Eine Beschreibung eines Fragments der englischen Sprache mit Hilfe von *TAGs* findet sich in [XTA95].

Als weitere Vertreter der Familie *mCSG* zu nennen sind *Head grammars* (*HGs*), eingeführt von Pollard [Pol84], und die sogenannten *Linear Index Grammars* (*LIGs*). Letztere wurden eingeführt von Gazdar [Gaz88]. *LIGs* sind von allgemeineren *Indexgrammatiken* (*IGs*) zu unterscheiden, die nicht auf lineares Wachstum beschränkt und deshalb schwerer zu parsen sind. Da im letzten Kapitel dieser Arbeit *LIGs* in einem Beweis benötigt werden, faßt Abbildung 3.5 die wesentlichen Eigenschaften zusammen.

Shanker und Weir [VSW94] haben gezeigt, daß die als *mildly Context-sensitive Grammars*

²⁷ [Sal81]

²⁸ Es sind sowohl die Bezeichnungen „*Tree Adjoining Grammars*“ als auch „*Tree Adjunct Grammars*“ üblich. Beide bezeichnen üblicherweise aber denselben Formalismus.

²⁹ Die Darstellung orientiert sich an [VSW94].

DEFINITION 10 (LINEARE INDEX GRAMMATIK): Eine Lineare Index Grammatik ist ein 5-Tupel $G = (V_N, V_T, V_I, S, P)$ mit²⁹:

V_N - endliche Menge von Nichtterminalsymbolen

V_T - endliche Menge von Terminalsymbolen

V_I - endliche Menge von Kellersymbolen

$S \in V_N$ - Startsymbol

P - endliche Menge von Produktionen der folgenden Form: $A[..i] \rightarrow XY[..j]Z$ oder $A[i] \rightarrow a$ mit $A, Y \in V_N$, $a \in V_T$, $X, Z \in (V_N[V_I^*] \cup V_T)^*$ und $i, j \in V_I^*$.

Für eine Grammatik $G = (V_N, V_T, V_I, S, P)$ ist ein Ableitungsschritt \Rightarrow_p für eine Produktion $p \in P$ folgendermaßen definiert: $p_1 = A[..i] \rightarrow XY[..j]Z$ angewendet auf $MA[..hi]N$ liefert $MXY[..hj]ZN$. Die Produktion $p_2 = A[..i] \rightarrow q$ angewendet auf $XA[..i]Y$ mit $X, Y \in (V_N[V_I^*] \cup V_T)^*$ liefert XqY . Kellersymbole werden also von rechts her hinzugefügt und entfernt. Sie können die Anwendbarkeit einer Regel beschränken. Ein Keller kann bei einem Ersetzungsschritt nur auf ein einziges Symbol der rechten Seite vererbt werden. Wie alle Nichtterminalsymbole ist das Startsymbol mit einem leeren Keller initialisiert. Eine nicht kontextfreie Sprache wird beispielsweise durch die LIG $G = (\{S, T\}, \{a, b, c, d\}, \{l\}, S, P)$ mit der Produktionsmenge $P = \{S[..] \rightarrow aS[..l]d, S[..] \rightarrow T[..], T[..l] \rightarrow bT[..]c, T[] \rightarrow \epsilon\}$ generiert: $L(G) = \{a^n b^n c^n d^n | n \geq 1\}$

Abbildung 3.5: Lineare Index Grammatiken

bezeichneten Formalismen tatsächlich dieselbe Klasse charakterisieren. Die beschreibbaren Zeichenketten sind *schwach äquivalent*.³⁰ Grammatiken dieses Typs sind in der Lage, eine Reihe von Sprachen zu generieren, die nicht kontextfrei sind, wie zum Beispiel $\{a^n b^n c^n d^n | n \in \mathbb{N}\}$, $\{a^m b^n c^m d^n | m, n \in \mathbb{N}\}$ und $COPY := \{ww | w \in \{a, b\}^*\}$.

Das Wortproblem für mCSG ist schwerer zu lösen als bei kontextfreien Sprachen. Die besten bekannten Parsingalgorithmen für die Klasse mCSG haben Laufzeiten in der Größenordnung von $O(n^6)$.

Das grobe Konzept der Chomsky Hierarchie kann auf unterschiedliche Weise verfeinert werden. Um auf diesem Weg zu einer Beschreibung von \mathcal{NL} zu gelangen, sind die Ergebnisse aus der Linguistik auf die Produktionen eines geeigneten Grammatiktyps abzubilden. Eine für unsere Zwecke geeignete grammatische Theorie soll dann ein Entscheidungskriterium dafür liefern, worauf die (Un-)Grammatikalität eines Satzes beruht. Der Status der Grammatikalität kann dann gleichsam berechnet werden. Wie die folgenden Abschnitte deutlich machen, ist diese Aufgabe nicht ohne formale Vorarbeit zu bewältigen. Aus historischen Gründen kommen in der Linguistik nämlich Techniken zur

³⁰ Dies bedeutet, daß sich dieselben Sprachen erzeugen lassen. *Starke Äquivalenz* liegt dann vor, wenn zusätzlich auch die Ableitungsbäume zweier Wörter übereinstimmen. Dies ist bei den mCSG-Formalismen nicht der Fall. Starke Äquivalenz ist für die syntaktisch korrekte Beschreibung einer Sprache für die Linguistik eine wichtige Eigenschaft.

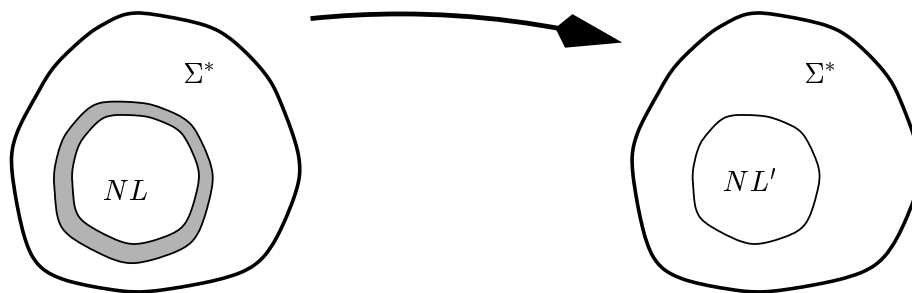


Abbildung 3.6: Ein Modell approximiert eine unscharfe Menge

Beschreibung von Regularitäten zur Anwendung, die sich nicht ohne weiteres in Produktionen umsetzen lassen.

3.4 Modelle in der Linguistik

Die Menge der grammatischen Sätze einer natürlichen Sprache stellt nur im Idealfall eine wohlbestimmte und scharf abgegrenzte Menge dar. In empirischen Experimenten erweist sich Grammatikalität nicht als fixe Größe, es existiert eine Grauzone, die formal schwer zu erfassen ist: „*Tatsächlich ist die Grenze zwischen Sätzen, die wir intuitiv als wohlgeformt auffassen, und anderen Ausdrücken nicht scharf, sondern es gibt ein Spektrum von Ausdrücken, die weder eindeutig wohlgeformt noch eindeutig nicht wohlgeformt sind, sondern nur mehr oder minder wohlgeformt. Es gibt in diesem Sinne Grade der Grammatizität.*“³¹

Dieser unscharfe Charakter einer natürlichen Sprache NL macht bekannte Formalismen zur Beschreibung von Mengen nur bedingt anwendbar. Abhilfe schafft ein Modell, welches eine klar definierte Menge charakterisiert. Gerade deshalb aber kann ein Grammatikformalismus für NL prinzipiell nicht mehr sein als eine Approximation. (Abbildung 3.6).

Die linguistische Literatur bietet unterschiedliche Modelle an, die solch eine Approximation leisten und dabei verschiedene Aspekte von NL modellieren. Die Güte einer solchen Approximation machen wir dabei *nicht* zum Gegenstand der Arbeit, weil dies eine empirische Aufgabenstellung darstellt, die allein mit formalen Methoden nicht gelöst werden kann. Die Qualität läßt sich nach unterschiedlichen Kriterien beurteilen, weil die Modelle unterschiedlichen Abstraktionsebenen voraussetzen. Sie sind dann gewissermaßen unvergleichbar. Um diese Sichtweisen benennen zu können, werden in der Literatur eine Reihe sogenannter *Adäquatheitskriterien* unterschieden.³² Damit kann zum Beispiel die Leistungsfähigkeit einen Grammatikformalismus zur Beschreibung von Sätzen aus NL

³¹ [Kut74, S. 206]

³² Als *observationell adäquat* oder auch *beobachtungsadäquat* wird eine Grammatik bezeichnet, welche ein Kriterium dafür liefert, ob ein gegebener Satz in einer Sprache möglich ist oder nicht. Eine bestimmte Struktur braucht den Sätzen dabei nicht zugewiesen zu werden [FF87a, S. 102]. Dieser Anspruch mag für viele Fälle ausreichen, unter Umständen aber sind auch tiefere Einsichten in die Zusammenhänge zwischen den Daten von Interesse: Als *deskriptiv adäquat* gilt eine Grammatik, welche ein Kriterium liefert, um grammatische Sätze einer Sprache als solche zu erkennen, und die

angemessen bewertet werden.³³ Weiter hat die Linguistik mit dem Konzept von einem fiktiven *ideal speaker*³⁴ eine Instanz konstruiert, welche die Bewertung der Güte eines Sprachmodells anhand eines bestimmten Adäquatheitskriteriums ermöglicht. Die Idee einer solchen Person, die ohne Ausnahme unbeeinträchtigt über die gesamte Kompetenz einer Sprache verfügt, stellt natürlich eine Idealisierung dar.³⁵ Chomsky macht aber deutlich, daß es zu dieser Vorgehensweise wenig Alternativen gibt: Um einen grammatischen Satz von einem ungrammatischen Satz zu unterscheiden, muß der Linguist sich auf das kompetente Urteil eines *ideal speaker* verlassen. Diese Personen zu finden und in einer Weise zu befragen, die wissenschaftlich verwertbare Daten als Grundlage liefert, stellt eine eigenständige Herausforderung dar, die der wissenschaftlichen Interpretation vorangeht. Spezifische Lebensumstände wie zum Beispiel der Bildungsstand oder die soziale Herkunft beeinflussen die ‘persönliche Muttersprache’ unmittelbar, wie auch der kulturelle Einfluß durch die Medien.³⁶

Von dieser Abstraktion des *ideal speaker* machen wir nachfolgend keinen Gebrauch. Wenn von NL und \mathcal{NL} die Rede ist, so ist einfach die vom jeweils gewählten syntaktischen Modell beschriebene Approximation gemeint. Wir verlassen uns dabei darauf, daß die von der Linguistik beschriebenen Modelle ihren Zweck gut erfüllen.

3.4.1 Sprachkompetenz und Performanz

Beobachtungen von Spracherwerbsprozessen führen Chomsky zu dem Schluß, daß der „*ideal speaker/hearer*“ über ein spezifisches Wissen verfügt, von dem er ständig Gebrauch macht. Dennoch hat er zu dem Wissen um die angewendeten Mechanismen keinen unmittelbaren Zugang, der es ihm erlauben würde, dieses Wissen in Form einer Menge von

darüber hinaus auch eine zugrundeliegende Struktur herauszuarbeiten vermag [FF87a, S. 50]. Hier ist zusätzlich zu der Antwort JA/NEIN noch eine abstrakte Repräsentation der Daten gesucht. Auch dieser Anspruch ist für manche Zwecke nicht weitgehend genug. So spielen zum Beispiel Faktoren wie Lernbarkeit oder Repräsentierbarkeit bei den ersten beiden Maßstäben überhaupt keine Rolle [FF87a, S. 63]. Als *explanatorisch adäquat* oder beschreibungsadäquat wird eine Theorie für \mathcal{NL} bezeichnet, welche in der Lage ist, die Vielfalt aller sprachlichen Erscheinungsformen auf grundlegende Prinzipien zurückzuführen [FF87a, S. 137]. Diese am weitestgehende Anforderung ist besonders umfassend und von den drei hier angegebenen am schwersten zu erfüllen.

³³ Ein praktisches Beispiel erläutert [Hau99, 131]

³⁴ Der *ideal speaker* soll bei folgendem Dilemma helfen: Der Begriff der Grammatikalität unterstellt, daß es grundsätzlich ein sinnvolles Ansinnen darstellt, einen grammatischen Satz von einem ungrammatischen Satz zu unterscheiden. Zugleich ist klar, daß der Beschreibungsgegenstand \mathcal{NL} nicht formal gegeben ist. Chomsky schlägt vor, eine fiktive menschliche Instanz als Entscheidungskriterium für den Grad der Grammatikalität zu definieren, und schlägt dazu das Konzept eines sogenannten „*ideal speaker/hearer*“ vor. Dessen Urteil ist maßgeblich für das Grammatikalitätsurteil und sein Sprachvermögen wird zum eigentlichen zum Forschungsgegenstand der Linguistik erklärt: „*Der Gegenstand einer linguistischen Theorie ist in erster Linie ein idealer Sprecher/Hörer, der in einer völlig homogenen Sprachgemeinschaft lebt, seine Sprache ausgezeichnet kennt und bei der Anwendung seiner Sprachkenntnis in der aktuellen Rede von solchen grammatisch irrelevanten Bedingungen wie – begrenztes Gedächtnis – Zerstreuung und Verwirrung – Verschiebung in der Aufmerksamkeit und im Interesse – Fehler (zufällige oder typische) nicht affiziert wird.*“ [Cho65, S. 13] (in der deutschen Übersetzung)

³⁵ Eine kritische Erörterung dieses konzeptuellen Ansatzes findet sich in [HSW80, Kapitel 2: Grundprobleme am Beispiel der Generativen Transformationsgrammatik]

³⁶ In der Dialektologie stellt sich beispielsweise das Problem der Auswahl repräsentativer ‘Gewährspersonen’. Siehe hierzu [Koe82]

Regeln zu verbalisieren. Diese Art von Wissen wird deshalb auch als *tacit knowledge* beschrieben.³⁷ So stellt sich die Frage, wie der Linguist versucht, Zugang zum Wissen des *ideal speaker* zu erlangen. Die Suche nach einem linguistischen Modell, welches bezüglich des Kriteriums der explanatorischen Adäquatheit eine hohe Qualität erreicht, kommt bei Chomsky deshalb in dem Bestreben zum Ausdruck, die Beschaffenheit des *tacit knowledge* zu ergründen. Dabei rückt die Sprachkompetenz des Sprechers bei Chomsky³⁸ in den Mittelpunkt des Interesses und unterscheidet dabei Kompetenz von Performanz.³⁹

3.5 Eine Universalgrammatik als Grundlage generischer Algorithmen

Bis zu diesem Abschnitt lies die Darstellung offen, ob disjunkte natürliche Sprachen auch durch verschiedene Grammatiken beschrieben werden. Ergebnisse der Syntaxforschung legen nun den Schluß nahe, daß es wesentliche Gemeinsamkeiten gibt. Zahlreiche Arbeiten liefern Argumente, welche die Position stützen, daß sich die syntaktischen Phänomene natürlicher Sprachen kompakter beschreiben lassen, wenn dies auf der Grundlage einer gemeinsamen Beschreibung geschieht. Diese These gründet auf der Feststellung, daß sich über alle bekannten Sprachen hinweg Universalien beobachten lassen. Entsprechend wird in der Linguistik keine Sprache als einer anderen Sprache strukturell überlegen angesehen: „*It is not claimed that evolution transfers simple languages into more complex ones. It is quite widely agreed among linguists that no language, living or dead, is „better“ than any other. Language alters, but it neither improves nor degenerates. For scientific discussions, Modern English maybe is better than Old English but the potential resources of both languages are the same.*“⁴⁰

3.5.1 Das logische Problem des Spracherwerbs

Spracherwerbsstudien zeigen, daß es dem Kind unmöglich ist, allein auf der Basis der sprachlichen Daten, die es aus seiner sozialen Umgebung heraus erfährt, genau den richtigen Regelapparat zu finden, den es faktisch in den ersten Lebensjahren erwirbt. Auf der Basis der Daten, der sich das Kind ausgesetzt sieht, wären eine Reihe von alternativen Sprachmodellen denkbar. Das Kind wählt aber immer das richtige aus. Dieser Umstand ist in der Literatur auch als *das logische Problem des Spracherwerbs*⁴¹ bekannt. Das Problem mündet in die Frage: Wie kommt das Kind an das grundlegende sprachliche Wissen, welches es zum Erwerb der Sprachkompetenz benötigt?

³⁷ [FF87a] diskutieren diesen Begriff im Kapitel: „*Grammatik als System mentaler Repräsentationen*“

³⁸ [Cho57]

³⁹ Hierbei spielt die Beobachtung eine Rolle, „... daß der Sprecher, der eine Sprache kennt, in dieser Sprache auch grammatikalisch korrekte Äußerungen hervorbringen kann, die er vorher noch nie gehört hat. Er verfügt über die Elemente des betreffenden Sprachsystems und über die Regeln für ihre Verknüpfung; er kann von endlichen Mitteln unendlich Gebrauch machen. Diese seine Fähigkeit wird als »Kompetenz« bezeichnet [...] . Das Anwenden dieser Kompetenz, also das Hervorbringen von Sprachäußerungen, wird »Performanz« [...] genannt.“ [Pel84, S. 154]

⁴⁰ [MS97, S.8]

⁴¹ Zur Diskussion dieser Fragestellung siehe auch [FF87a, S. 101ff].

3.5.2 Eine angeborene Universalgrammatik liefert Basisinformationen

Allein aus der Umwelt kann die zum Spracherwerb nötige Information nicht kommen: „*How much preprogramming is there in the brain when a child starts learning a language? Any child can learn any language without making as many grammatical and other mistakes as one would expect from a structure without programming, a tabula rasa. It can be claimed that language is as innate in the infant as flight is in the eaglet. This means that children do not so much learn language as somehow just develop it in response to a stimulus.*“⁴² Dies bedeutet, daß es gleichsam ein angeborenes Programm gibt, welches den Lernprozeß in die richtige Richtung weist. Bereits mit der Geburt ist ein Vermögen angelegt, welches Chomsky als *Universalgrammatik* (UG) bezeichnet. Diese UG spezifiziert parametrisierte Regeln, die nicht endgültig fixiert sind, sondern Freiheitsgrade in ihrem Anwendungsbereich haben.

Mit diesem parametrisierten Modell liefert Chomsky eine elegante Lösung für das *logische Problem des Spracherwerbs*: Im Rahmen des Spracherwerbs werden Freiheitsgrade nach und nach eingeschränkt und fixiert. Dies kann das Kind auf der Basis der erfahrenen linguistischen Daten seiner Umwelt leisten.

	Sprache 1	Sprache 2	...	Sprache n
erlernt:	Parametersatz 1	Parametersatz 2	...	Parametersatz n
angeboren:	Universalgrammatik (UG)			

Abbildung 3.7: Eine parametrisierte Universalgrammatik beschreibt alle Sprachen

Universalsprachliche Eigenschaften können als durch die UG mitgegeben (= angeboren) betrachtet werden. Sprachspezifische Eigenschaften müssen dagegen individuell erlernt werden. Als Kind die spezifische Grammatik einer (Mutter-)Sprache zu erlernen, heißt dann nur noch, als ursprünglich variabel angelegte Parameter zu fixieren. Auf dieser Weise wird das Ziel erreicht, in einem einzigen Modell sowohl universalsprachliche als auch sprachspezifische Konzepte zu vereinen. Für die vorliegende Arbeit stellt die UG-These das wesentliche Fundament dar. Kritisch gesehen verlagert der Lösungsvorschlag von Chomsky das logische Problem des Spracherwerbs nur auf eine andere Ebene weg vom Individuum hin zur Gattung Mensch, so daß sich ein ‘logisches Problem des phylogenetischen Spracherwerbs’ ergibt. Wie aus evolutionärer Sicht die UG entstanden ist, bleibt bei Chomsky ungeklärt⁴³. Eine Reihe von Fragen bleiben zu klären, um deutlich zu machen, welcher Selektionsdruck zu einer Form von Sprache führt, die wir heute als Gegenstand der Untersuchung betrachten.⁴⁴ Eine vergleichende Literaturrecherche zeigt, daß bezüglich dieser Frage kein Konsens auszumachen ist.⁴⁵

⁴² [MS97, S.9]
⁴³ Dies wird nicht oft kritisiert: Siehe als Ausnahme von der Regel zum Beispiel [Hat87]
⁴⁴ [Hur92]
⁴⁵ Die vorgeschlagenen Ansätze sind zum Teil sehr unterschiedlich, erweisen sich aber als empirisch nicht

Die große Anzahl an natürlichen Sprachen und die reichhaltige Menge an Phänomenen läßt vermuten, daß ein umfangreiches Regelsystem nötig ist, um mit einem einzigen Modell für alle Sprachen einen erschöpfenden Grad an explanatorischer Adäquatheit zu erreichen. Hierin besteht die große Herausforderung für die linguistische Forschung, ein Modell herauszuarbeiten, welches mit sehr viel weniger Parametern auskommt, als es linguistische Phänomene gibt.

3.5.3 Motivation: Universalgrammatik verspricht generische Lösungen

Das Konzept UG unterstellt, daß viele (syntaktische) Phänomene in natürlichen Sprachen mit einem einzigen (parametrisierten) Formalismus beschrieben werden können. Dies stellt generische Verfahren in Aussicht. Algorithmen – zum Beispiel zur automatischen Übersetzung – müssen dann nicht für jedes Paar von Sprachen neu geschrieben werden. Vielmehr sollte sich ein auf dem UG-Konzept basierender Algorithmus auf der Grundlage einer gemeinsamen Datenstruktur mit vergleichbar wenig Aufwand an andere Sprachen anpassen lassen. Abbildung 3.8 illustriert diesen Gedankengang.

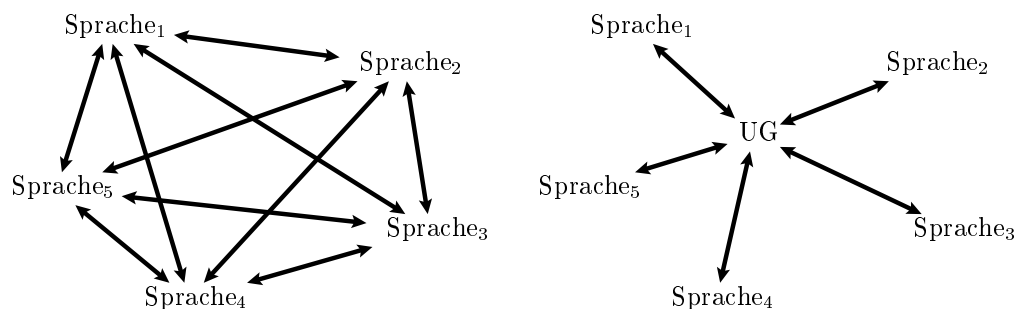


Abbildung 3.8: Eine UG erlaubt generische Ansätze für linguistische Algorithmen

Mit dieser Motivation soll jetzt auf ein Syntaxmodell detailliert eingegangen werden, welches die Idee einer UG konkret umsetzt. Chomsky hat dieses Modell unter dem Namen *Government and Binding* vorgestellt.

3.6 Government & Binding (GB)

Das Syntaxmodell *Government and Binding* (GB) wurde in seiner ursprünglichen Konzeption von Chomsky vorgestellt.⁴⁶ Seitdem hat das Modell in zahlreichen Veröffentlichungen viele Modifikationen und Verfeinerungen erfahren. Infolge dieser Entwicklung hat sich eine Theorieninflation ergeben, die nicht leicht zu überschauen ist.⁴⁷ Eine ähn-

überprüfbar und damit als spekulativ, vergleiche dazu beispielsweise ohne Anspruch auf Vollständigkeit [Gre92], [BP92], [Dea], [GM92], [Haw92], [JJ82], [Mei88], [Ruh92] oder [Com92]. Häufig wird das Thema auch nur eingangs erwähnt, im Inhalt dann aber auf andere Fragestellungen wie zum Beispiel Phonologieerwerb ([Lan88], [Lie92], [Kin71]) oder Morphologie ([Rom92] oder [And92]) eingegangen.

⁴⁶ [Cho81], [Cho86b], [Cho86a]

⁴⁷ Diese Situation ist allerdings keineswegs untypisch für die Literatur. 1983 beschreibt Winograd folgende Situation: "It is quite difficult for anyone not an expert in the field to read the literature, since

liche Bestandsaufnahme neueren Datums findet sich bei Hausser.⁴⁸ Durch die Vielzahl vorliegender Beiträge zu GB wird die vergleichende Bewertung unterschiedlicher Ansätze erschwert. Auch gestaltet sich die Erweiterung und das Einbringen zusätzlicher Prinzipien nicht immer einfach.⁴⁹ Neuere Entwicklungen wie Chomsky's „minimalist program“⁵⁰ in den letzten Jahren zielen darauf ab, die Regeln und Prinzipien durch abstraktere Ansätze zu ersetzen, um zu kompakteren Beschreibungen zu gelangen.

Die Prinzipien von GB beschreiben keine zulässigen Zeichenketten, sondern die Eigenschaften von zulässigen Baumstrukturen.⁵¹ Diese Baumstrukturen stehen in einer engen Beziehung zum grammatischen Satz: Diesen erhält man, indem man die Blätter des Baumes von links nach rechts abliest. Die geordnete Blattfolge wird in der Literatur auch als *yield* bezeichnet.

Mit Hilfe des Begriffes *yield* lassen sich nun auch die Schlüsselbegriffe *grammatisch* und *ungrammatisch* definieren: Für eine Zeichenkette $s \in \Sigma^*$ gilt:

$$s \text{ ist ein grammatischer Satz} \Leftrightarrow \exists SB, SB \text{ Syntaxbaum} : yield(SB) = s$$

Andernfalls ist s ungrammatisch. Ein ungrammatischer Satz wird üblicherweise mit dem Symbol $*$ markiert wie in: $*$ „John believes that loves Mary“.⁵² Den zu Beginn dieses Kapitels untersuchten Begriff *wohlgeformt* identifizieren wir fortan mit dem Begriff *grammatisch*.

3.6.1 GB als repräsentationelles Modell

GB wurde von Chomsky mit dem Anspruch konzipiert, syntaktische Strukturen mit einem hohen Grad an explanatorischer Adäquatheit zu modellieren. Das bedeutet, daß nicht nur grammatische Sätze als solche erkannt, sondern darüber hinaus auch tieferliegende Mechanismen beschrieben werden sollen. Damit bietet er zum Beispiel eine Lösung

the discussion is usually couched in terms of detailed syntactic arguments, understandable only in the context of all the proposals that went before. We could characterize the contents of a particular paper (or segment of a paper) in terms of a multi-dimensional matrix of theories, authors, papers, phenomena, mechanisms, and criteria. Each element is something like ‘Author A_1 in paper P_1 argues that a particular set of data D_1 is handled better by a particular combination of mechanisms M_1 within a theory T_1 than by another set M_2 which was proposed by author A_2 in paper P_2 ’ or ‘A collection of mechanisms M_1 is to be preferred on theoretical grounds over other mechanisms M_2 that deal with the same data D_1 on the grounds of some meta-theoretical criterion C_1 .’ [Win83, S. 557]

⁴⁸ [Hau99, S.179]

⁴⁹ Winograd weiter: *In order to have a new coherent theory, one cannot typically make a single small modification to a previous theory. Any one change has consequences for other parts of the system, and anything major can lead to a wholesome revision of a number of assumptions. As a result, any one paper or any one person's position is based on a complex mixture of modifications and assumptions that depend on each other. An argument for a change must be seen in light of the entire system, not just the specific details of the proposal. In addition, different formal devices can have equivalent effects, so it is important to sort out the problems being solved by a given proposal from the formal details of the solution.* “ [Win83, S. 557]

⁵⁰ siehe [Cho93] und [Cho94]

⁵¹ Die dazu benötigten Mechanismen, die aus einer Menge beliebiger Graphen zulässige Kandidaten identifizieren, werden auf den nächsten Seiten erläutert.

⁵² Diese Konvention läßt sich nach Hausser mindestens bis zu Bloomfield [Blo33] zurückverfolgen.

für das Lernbarkeitsproblem. Konzeptionell ist GB als *Prinzipien- und Parametermodell* angelegt: Eine Menge zugrundeliegender Wohlgeformtheitsprinzipien beschreibt die notwendigen elementaren Eigenschaften von zulässigen Strukturen, während die Parameter eine Adaption an sprachspezifische Eigenheiten ermöglichen.

GB setzt dazu eine beliebige Graphenstruktur als gegeben voraus. Die parametrisierten Prinzipien unterscheiden zulässige von unzulässigen Graphen. Ein Graph, der alle GB-Prinzipien erfüllt, heißt dann *Syntaxbaum*.⁵³

Dieser in GB zur Anwendung kommende Filtermechanismus wird in der Literatur auch als *repräsentationelle Methode* bezeichnet und von *derivationellen* Ansätzen unterschieden.

3.6.2 Derivationelle vs. repräsentationelle Mechanismen

Die Begriffe *derivationell* und *repräsentationell* beschreiben unterschiedliche Herangehensweisen dafür, eine Menge zu definieren:

Die *derivationelle Methode* gründet auf einer Menge von Axiomen und Folgerungsbegriffen, die es erlauben, durch beliebig häufige aber endliche Anwendung eine potentiell unendlich große Menge von Instanzen zu generieren. Die im Zusammenhang mit der Chomsky Hierarchie bekannten Wortgrammatiken sind derivationell in diesem Sinne.

Die *repräsentationelle Methode* definiert Wohlgeformtheitskriterien auf für eine zunächst nicht beschränkte Menge. Nur solche Elemente, die allen Kriterien genügen, gehören letztlich der Menge an. GB ist repräsentationell in diesem Sinne.

Bei der repräsentationellen Methode steht eine zu beschreibende Menge M nur im Idealfall am Ende eines Approximationsprozesses. Jedes zusätzliche Wohlgeformtheitskriterium verkleinert die Menge M_i , die beschriebenen Mengen werden mit jedem zusätzlichen Kriterium monoton fallend immer kleiner. Es kann durchaus passieren, daß in Folge eines unglücklich gewählten mächtigen Ausschlußprinzips schließlich die leere Menge beschrieben wird. Die folgende Abbildung skizziert den angestrebten Approximationsprozeß bezüglich einer natürlichen Sprache für fortwährend verbesserte Modelle GB_i :

$$L(GB_1) \supseteq L(GB_2) \supseteq \dots \supseteq L(GB_n) = NL$$

Umgekehrt können derivationelle Mechanismen in dem Sinne als Approximation „von innen“ betrachtet werden, als daß sie tendenziell eher zu wenig, als etwas falsches modellieren. Eine zusätzliche Produktion kann die Menge M erweitern, nie aber verkleinern. Mit fortwährend besserer Modellierung ergibt sich eine monotone Folge M_1, M_2, M_3, \dots von Mengen, die immer umfassender werden. Da weder strenge Monotonie vorliegt noch eine Beschränkung, kann dieser Folgenbegriff keine Konvergenz bezüglich NL garantieren. Mit zu mächtigen Produktionen wird im Extremfall Σ^* generiert. Folgende Abbildung

⁵³ Streng angewendet läßt diese Definition es nicht zu, daß ein Syntaxbaum ein GB-Prinzip nicht erfüllt (denn dann wäre er ja gar kein Syntaxbaum). Um zu erklären, warum ein ungrammatischer Satz fehlerhaft ist, finden sich zur Illustration in der Literatur dennoch auch 'degenerierte Syntaxbäume', also solche Graphen, bei denen 'fast alle' Prinzipien eingehalten werden.

skizziert den angestrebten Approximationsprozeß für fortwährend verbesserte Grammatiken G_i bezüglich einer natürlichen Sprache NL :

$$L(G_1) \subseteq L(G_2) \subseteq \dots \subseteq L(G_n) = NL$$

Abbildung 3.9 demonstriert unterschiedliche Herangehensweisen zur Spezifikation einer Menge M im Bild in einer Notation, wie wir sie auch nachfolgend verwenden werden. Bei der derivationellen Methode sind Produktionen nur sinnvoll, wenn sie von einem gemeinsamen Startsymbol aus erreichbar sind. Bei der repräsentationellen Methode stehen die filternden Prinzipien gleichberechtigt nebeneinander.

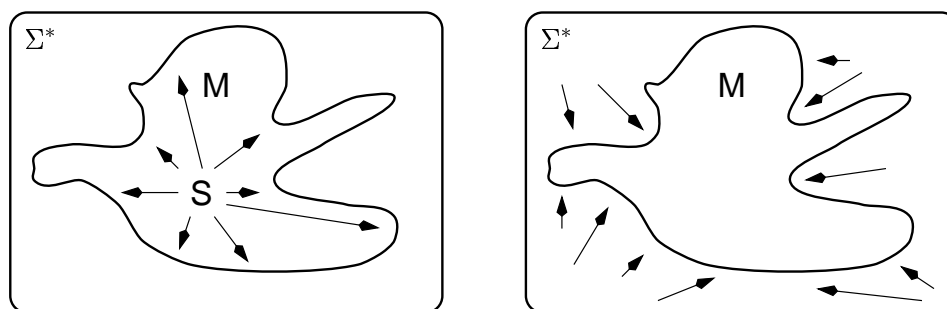


Abbildung 3.9: Approximation von „innen“ und von „außen“

Die Unterschiede zwischen den Mechanismen seien am Beispiel der Definition freier Bäume illustriert. Es wird deutlich, daß beide Mechanismen in der Informatik praktische Verwendung finden:

freier Baum derivationell:

Jeder isolierte Knoten ist ein freier Baum.

Sei t ein freier Baum. Erzeuge nun einen neuen Knoten k . Verbinde k mit einem beliebigen Knoten aus t . Dann ist auch der neu entstandene Graph ein freier Baum.

freier Baum repräsentationell:

Gegeben sei ein beliebiger ungerichteter Graph g . Wenn es zwischen zwei beliebigen Knoten immer genau einen Weg gibt, dann handelt es sich um einen freien Baum.

Deutlich wird an den Beispielen, daß derivationelle Mechanismen einen konstruktiven Charakter haben. Die repräsentationelle Technik dagegen betrachtet nur fertig gegebene Strukturen und bewertet diese. Sie liefert ein Entscheidungskriterium, um gute von schlechten Strukturen zu unterscheiden. Das Wortproblem läßt sich mit der repräsentationellen Methode trivial lösen: Eine Instanz wird einfach daraufhin überprüft, ob sie jedem filternden Prinzip genügt.

Dieser Vorteil verkehrt sich bei der Generierung von Strukturen ins Gegenteil: Als repräsentationelles Modell fragt GB nicht nach der Herkunft von Syntaxbäumen. Es beschränkt sich darauf, *beliebige* Bäume zu bewerten. Dieser Umstand kommt terminologisch darin zum Ausdruck, daß GB Syntaxbäume als *basisgeneriert* bezeichnet. Es gibt

Informatik: Theorie der formalen Sprachen	Linguistik: Syntax
Alphabet mit elementaren Symbolen	Alphabet als Baustein von Wörtern oder: Lexikon mit Wörtern
Wort = Sequenz von Symbolen	Satz = Sequenz von Wörtern
Grammatik	Syntax/Grammatik
Ableitungsbaum	syntaktische Struktur
Sprache = Menge von Wörtern	Sprache = Menge von Sätzen

Abbildung 3.10: Beziehungen wichtiger Schlüsselbegriffe in Informatik und Linguistik

keine Theorie, wie man diese Strukturen systematisch aufbaut oder konstruiert. Aus der Sicht von GB sind sie einfach da.

Dieser Sachverhalt wird in der Literatur auch als *Format-Problem*⁵⁴ bezeichnet. Hierin mag eine wesentliche Ursache dafür begründet liegen, daß nur wenige Parser existieren, die ausgewählte Konzepte von GB umsetzen.⁵⁵ Solche Systeme müssen zusätzlich zur Lösung des Wortproblems auch die Aufgabe bewältigen, zumindest einen Teil von GB so zu formalisieren, daß das Format-Problem für das betrachtete Sprachfragment gelöst wird.

Für die in der Linguistik relevanten Aufgabenstellungen wie zum Beispiel den Vergleich von syntaktischen Phänomenen bei zwei unterschiedlichen Sprachen stellt das Format-Problem kein Hindernis dar. Für die Implementierung linguistischer Algorithmen auf der Grundlage von GB ist dieser Zugang aber problematisch, denn um die Wohlgeformtheitskriterien in einem Algorithmus anwenden zu können, braucht dieser eine zu bewertende Graphenstruktur.

Als einen grundsätzlichen Lösungsansatz für das Format-Problem verfolgen wir in dieser Arbeit das Ziel, den GB-Formalismus in eine Menge von derivationellen Regeln zu übersetzen. Mit der Aufgabenstellung einer entsprechenden Umformulierung setzen wir uns in Kapitel 5 auseinander. Die derivationellen Regeln werden dann als Produktionen einer Graphgrammatik ausgedrückt.

3.7 Zusammenfassung

Das vorangehenden Kapitel führt in linguistische Grundbegriffe und Methoden ein, die in dieser Arbeit eine Rolle spielen. Die Beschreibung erfolgt aus der Perspektive der Informatik und beschäftigt sich schwerpunktmäßig damit, wie die linguistischen Methoden aus der Sicht der Theorie der formalen Sprachen zu bewerten sind. Dabei wird deutlich, in welche Relation die unterschiedlichen Begriffe zueinander stehen (Abbildung 3.10).

Auf Grund überladener Begriffe ist die Zuordnung in der Tabelle nicht immer eindeutig, gibt aber recht gut die Verwendungsweise in der Literatur wieder. Vor dem Hin-

⁵⁴ [FFO92].

⁵⁵ Als Beispiel für Implementationen siehe zum Beispiel [FFO92] oder [Sch93].

tergrund der Komplexität sprachlicher Erscheinungen konzentrieren wir uns auf den Bereich schriftsprachlicher Regularitäten, der „nach unten“ durch den Wortbegriff und „nach oben“ durch den Satzbegriff beschränkt ist. Die Analyse der hierarchischen Zusammenhänge der Satzbestandteile wird in dem Gebiet der *Syntax* untersucht. Die Tabelle macht deutlich, daß es aus der Sicht der formalen Sprachen sinnvoll ist, Wörter als elementare Einheiten zu betrachten. Denkbare Verfeinerungen dieser Sichtweise berühren Fragestellungen aus Bereichen, die nicht mehr zur Syntax gehören.

Eine Betrachtung des Grammatikbegriffs leitet über zu der Frage nach der Mächtigkeit von \mathcal{NL} aus der Sicht der Chomsky Hierarchie. Empirische Beobachtungen am Beispiel des Schweizerdeutschen führen zu dem Ergebnis, daß kontextfreie Mechanismen nicht mächtig genug sind, um alle Phänomene von \mathcal{NL} zu beschreiben. Andererseits erweisen sich die kontextsensitiven Sprachen als zu mächtig. Aus diesem Grund werden einige Vorschläge zusammengestellt, welche das Konzept der Chomsky Hierarchie durch die Einführung alternativer Grammatikformalismen verfeinern. Gegenüber kontextfreien Grammatiken ist das Wortproblem hier schwerer zu lösen. Auch ist nicht klar, inwiefern sich Ergebnisse aus der Linguistik auf diese Formalismen abbilden lassen.

Aus diesem Grund betrachten wir Chomskys *Government and Binding* als Modell für syntaktische Phänomene. GB stellt eine Approximation für \mathcal{NL} dar, die in Anspruch nimmt, auch explanatorischen Adäquatheitsansprüchen zu genügen, indem es einen Lösungsvorschlag für das logische Problem des Spracherwerbs anbietet. Während eine Universalgrammatik (UG) grundlegende Prinzipien bereitstellt, wird im Spracherwerbsprozeß durch die Belegung von Parametern eine sprachspezifische Grammatik fixiert. Dieser konzeptionelle Ansatz stellt die Möglichkeit der Entwicklung generischer Algorithmen in Aussicht.

Als Modell zur Beschreibung syntaktischer Strukturen und grammatischer Sätze von natürlichen Sprachen liegt die Frage nahe, wie sich die Mächtigkeit von GB aus der Sicht der Theorie der formalen Sprachen bewerten läßt. Dies ist schon deshalb nicht einfach zu beantworten, weil die klassische linguistische Literatur selten so etwas wie eine umfassende Darstellung von GB in einer Form anbietet, die sich an den Schreibweisen orientiert, die in der Informatik üblich sind.

Es einige Arbeiten, welche sich diesbezüglich um Abhilfe bemühen.⁵⁶ GB Prinzipien werden dort mit Hilfe von Logiksprachen ausgedrückt. Dies ermöglicht zwar grundsätzlich eine formale Herangehensweise an syntaktische Regularitäten, führt aber zu komplexen und unübersichtlichen Termausdrücken. Die graphentheoretische Herangehensweise, die in dieser Arbeit den Schwerpunkt darstellt, kann ihre Ergebnisse anschaulicher präsentieren, was dem interdisziplinären Erfahrungsaustausch zugute kommt.

Da die von GB charakterisierten Strukturen sich repräsentationell mit dem Graphenmodell beschreiben lassen, erscheint es naheliegend, die Einsatzmöglichkeiten von Graphgrammatiken zur derivationalen Beschreibung syntaktischer Strukturen zu untersuchen. Ob sich \mathcal{NL} mit diesen Mitteln genau charakterisieren läßt, ist nicht von vorneherein klar – schon allein deshalb, weil der Zugang zu \mathcal{NL} über eine (in welchem Sinne auch immer) adäquate derivationale Grammatik nicht gegeben ist. Bei der Umsetzung ist also

⁵⁶ Siehe zum Beispiel [Kra97], [Rog97], [Rog98]

in jedem Falle Handarbeit bei der Modellierung gefordert. Diese Aufgabe bewältigen wir in zwei Schritten, denen jeweils eines der folgenden Kapitel gewidmet ist:

In Kapitel 4 geht es darum, die in GB formulierten Prinzipien zu formalisieren. Dabei kommt eine graphentheoretische Terminologie zur Anwendung: Es wird deutlich, daß sich die in GB formulierten Aussagen als Restriktionen auf zulässige Knoten und Kantenmarkierungen von Graphen ausdrücken lassen. Aus Gründen der Überschaubarkeit beschränkt sich die Darstellung auf ein Fragment der englischen Sprache. Für Erweiterungen auf andere Sprachen sind die zu modifizierenden Stellen markiert. Die Charakterisierung der syntaktischen Strukturen bleibt bis an diesen Punkt repräsentationell. Sie stellt für die spätere Analyse diejenige Instanz dar, die in der Linguistik der „ideal speaker“ einnimmt. Den empirischen Problemen bei der Beschäftigung mit den Begriffen Performanz und Sprachkompetenz gehen wir auf diese Weise aus dem Weg. Erst auf der Grundlage einer formalisierten Darstellung sind die Schritte der nachfolgenden Kapitel möglich.

In Kapitel 5 ziehen wir die Konsequenz aus der Beobachtung, daß keine der angesprochenen derivationellen Wortgrammatiken Mechanismen anbietet, welche es erlauben, ein repräsentationelles Prinzip unmittelbar auszudrücken. Deshalb ist Handarbeit bei der Übersetzung der Formalismen gefragt. Um dieser Aufgabenstellung zu begegnen, bieten sich grundsätzlich zwei Möglichkeiten an: Wir müssen entweder ein neues derivationelles Modell für \mathcal{NL} formulieren, welches zugleich Ansprüche bezüglich explanatorischer Adäquatheit befriedigt, oder wir müssen alternativ neue formale Werkzeuge und Grammatikformalismen schaffen. Der erste Weg erscheint gangbarer und der Erreichung dieses Ziels ist ein wesentlicher Teil des übernächsten Kapitels gewidmet.

Unfortunately theoretical linguists have consistently confused grammar formalisms with grammars. This tendency reaches its apogee in the „*Government Binding*“ framework associated with N. Chomsky and his students where the formalism employed entirely lacks a mathematical underpinning in terms of a class of admissible grammars.

GAZDAR UND PULLUM IN [GP87]

4 Eine graphentheoretische Charakterisierung von GB

In diesem Kapitel formulieren wir fünf Prinzipien, die aus einer beliebigen Menge von Graphen diejenigen markieren, die einen Syntaxbaum darstellen, der die wesentlichen GB-Eigenschaften besitzt. Dabei kommt der *repräsentationelle* Mechanismus von GB zur Anwendung: Zunächst werden in einem ersten Schritt mit Hilfe von Definitionen Begriffe gebildet, auf deren Basis dann in einem zweiten Schritt Prinzipien formuliert werden, die Restriktionen auf beliebige Graphenstrukturen aussprechen. Abbildung 4.1 illustriert diesen Vorgang der Übersetzung von einem gegebenen Modell in ein neues Modell.

Die Definitionen der Prinzipien bedient sich einer graphentheoretischen Terminologie, was eine spätere formale Analyse erleichtert. Um dabei den Hintergrund der GB-Theorie nicht aus den Augen zu verlieren, werden die mit den graphentheoretisch modellierten Prinzipien korrespondierenden GB-Begriffe begleitend zum Text erläutert. Dies erlaubt es dem bereits mit GB vertrauten Leser, Zusammenhänge und Motivation der graphentheoretischen Umschreibungen wiederzuerkennen. Für die in dieser Arbeit später folgenden formalen Betrachtungen ist der Inhalt dieser Hintergrundinformationen nicht essentiell, kann für einen schnellen Quereinstieg in GB aber hilfreich sein. Eine Einführung in GB ersetzen die begleitenden Bemerkungen jedoch nicht.

Die GB-Literatur zählt über viele Beiträge hinweg zahlreiche Prinzipien zur Charakterisierung von Syntaxbäumen auf.¹ Neben einer allen Versionen gemeinsamen Terminologie – wir nennen hier exemplarisch die Schlagwörter *X-Bar Schema*, *Kasustheorie*,

¹ Es ist nicht einfach, eine Zahl für die Menge der Prinzipien in GB anzugeben, da einige wiederum aus Unterprinzipien bestehen. Es ist nicht offensichtlich, ob diese bei einer Zählung gemeinsam oder einzeln zu erfassen sind. Der Umfang der folgenden Seiten vermittelt aber einen Eindruck von der Zahl der in dieser Arbeit berücksichtigten Phänomene.

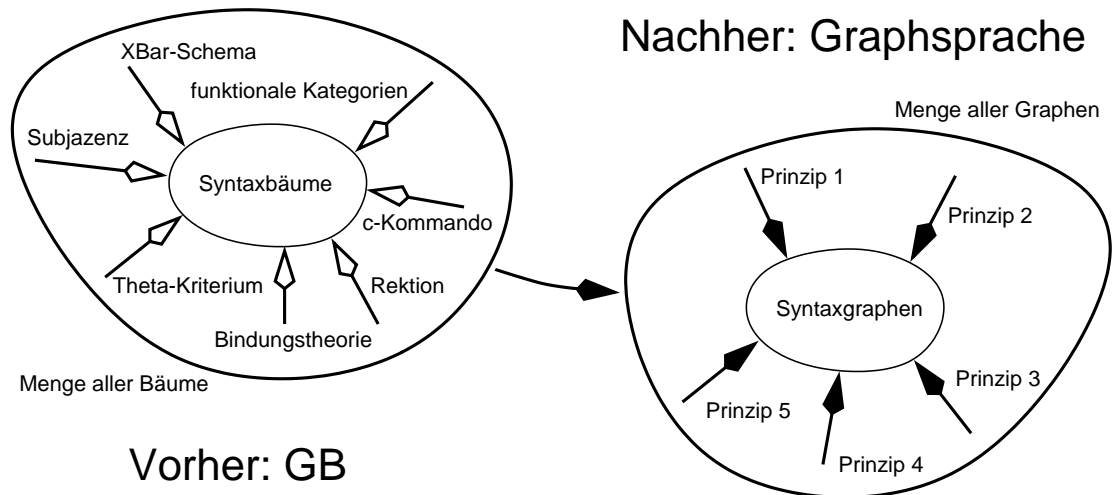


Abbildung 4.1: Ziel: GB in eine graphentheoretische Spezifikation übersetzen

Θ -Theorie, Bindungstheorie, Subjazen, c-Kommando, Rektion oder ECP – gibt es eine Reihe von individuellen Verfeinerungen. Letztere sind motiviert durch empirischen Daten aus verschiedensten Sprachen. Die Anzahl der dort speziell diskutierten Phänomene und abgeleiteten Konzepte ist umfangreich und unübersichtlich.² Das Zusammenspiel der Prinzipien und die Seiteneffekte bei der Neueinführung oder der Modifikation eines Prinzips werden in der Regel nicht in der nötigen Gesamtheit thematisiert.³

Um auf dieser Grundlage überhaupt systematisch arbeiten zu können, sprechen wir eine Reihe von Idealisierungen und Beschränkungen aus: Wir betrachten nachfolgend *ein Fragment der englischen Sprache* und verzichten dabei auf die Modellierung einer Reihe von linguistischen Phänomenen. Beispielsweise werden wir uns auf Aussagesätze beschränken und Fragesätze sowie auch Befehlssätze unberücksichtigt lassen. Auch auf Passivkonstruktionen gehen wir nicht ein. Weitere Einschränkungen werden an passender Stelle im fortlaufenden Text dargestellt. Mit dieser Vorgehensweise bringen wir zum Ausdruck, daß es uns wichtiger ist, ein eingeschränktes Syntaxmodell konsistent, als viele linguistische Phänomene ‘irgendwie’ modellieren zu können.

GB-Prinzipien lassen sich nicht einzeln in adäquate graphentheoretische Prinzipien übersetzen. Dies liegt daran, daß ein GB-Prinzip Restriktionen in ganz unterschiedlicher Hinsicht formulieren kann. Deshalb zerlegen wir die Auswirkungen der GB-Prinzipien anhand ihrer graphentheoretischen Aspekte und fassen die systematisch wieder zusammen. Wenn ein GB-Prinzip sowohl Einfluß auf zulässige Namen benachbarter Knoten hat, wie auch auf den Namen der verbindenden Kante, dann geht es in mehrere gra-

² In [BBR87] machen sich die Autoren eine formale Analyse unterschiedlicher Grammatikformalismen für \mathcal{NL} zur Aufgabe. Im Falle von GB kritisieren sie das Fehlen einer einheitlichen formalen Spezifikation und gehen deshalb auf dieses Modell nicht weiter ein.

³ „For how do we know the consequences of a theory, if, as has been pointed out on numerous occasions, GB is so highly modular that small changes in one component can have bewildering global effects? It might be trivial to evaluate the effects given a specific construction, but who can foresee the precise overall effect of such changes? And who knows the exact limits of this enterprise in full? These are serious questions and an answer is called for.“ [Kra97, S.46]

phentheoretische Prinzipien auf. Wir zeigen durch Konstruktion, wie sich GB-Prinzipien unter fünf Prinzipien subsumieren lassen, die folgende Eigenschaften eines Graphen reglementieren:

Prinzip 1: Wie sieht eine zulässige Gerüststruktur aus?

Prinzip 2: Wie darf eine Kante benannt werden?

Prinzip 3: Wie darf ein Knoten benannt werden?

Prinzip 4: Welche Regeln bestimmen die Vergabe von Indizes an Knoten?

Prinzip 5: Welche Restriktionen gibt es für Nachbarschaftsbeziehungen von Knoten?

Implizit ist in der von uns gewählten Vorgehensweise die repräsentationelle Methode enthalten, die man als sechstes (Meta)-Prinzip der Liste explizit hinzufügen könnte:

repräsentationelles Prinzip: Ein Graph ist nur dann ein *Syntaxbaum*, wenn er jedes der fünf Prinzipien erfüllt.

Aus der Sicht von GB sprachspezifischen Komponenten sind mit einer entsprechenden Randbemerkung „*Parameter*“ markiert. In einer weitergehenden Darstellung, die sich nicht am Englischen orientiert, müßten an diesen Stellen Anpassungen vorgenommen werden. Im Sinne von GB gehen wir bei der Modellierung so vor, daß sprachspezifische Änderungen stets ‘kleine’ Anpassungen erfordern.

4.1 Grundstruktur für einen GB Syntaxbaum der englischen Sprache

Ziel: Entwicklung des ersten Prinzips, welches die Beschaffenheit von zulässigen Graphen für Gerüste von Syntaxbäumen beschreibt.

Die zugrundeliegende Graphenstruktur eines GB-Syntaxbaums für eine Teilmenge der englischen Sprache kann unter Vernachlässigung der Knoten und Kantenmarkierungen durch eine Graphgrammatik definiert werden. Diese formulieren wir nachfolgend als „Prinzip 1“:

PRINZIP 1 (GERÜST EINES SYNTAXBAUMS, MINIMALANFORDERUNGEN):

Das zugrundeliegende Gerüst eines Syntaxbaums für grammatische Sätze der englischen Sprache ist ein Graph $st = (V, E, m_V, m_E)$ mit der Knotenmenge V , der Kantenmenge E , sowie jeweils einer Markierungsfunktion für Knoten und Kanten m_V und m_E . Das Gerüst wird durch die Graphgrammatik GG_{X-Bar} beschrieben. Diese kommt mit sieben

Produktionen aus, welche durchnummeriert sind, um später auf sie Bezug nehmen zu können:

Startsymbol : *NewPhrase*

Nichtterminalknoten : *NewPhrase*, *RefinePhrase*, *LexInsertion*

Terminalknoten : *max*, *project*, *head*, *grammar*

Produktionen : siehe Abbildung 4.2

□

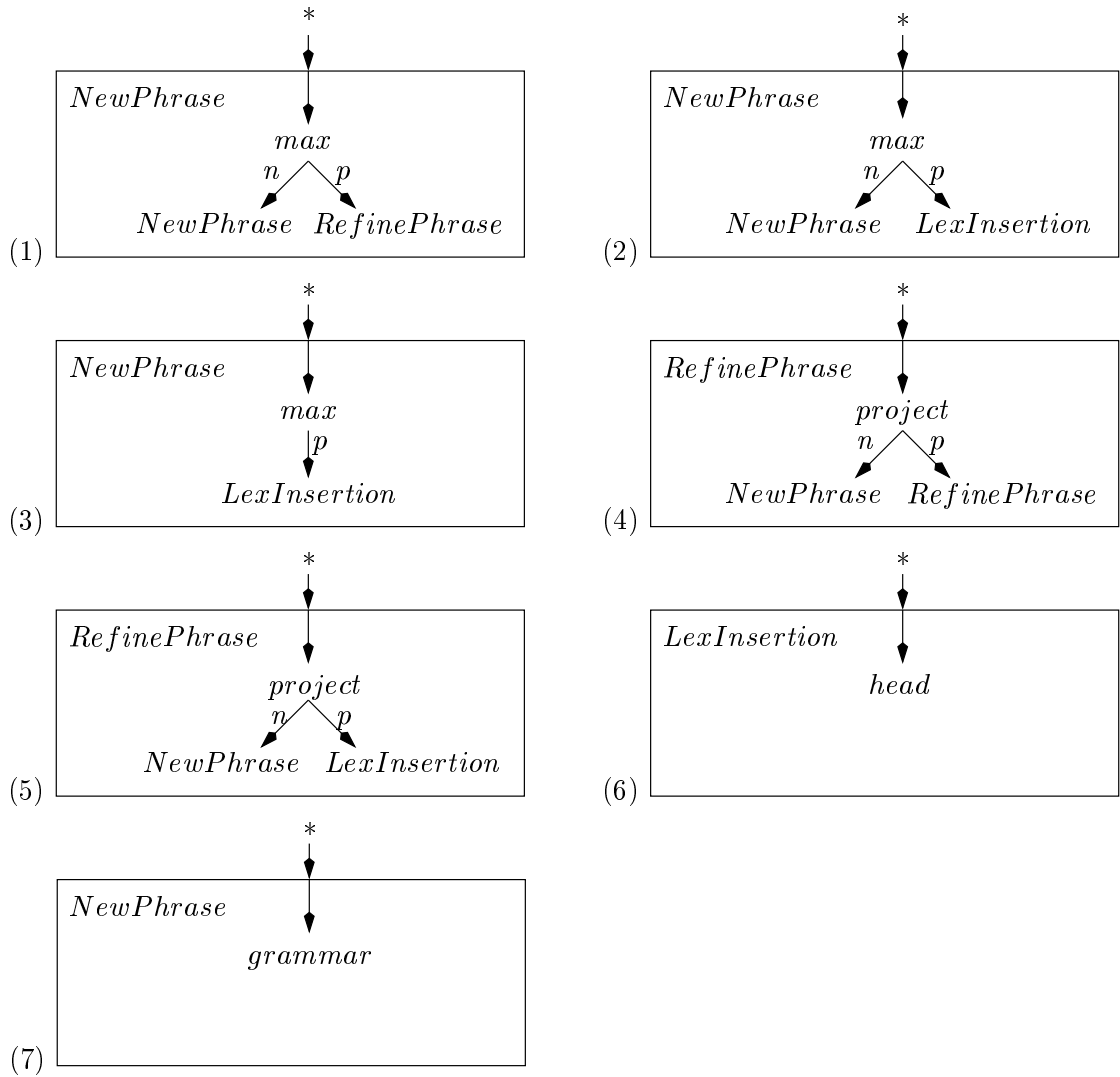


Abbildung 4.2: edNCE-Graphgrammatik für Gerüste von Syntaxbäumen

Hintergrund: Das von der Graphgrammatik GG_{X-Bar} beschriebene Gerüst ist in GB im Zusammenhang mit dem sogenannten *X-Bar Schema* bekannt. Das *X-Bar Schema* formuliert universelle Minimalanforderungen an die *lokalen* Struktureigenschaften eines

Graphen, die notwendig erfüllt sein müssen, damit er einen Syntaxbaum darstellt. Hinreichend sind diese Anforderungen aber nicht. Anders als es die Herangehensweise über eine Graphgrammatik hier erscheinen läßt, versteht GB das X-Bar Schema *nicht* als ein generierendes Konstrukt, sondern als ein Wohlgeformtheitskriterium. GB untersucht nicht die Frage, woher die zugrundeliegende Struktur stammt, sondern schreibt nur vor, daß diese dem X-Bar Schema genügen muß.⁴

Die Graphgrammatik GG_{X-Bar} erzeugt *ungeordnete* Binärbäume. Bezüglich der Knoten kommt eine Hierarchie zum Ausdruck, nicht jedoch eine links-rechts Ordnung. Daraus folgt nun *nicht*, daß jede Permutation von Knoten einer legalen Wortreihenfolge entspricht. In keiner bekannten natürlichen Sprache ist so ein Phänomen bekannt. Vielmehr wird hier *gar keine* Reihenfolge spezifiziert. Dies geschieht explizit erst durch weitere (sprachspezifische parametrisierte) Prinzipien (Prinzip 2 und Prinzip 5), die zulässige Ordnungen auf die Knoten spezifizieren. Grundsätzlich ließe sich die Ordnungsinformation zwar in die Graphgrammatik hineincodieren. Dies vermeiden wir jedoch aus zwei Gründen:

1. Das fünfte Prinzip erlaubt es, Wortstellungsphänomene sprachspezifisch zu modellieren, ohne dabei die zugrundeliegende Struktur unterschiedlich definieren zu müssen. Die Graphgrammatik $GG_{Xmbox-Bar}$ kann durch diese differenzierte Herangehensweise universell formuliert werden. Diese Art der Modellierung folgt dem umfassenden Anspruch des GB zugrundeliegenden X-Bar Schemas.
2. Eine Integrierung von Wortstellungsinformationen in $GG_{Xmbox-Bar}$ würde für jede natürliche Sprache eine eigene Graphgrammatik $GG_{Deutsch}$, $GG_{Englisch}$ etc. erfordern, was das generische Potential von GB in unserem Modell unnötig verschleiern würde.

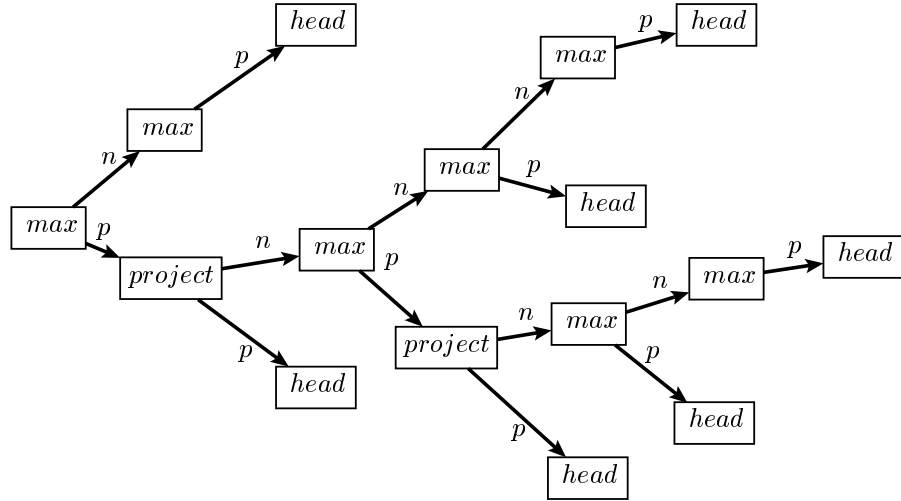
In Kapitel 5 wird bei der Entwicklung eines derivationellen Verfahrens gezeigt, wie sich die Prinzipien in ihrer Gesamtheit alle gemeinsam in einer einzigen Graphgrammatik ausdrücken lassen.

BEISPIEL 3: Eine Graphenstruktur, welche die vorangehenden Restriktionen erfüllt, zeigt Abbildung 4.3. Um zu verdeutlichen, daß der Graph keine Ordnungsrelation beinhaltet, zeigt die Darstellung ein radiales Layout:

Die Graphgrammatik ist boundary (und damit auch konfluent). Aus dem Aufbau von GG_{X-Bar} folgt eine Reihe elementarer Eigenschaften von Syntaxbäumen: Für ein Wort $st \in L(GG_{X-Bar})$ gilt stets: st ist ein Wurzelbaum mit maximalem Knotenausgangsgrad 2. Bei den durch GG_{X-Bar} beschriebenen Graphen handelt es sich also um Binärbäume. Solche Knoten mit Grad 1 heißen *Blatt*. Die übrigen Knoten heißen *innere Knoten*. Die Wurzel hat immer den Grad 2, alle übrigen Knoten mit Grad 2 sind stets unmittelbare Vorgänger eines Blattes. Daraus folgt, daß es keine langen Kettengraphen geben kann und daß ein Syntaxbaum $O(|V|)$ Blätter besitzt.

Über die Anzahl der Blätter im Syntaxbaum läßt sich noch eine schärfere Aussage formulieren.

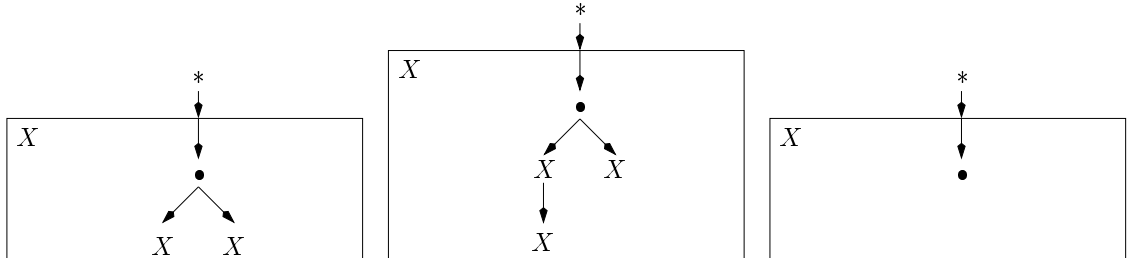
⁴ [FF87b, S.51ff], [Cul97, 134]

Abbildung 4.3: Ein Baum aus $L(GG_{X-Bar})$

SATZ 1: Für die Anzahl der Blätter B im Syntaxbaum gilt die Beziehung:

$$\frac{1}{3}|V| \leq |B| \leq \frac{1}{2}|V|$$

BEWEIS: Die Beziehungen werden deutlich, wenn man die Graphgrammatik GG_{X-Bar} umformuliert. Da die Produktionen (3) und (6) einen Zweig des Graphen terminal werden lassen, können sie in die übrigen Produktionen integriert werden. Dieser Schritt führt zwar zu einer Verdoppelung der Produktionen, wir brauchen aber nur noch drei Typen von Produktionen zu unterscheiden, was die Argumentation vereinfacht:



Die letzte Produktion läßt das Verhältnis der Knoten zu den Blättern unverändert und dient nur dazu, ein Nichtterminalsymbol terminal zu machen. Die linke Produktion vergrößert die Anzahl aller Knoten um 2 und fügt der Struktur ein Blatt hinzu. Die mittlere Produktion fügt ebenfalls ein Blatt hinzu, insgesamt vergrößert sich die Anzahl der Knoten aber um 3. Unter Anwendung dieser Produktionstypen folgt die Aussage des Satzes aus Induktion über die Produktionsanwendungen. \square

DEFINITION 11 (NAMEN FÜR KNOTEN): Ein Knoten mit der terminalen Markierung *head* heißt auch *Kopf*, ein Knoten mit terminaler Markierung *max* wird *maximale Projektion* genannt. Knoten mit der terminalen Markierung *grammar* heißen *funktionale Knoten* und die übrigen Knoten *Projektionsknoten*.

Hintergrund: Die Begriffe *Kopf* und *maximale Projektion* sind der GB-Terminologie entnommen. Der Begriff *Projektionsknoten* repräsentiert das, was im X-Bar Schema als Projektionsebene bezeichnet wird. Die *funktionalen Knoten* kennt GB unter dieser Bezeichnung nicht. Wir verstehen darunter einen Knoten, der eine Reihe von speziellen Funktionen wie Spuren, Pronomen oder Anaphern etc. strukturell einheitlich zu behandeln vermag. Technische Details dazu werden im Zusammenhang mit dem vierten Prinzip erläutert.

LEMMA 2: *Es gibt genausoviele Knoten mit der Markierung head, wie es Knoten mit der Markierung max gibt.*

BEWEIS: Durch Induktion über Ableitungsfolgen.

Hintergrund: Das Lemma macht deutlich, daß die Einhaltung des in GB verwendeten sogenannten Kopfprinzips „*Jede Phrase hat einen Kopf*“⁵ von der Graphgrammatik GG_{X-Bar} automatisch sichergestellt wird. Diese Eigenschaft braucht also nicht explizit gefordert zu werden, wie es in der GB-Literatur in der Regel geschieht.

DEFINITION 12 (PROJEKTIONSLINIE, PROJEKTIONSKANTE): *Eine Kante mit der Markierung p heißt Projektionskante. Ein maximaler Pfad⁶ entlang von Projektionskanten definiert eine Projektionslinie.*

Eine Projektionslinie stellt als Subgraph von st betrachtet einen zusammenhängenden gerichteten Kettengraphen dar. Ausgehend von einem Kopf, welcher immer ein Blatt ist, verläuft eine Projektionslinie entlang beliebig vieler vorangehender *Projektionskanten* in Richtung Wurzel bis zur ersten maximalen Projektion hin.⁷ Entlang einer Projektionslinie zweigen die Nachbarn des Kopfes ab.

BEISPIEL 4 (PROJEKTIONSLINIEN): Abbildung 4.4 zeigt die jeweils zu einer Projektionslinie gehörenden Knoten anhand einer gemeinsamen Nummer. Knoten mit gleicher Nummer sind durch eine Projektionskante verbunden. Die übrigen Kanten sind Nachbarschaftskanten. So besitzt zum Beispiel der mit ‘5’ markierte Kopf entlang seiner Projektionslinie zwei benachbarte Bäume, die wiederum einen Kopf besitzen.

Das Beispiel illustriert eine invariante Eigenschaft des Syntaxbaums. Er zerfällt stets in eine paarweise disjunkte Menge von sogenannten *Projektionslinien*, die durch Nachbarschaftskanten miteinander verbunden sind.

Die Definition über die Graphgrammatik stellt so sicher, daß jeder Syntaxbaum eindeutig in disjunkte Projektionslinien zerfällt. Ein Syntaxbaum läßt sich darüber hinaus auch in anderer Hinsicht als Komposition einfacherer Grundstrukturen ansehen. Eine entsprechende Partitionierung erscheint dann als sinnvolles Konzept, wenn sich in natürlichen

⁵ [GHS87, S. 199]

⁶ unter einem maximalen Pfad verstehen wir eine Folge von aufeinanderfolgenden Kanten, die nicht mehr durch Hinzunahme weiterer Kanten mit derselben Markierung erweitert werden kann.

⁷ Eine Ausnahme bilden die bezüglich der Kanten p isolierten *grammar*-Knoten. Da wir nicht ausschließen, daß ein Pfad die Länge 0 hat, kann auch ein solcher Knoten eine Projektionslinie bilden.

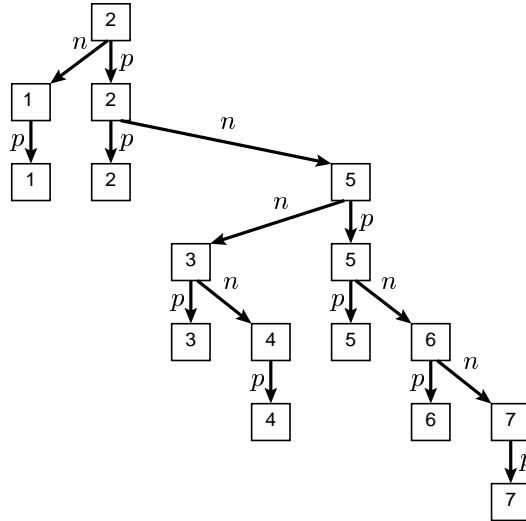


Abbildung 4.4: Syntaxbäume zerfallen in Projektions- und Nachbarschaftskanten

Sprachen Regularitäten finden lassen, die auf genau solche „Subgraphen“ Anwendung finden.

Hintergrund: Eine Partitionierung des Syntaxbaumes in wohldefinierte Teilgraphen – wie bei den Projektionslinien bereits geschehen – ist durch empirische Tests motiviert. In der Linguistik sind sogenannte *Konstituententests* üblich, die deutlich machen, welche Satzbestandteile mehr miteinander zusammenhängen, und welche weniger. Konstituententests untersuchen systematisch die intuitive Einschätzung, wonach beispielsweise bei einem Satz wie „*The man loves the woman*“ die beiden Artikel „*the*“ weniger miteinander zu tun haben, als die Artikel mit dem jeweils folgenden Substantiv. Untersuchungen dieser Art begründen die Annahme, daß ein Satz nicht nur eine Sequenz von Wörtern darstellt, sondern darüber hinaus eine hierarchische Struktur besitzt.

DEFINITION 13 (PHRASE, PROJIZIEREN): *Der von einer maximalen Projektion dominierte Teilbaum heißt Phrase. Jede Phrase hat einen Kopf, der entlang der Projektionslinie nach oben projiziert bis zur maximalen Projektion.*

Hintergrund: Ein Knoten mit der Markierung *grammar* bildet keine Phrase im Sinne dieser Definition, weil er keine maximale Projektion ist. Mit dem Knotentyp *grammar* werden linguistisch relevante Attribute repräsentiert, welche *keine* innere Struktur besitzen und damit das Gegenteil einer Phrase darstellen. Wir werden das ausnutzen für die Modellierung von GB-Begriffen wie *Spur*, *Pronomen*, *Anapher* und INFL-Attributen (zum Beispiel: [+finit]). Wir gehen auf weiterführende Arbeiten der Literatur nicht ein, welche auch für diese Knoten eine innere Struktur unterstellen⁸. Aus Gründen der einheitlichen Betrachtung macht es hier dennoch Sinn, einen Knoten mit der Markierung *grammar* als Projektionslinie der Länge 0 zu betrachten.

Die Köpfe der Projektionslinien können Wörter repräsentieren. Was die Phrase solch einem Wort hinzufügt, ist eine strukturelle Information, die zum Ausdruck bringt, in

⁸ siehe dazu zum Beispiel Stichwort „Split-INFL“ in der Folge der Arbeit von Pollock [Pol89]

welcher (syntaktischen) Beziehung die Wörter im Satz relativ zueinander stehen. Der Phrasenbegriff ist von zentraler Bedeutung für GB, weil er eine Reihe von elementaren lokalen Restriktionen ausspricht. Phrasen besitzen unabhängig von ihrer Rolle im Syntaxbaum einen gemeinsamen Bauplan. Gleichzeitig können sie als Grundbausteine angesehen werden, aus denen sich der Syntaxbaum zusammensetzt.⁹

4.1.1 Relationen zwischen Knotenpaaren

Aus der GB-Terminologie übernehmen wir folgende Begriffe, die es erlauben, Relationen zwischen Knotenpaaren im Syntaxbaum zu benennen:

DEFINITION 14 (DOMINANZ): *Seien u und v Knoten in einem Syntaxbaum: u dominiert v genau dann, wenn u auf dem (immer eindeutigen) Weg von der Wurzel zu v liegt. u dominiert v unmittelbar, wenn es eine Kante von u nach v gibt.*

Der Begriff *Dominanz* stellt damit die transitive Erweiterung des unmittelbaren Dominanzbegriffes dar. Ein Knoten dominiert sich stets selber. Es kann passieren, daß zwei Knoten u und v weder in der einen noch in der anderen Richtung in einer Dominanzrelation zueinander stehen. Das ist insbesondere bei zwei Blättern der Fall.

DEFINITION 15 (C-KOMMANDO): *Ein Knoten u c-kommandiert einen Knoten v , wenn der unmittelbare Vorgänger w von u ($w \neq v$) verzweigt und v dominiert. Außerdem darf u v nicht dominieren.*

Hintergrund: In GB spielt der Begriff c-Kommando („*constituent command*“) eine zentrale Rolle, um regelrelevante Relationen zwischen Paaren von Knoten zu benennen.¹⁰

Falls ein Knoten den Ausgangsgrad 2 hat, dann c-kommandieren die beiden unmittelbaren Nachfolger eines Knotens sich gegenseitig.

DEFINITION 16 (M-KOMMANDO): *Ein Knoten u m-kommandiert einen Knoten v , wenn die zu u gehörende maximale Projektion den Knoten v dominiert und zugleich u und v sich in keine Richtung gegenseitig dominieren.*

BEISPIEL 5: Im Baum in Abbildung 4.6 gelten folgende Relationen:

1. Dominanz: Knoten 1 dominiert alle Knoten des Baumes, Knoten 7 dominiert die Knoten 7, 8, 9 und 10. Knoten 16 dominiert nur sich selber.
2. c-Kommando: Knoten 8 c-kommandiert die Knoten 9 und 10. Knoten 10 c-kommandiert keinen Knoten und Knoten 11 c-kommandiert die Knoten 7 bis 10.
3. m-Kommando: Knoten 12 (die dazugehörige maximale Projektion ist Knoten 6) m-kommandiert die Knoten 13 bis 16, sowie 7 bis 10.

⁹ Syntaxbäume lassen sich auf der Basis von Phrasen auch rekursiv charakterisieren wie folgt: Der ganze Syntaxbaum ist eine Phrase, die wiederum aus kleineren Phrasen bestehen. Die kleinstmöglichen Phrasen sind aus dieser Sicht dann die strukturelose Köpfe (= Abbruchbedingung der Rekursion).

¹⁰ [FF87b, S.97ff], [Cul97, S.26ff]

Die hier vorgestellten Relationen sind in ihrer Reichweite nicht beschränkt. Später werden weitere Relationen vorgestellt, welche nur Aussagen über lokale Beziehungen formulieren.

4.2 Kantenbeschriftungen im Syntaxbaum

Ziel: Entwicklung des zweiten Prinzips, welches zulässige Kantenbezeichnungen für Syntaxbäume definiert.

Wie bereits dargestellt, zerfällt ein Syntaxbaum in Projektionslinien, innerhalb derer die Knoten mit Projektionskanten verbunden sind. Bis zu diesem Zeitpunkt ist die Reihenfolgeinformation der verbindenden Nachbarschaftskanten noch nicht weiter spezifiziert. Um auf den Syntaxbaum eine Ordnung einzuführen, können wir deren Rolle nun genauer definieren.

DEFINITION 17 (NACHBARSCHAFTSKANTE): *Eine Kante, die keine Projektionskante ist, heißt Nachbarschaftskante. Nachbarschaftskanten haben stets eine maximale Projektion als Zielknoten.*

LEMMA 3: *Zwei Projektionslinien sind paarweise durch höchstens eine Kante unmittelbar verbunden.*

BEWEIS: Da es sich bei einem Baum um einen zyklensfreien Graphen handelt, können zwei Projektionslinien durch höchstens eine Kante verbunden sein. Sonst gäbe es einen Zyklus. □

Die bis jetzt verwendeten Kantenmarkierungen n und p betrachten wir als durch die Graphgrammatik implizit gegeben. Explizite Kantenbeschriftungen führen wir folgendermaßen ein:

PRINZIP 2 (KANTENBESCHRIFTUNG UND ORDNUNG):

Die Kantenbeschriftungsfunktion $m_E : E \rightarrow \{l, p, r\}$ ordnet einer Kante einen Namen zu. Die Bezeichnungen stehen für eine *Links-* (l) oder *Rechtsverzweigung* (r) oder alternativ für eine *Projektionskante* (p). Zwei benachbarte Kanten sind nie gleich benannt. Die Kanten sind gemäß der Relation $l < p < r$ von links nach rechts geordnet. □

Das vorangehende Prinzip legt noch nicht fest, nach welchen Regeln die Kantenbeschriftungen vergeben werden. Dies wird im fünften Prinzip geregelt.

DEFINITION 18 (KOMPakte NOTATION VON KANTENTYPEN): *Aus dem zweiten Prinzip folgt, daß sich die Kantenmenge E eines Syntaxbaums disjunkt in drei Mengen zerlegen läßt. Diese nennen wir $L := \{ \text{Menge der Kanten mit Namen } l \}$, $R := \{ \text{Menge der Kanten mit Namen } r \}$ und $P := \{ \text{Menge der Kanten mit Namen } p \}$. Für die Menge der Nachbarschaftskanten schreiben wir kurz $N := L \cup R$.*

Die folgende Ungleichung zeigt einen Zusammenhang zwischen den Größen $|N|$ und $|P|$:

LEMMA 4 (ANZAHL DER KANTENTYPEN): *Für jeden Syntaxbaum mit mehr als einer Kante gilt:*

$$|N| = |L| + |R| \leq_{(i)} |P| \leq_{(ii)} 2 * (|L| + |R|) = 2 * |N|$$

Dies gilt nicht in dem Sonderfall eines minimalen Syntaxbaums, der unter Anwendung der Produktionsfolge (3), (6) entstehen kann. Dieser Sonderfall ist aber irrelevant, weil ein solches Gerüst durch die anderen Prinzipien sowieso herausgefiltert wird.

BEWEIS: (i) $|N| \leq |P|$: Die Graphgrammatik GG_{X-Bar} erzeugt in jedem Ableitungsschritt eine Projektionskante, aber nur bei manchen Schritten zusätzlich eine Nachbarschaftskante. Daraus folgt unmittelbar $|N| = |L| + |R| \leq |P|$. (ii) Die Beziehung $|P| \leq 2 * |N|$ ergibt sich mit struktureller Induktion über die möglichen Ableitungsfolgen von GG_{X-Bar} . □

4.2.1 Ordnungsrelation im Syntaxbaum

Die Kanten des Syntaxbaums spielen in GB eine sekundäre Rolle, weil die Ordnung des Baumes dort bereits durch die Darstellung implizit gegeben ist. Ebenso kann eine Projektionslinie anhand der Knotenmarkierung identifiziert werden. Entsprechend ist es in GB nicht nötig, Kanten explizit zu benennen. Das ist bei einer graphentheoretischen Formalisierung anders, denn ein unmarkierter Graph (V, E) enthält von Haus aus keine Ordnungsinformation.

BEISPIEL 6: Auf einen Syntaxbaum angewendet induziert die lokale Ordnung auf die Kanten für den Syntaxbaum ein kanonisches Layout. Deshalb verzichten wir bei Kantenbeschriftungen gegebenenfalls darauf, sie explizit mit anzugeben, wenn die Ordnung implizit durch die Zeichnung repräsentiert wird.

Wegen der zugrundeliegenden Baumstruktur ist der Weg zwischen zwei Knoten immer eindeutig, welches folgende Definitionen ermöglicht.

DEFINITION 19 (PFAD, *path*, PFADSTRING): *Der Pfad von der Wurzel zu einem Knoten v kann durch eine eindeutige Zeichenkette – genannt Pfadstring – über den regulären Ausdruck $\{l|r|p\}^*$ notiert werden. Für so einen String schreiben wir kurz $path(v)$.*

Mit Hilfe von Pfadstrings können Numerierungen auf der Knotenmenge eines Syntaxbaums definiert werden:

DEFINITION 20 ($num_{preorder}$): *Werden für alle Knoten v eines Syntaxbaums die Zeichenketten $path(v)$ lexikographisch sortiert¹¹, dann bezeichnet die Position der Zeichenkette $path(v)$ in der sortierten Liste die Ordnungsnummer $num_{preorder}$ von v .*

¹¹ unter der Voraussetzung $l < p < r$

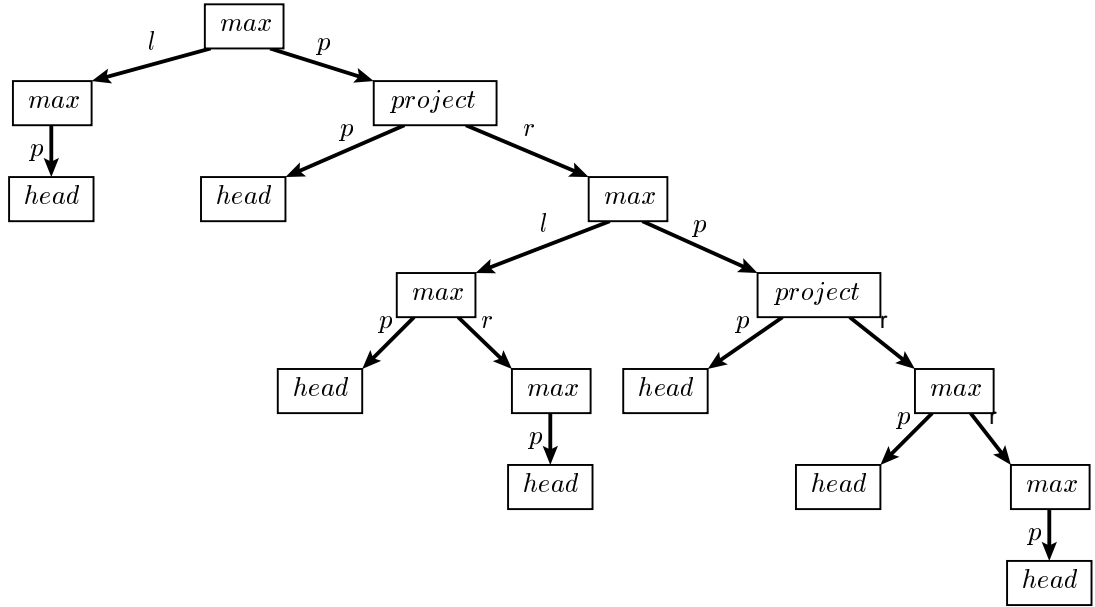


Abbildung 4.5: Syntaxbaumgerüst mit Kantenbeschriftungen

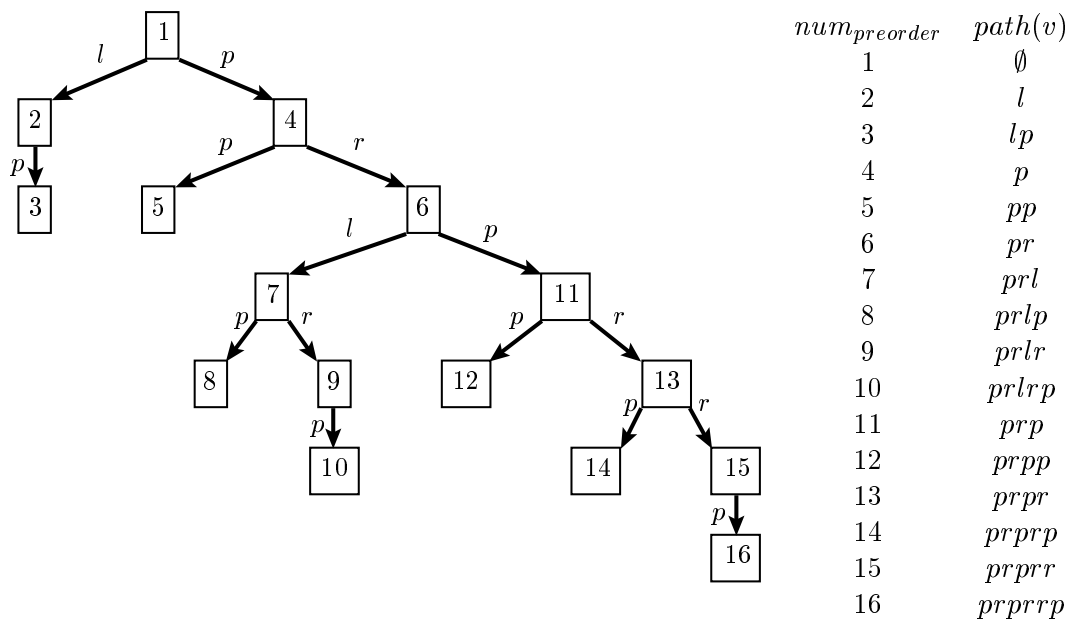
Die Bestimmung der Knotennummern über Pfadstrings erscheint an dieser Stelle unangemessen aufwendig, wird sich aber später als nützlich erweisen, wenn das Modell um weitere Kantentypen von einem Baum zum Graphen verfeinert wird.

Die geordnete Darstellung des Syntaxbaums wie in Abbildung 4.6 gründet auf der Ordnungsrelation, die wiederum auf den strukturellen Eigenschaften des Syntaxbaums basiert. In GB ist die Ordnung auf die Knoten implizit durch die Darstellung gegeben.

DEFINITION 21 (ORDNUNG AUF DIE BLÄTTER EINES SYNTAXBAUMS): Seien u und v Blätter in einem Syntaxbaum ($u \neq v$): u liegt links-von v , wenn $num_{preorder}(u) < num_{preorder}(v)$. Andernfalls liegt u rechts-von v .

In der GB-Literatur ist es üblich, die von einer Baumstruktur repräsentierte Wortstellung zu ermitteln, indem man auf der Grundlage der überschneidungsfreien Darstellung eines Baumes die Blätter von links nach rechts abliest. Dies ist aus graphentheoretischer Sicht nur dann wohldefiniert, wenn der Baum geordnet ist. Das wird in GB immer vorausgesetzt. Aus diesem Grund ist es sinnvoll, geordnete Bäume zu betrachten, was wir nachfolgend durch explizite Kantenbeschriftungen darstellen oder die Ordnungsinformation als implizit durch das Layout gegeben betrachten. Dann fallen unsere Begriffe *links-von* und *rechts-von* mit der in der Linguistik üblichen Lesart von Wortstellung zusammen.

DEFINITION 22 (yield): Als *yield* definieren wir eine Funktion $yield : \text{Syntaxbaum} \rightarrow \text{String}$, welche einen Syntaxbaum so durchläuft, daß die Blätter in der Reihenfolge von $num_{preorder}$ ausgegeben werden.

Abbildung 4.6: Ordnung auf Knoten gemäß $num_{preorder}$

4.2.2 Exkurs: Syntaktische Struktur beeinflusst Perzeptionseigenschaften

Der folgende Abschnitt zeigt, daß graphentheoretisch ähnliche Strukturen unter perzeptorischen Verarbeitungsaspekten sehr unterschiedlich abschneiden können. Wenn also das Gerüst zweier Syntaxbäume ähnlich ist, werden die repräsentierten Sätze nicht automatisch als ähnlich schwer empfunden. Wir können dies auf die Rolle der Kanten in der Struktur zurückführen.

Eine Links- oder Rechtskante ergänzt eine bestehende Phrase um zusätzliche (Unter-)Phrasen, die rekursiv wieder ergänzt werden können. Diese Beobachtung führt zu folgender Definition:

DEFINITION 23 (L-DEPTH, R-DEPTH): Als $l\text{-depth}$ ($r\text{-depth}$) eines Knotens v bezeichnen wir die Anzahl der Linkskanten (Rechtskanten), die der Pfad von v zur Wurzel enthält.

DEFINITION 24 (VERZWEIGUNGSTIEFE): Die Summe $l\text{-depth}(v) + r\text{-depth}(v)$ eines Knotens heißt *Verzweigungstiefe*.

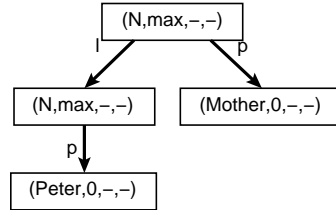
In GB spielen diese Begriffe keine Rolle, wir halten sie aber *aus linguistischer Sicht für nützlich*. Eine Reihe von empirisch ersichtlichen Verarbeitungsaspekten lassen sich so auf strukturelle Eigenschaften des Syntaxbaums zurückführen. Da Phrasen mit größerer $l\text{-depth}/r\text{-depth}$ Attributierungen von Phrasen mit kleinerer $l\text{-depth}/r\text{-depth}$ darstellen, können letztere als die wesentlicheren Bestandteile eines Satzes angesehen werden. Die $l\text{-depth}/r\text{-depth}$ eines Knotens läßt sich somit als ein *Maß für seine Bedeutung im strukturellen Satzzusammenhang* interpretieren.

Enthält ein Syntaxbaum viele Rechts- und Linkskanten bei gleichzeitig *kurzen Projektionslinien*, so handelt es sich um eine komplexe Phrase, die schwerer zu verarbeiten ist,

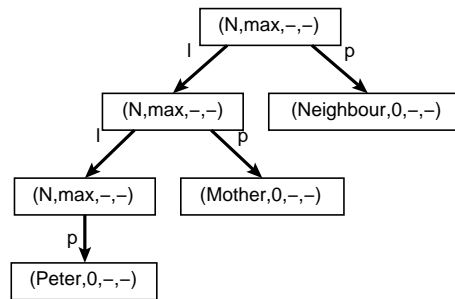
als Phrasen mit *langen Projektionslinien*. Dies verdeutlichen die folgenden Beispiele:

BEISPIEL 7: Die folgenden drei Satzfragmente werden zunehmend schwieriger zu verstehen, die Verzweigungstiefe der meisten Blätter nimmt in jedem Beispiel zu¹²:

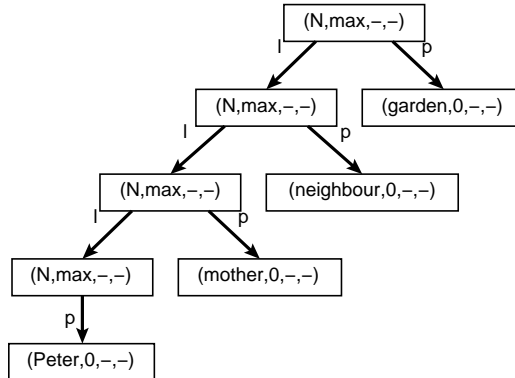
1. „*Peter's Mother.*“ – Die Mutter von Peter.



2. „*Peter's Mother's neighbour.*“ – Der Nachbar der Mutter von Peter.



3. „*Peter's Mother's neighbour's garden.*“ – Der Garten vom Nachbarn der Mutter von Peter.

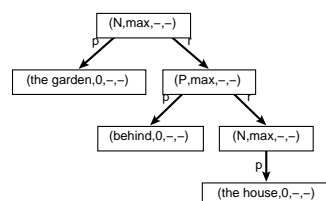


Insgesamt wirken Sätze, die Knoten mit großer Verzweigungstiefe enthalten, schwerer verständlich. Anders verhält es sich bei Sätzen mit zunehmender Länge der Projektionslinien (und dementsprechend kleiner Verzweigungstiefe. Hier werden die Sätze zwar ebenfalls länger, sind aber viel einfacher zu verarbeiten als die Sätze aus dem vorangehenden Beispiel:

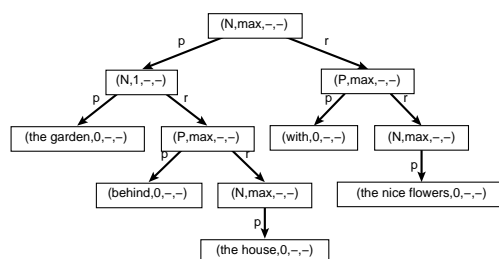
¹² Die Art der Knotenmarkierungen nimmt bereits die im dritten Prinzip eingeführte Konvention vorweg.

BEISPIEL 8: Die folgenden drei Satzfragmente enthalten Konstruktionen mit konstant beschränkter Verzweigungstiefe. Dafür wachsen in den Beispielen die Projektionslinien in die Länge.

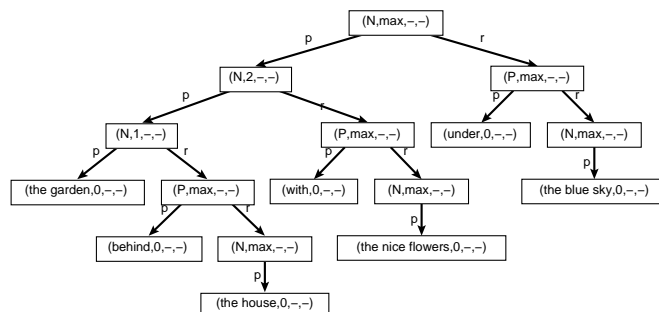
1. „the garden behind the house.“



2. „the garden behind the house with the nice flowers.“



3. „the garden behind the house with the nice flowers under the blue sky.“



Es ist bemerkenswert, daß die Sätze sich unter Verarbeitungsgesichtspunkten so unterschiedlich verhalten, obwohl die reine Graphenstruktur keinen derartig großen Unterschied nahelegt. Erst die Rolle der Kanten vermag die empirisch ersichtlichen Unterschiede auf strukturelle Eigenschaften der syntaktischen Struktur zurückzuführen.

4.3 Namen und Typen im Syntaxbaum

Ziel: Entwicklung des dritten Prinzips, welches zulässige Knotenbezeichnungen für Syntaxbäume definiert.

Neben den Kanten im Syntaxbaum sind auch die Knoten benannt. Die bis jetzt eingeführten Begriffe *maximale Projektion*, *Projektionsknoten* und *Kopf* stellen genauso wie die Begriffe *Wurzel* und *Blatt* keinen Bestandteil dieses Namens dar, weil sie als Markierungen implizit durch die Graphgrammatik $GG_{Xmbox-Bar}$ gegeben sind. Die nun folgenden Begriffe zur Namensgebung von Knoten erweitern die Möglichkeiten, einen Syntaxbaum zu spezifizieren. Damit sind zusätzliche Differenzierungen möglich.

4.3.1 Namen im Syntaxbaum

Zur Benennung von Knoten dienen mehrere Alphabete, mit deren Hilfe unterschiedliche Funktionalitäten modelliert werden. Aus formaler Sicht würde mit einer geeigneten Codierung auch ein einziges Alphabet ausreichen. Für das bessere Verständnis und für spätere Beweisansätze bietet der hier gewählte differenzierte Ansatz jedoch Vorteile.

DEFINITION 25 (ALPHABETE IM SYNTAXBAUM, LEXIKON): *Wir unterscheiden sechs Alphabete, die zur Benennung von Knoten zur Verfügung stehen:*

1. Ein *Typenalphabet* $\mathcal{T} = \{N, V_1, \dots, V_5, P, A, C, I\}$

Hintergrund: Die Elemente der Menge \mathcal{T} beinhalten nur eine Auswahl der in der Literatur üblichen Bezeichnungen. Beispiele:

N : Noun/Nomen: „*John*“, „*house*“, „*Mary*“, „*car*“, ...

$V_0, V_1, V_2, \dots, V_5$: Verb/Verb: „*rain*“ (V_0), „*seem*“ (V_1), „*dream*“ (V_2), „*believe*“ (V_3), „*love*“ (V_4), „*persuade (somebody that)*“ (V_5). Die Indizes 0 bis 5 codieren Typ und Stelligkeit des entsprechenden Symbols. Die Liste ist erweiterbar. Für verfeinerte Modellierungen sind weitere Indizes denkbar, um zusätzliche Verbtypen – zum Beispiel die ECM-Verben¹³ – zu modellieren. Wir gehen aber in jedem Fall davon aus, daß es endlich viele Typen sind.

P : Preposition/Präposition: „*with*“, „*above*“, „*next to*“, „*of*“, ...

A : Adjective/Adjektiv: „*nice*“, „*red*“, „*big*“, ...

C : Complementizer/Konjunktion: „*because*“, „*though*“, „*while*“, „*when*“, ...

I : Inflection/Flektion (in GB oft auch mit INFL abgekürzt). Dieses Symbol spielt dahingehend eine Sonderrolle, als daß es keine isolierte Wortart repräsentiert, sondern vielmehr ein Flexionsmorphem, welches sprachabhängig unterschiedlich realisiert wird. Im Englischen erscheint es als Endung „-s“ bei Verben der dritten Person („*John loves Mary.*“), als Endung „-ed“ zum Ausdruck der Vergangenheitsform oder auch als Auxiliar („*will*“, „*has*“, „*is*“, ...). Für Fragesätze (kein Thema dieser Arbeit) repräsentiert I solche Fragepartikel wie „*did*“ und „*does*“.

2. Ein *Lexikon* $\mathcal{L} \subset \text{DICTIONARY}$ mit unflektierten Wörtern.

Hintergrund: Diese Menge ist üblicherweise groß, hat aber endlich viele Elemente. Für eine Charakterisierung deutscher Syntaxbäume würden wir entsprechend

Parameter

¹³ [FF87b, S.251f]

DUDEN statt DICTIONARY schreiben. GB betrachtet das Lexikon als eine mächtige Modellkomponente, die weit mehr enthält, als nur eine Menge von Strings. Hier werden Informationen über Typen (\cong Wortart), Stelligkeit und Kontextanforderungen lokalisiert, sowie alle sprachspezifischen (nicht universalen) Aspekte, insbesondere die Belegung der Parameter.¹⁴

3. Ein Lexikon mit grammatischen Schlüsselwörtern $\mathcal{G} := \mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_3$, wobei gilt: $\mathcal{G}_1 := \{\text{PRONOMEN}, \text{ANAPHER}\}$, $\mathcal{G}_2 := \{\text{trace}\}$ und $\mathcal{G}_3 := \{[+finit]\}$

Hintergrund: PRONOMEN steht für stellvertretende Nominalausdrücke wie „he“, „she“, „it“ und ANAPHER repräsentiert Wörter wie „himself“, „herself“ oder „itself“. Der Begriff *trace* (engl.: Spur) wird später dazu benötigt, Wortstellungsphänomene zu modellieren. „[+finit]“ repräsentiert den Status einer Aussage, bei der das Verb alle für einen Satz notwendigen Informationen enthält. Denkbare Erweiterungen, auf die wir in dieser Arbeit nicht eingehen, könnten zusätzlich „[-finit]“ (für Infinitivkonstruktionen), „[+question]“ (für Fragesatz), „[+command]“ für Imperativsätze lauten. Auch Modus- und Zeitinformationen (Satz im Indikativ/Konjunktiv, Vergangenheitsform/Futur) würde GB hier lokalisieren. Da diese Arbeit sich auf Aussagesätze beschränkt, reicht der reduzierte Wertebereich für \mathcal{G}_3 aus.

4. Eine unbeschränkte Indexmenge $\mathcal{B} = \mathbb{N}_0 \cup \{\text{max}\}$

Hintergrund: Bezeichnung \mathcal{B} erinnert an den GB-Terminus *Barlevel*. Der Unbeschränktheit von \mathcal{B} liegt die Überlegung zugrunde, daß aus jedem grammatischen Satz ein längerer grammatischer Satz gewonnen werden kann, wie in: „*The beautiful big yellow . . . expensive house.*“. Dieser in der Linguistik als *Rekursion* bezeichnete Mechanismus (siehe Produktion (4) in $GG_{X\text{-}Bar}$) erlaubt es, in jeder natürlichen Sprache beliebig lange und beliebig viele Sätze zu konstruieren, obwohl nur ein endliches Lexikon zugrundeliegt. Dazu müssen manche Wörter beliebig oft wiederholt werden, wie in dem Elementarbeispiel: „*John $_{\alpha}$ dreams, \{that he $_{\alpha}$ dreams,\}^* that he $_{\alpha}$ dreams.*“

5. Eine unbeschränkte Indexmenge $\mathcal{I} := \{\alpha, \beta, \gamma, \delta \dots\} \cup \{-\}$

Hintergrund: Die Bezeichnung \mathcal{I} steht für *Index*. Die Unbeschränktheit von \mathcal{I} gründet darauf, daß in einem Satz auf unbeschränkt viele Sachverhalte immer wieder referiert werden kann, wie im folgenden Beispiel angedeutet: „*John $_{\alpha}$ dreams, that he $_{\alpha}$ dreams that Mary $_{\beta}$ dreams that she $_{\beta}$ dreams, that Peter $_{\gamma}$ dreams, that he $_{\gamma}$ sings . . .*“

6. Eine unbeschränkte Indexmenge $\mathcal{J} := \{i, j, k, l, m, \dots\} \cup \{-\}$

Hintergrund: Auch die Bezeichnung \mathcal{J} steht für *Index*. Die Unbeschränktheit von \mathcal{J} wird gebraucht, um globale, also bezüglich der Entfernung unbeschränkte Sachverhalte durch unbeschränkt häufige Anwendung von lokal beschränkten Mechanismen zu modellieren. Dieser Mechanismus wird im Zusammenhang mit Prinzip 4 (Stichwort *Subjazen*) noch genauer erläutert.

¹⁴ Die Bedeutung eines Lexikons bei automatischer Sprachverarbeitung diskutiert [GPWS96].

Die sechs vorab aufgezählten Alphabete sind paarweise disjunkt bis auf das Platzhalter-symbol „–“, welches zum Ausdruck bringt, daß von einer Option nicht Gebrauch gemacht wird.

Die Mengen \mathcal{B} , \mathcal{I} und \mathcal{J} haben jeweils *unendlich* viele Elemente. Diese Komponente des Modells übernehmen wir von GB, wo konzeptionell keine Beschränkung für diese Größen vorgesehen ist. Für eine graphentheoretische Beschreibung von GB wird sich dieser Ansatz als problematisch erweisen. Wie wir zeigen werden, lassen sich diese Schwierigkeiten aber lösen, ohne die Mächtigkeit des beschreibenden Formalismus zu beschränken.

Auf die Elemente von \mathcal{B} definieren wir die folgende Relation:

DEFINITION 26 (RELATION $<_{max}$): Seien a und b Elemente der Menge \mathcal{B} . Dann gilt:

$$a <_{max} b \Leftrightarrow \begin{cases} (a \neq max \wedge b = max) \\ \text{oder} \\ (a < b) \end{cases}$$

Anders gesagt: Jede natürliche Zahl ist kleiner als max .

DEFINITION 27 (KNOTENNAME): Ein Knotenname ist ein Viertupel $n = (\text{Bezeichner}, \text{Index}_1, \text{Index}_2, \text{Index}_3)$.

Als Motivation für die Notation als Tupelschreibweise dient die Beobachtung, daß GB viele Knotennamen – bei uns durch den ersten Tupel­eintrag beschrieben – mit einem (meist einem einzigen) Index versieht. Dieser Index kann verschiedene Funktionen und Abhängigkeiten ausdrücken, die *Indexschreibweise in GB ist stark überladen*. Wir unterscheiden mit unserer Notation drei Verwendungsweisen dieses Index und sehen dementsprechend unterschiedene Komponenten im Tupel vor. Diese Notation soll eine mehrdeutige Schreibweise durch eine eindeutige ersetzen.

Auf den Knotennamen und seine Komponenten kann mit Hilfsfunktionen zugegriffen werden:

DEFINITION 28 ($label, BEZEICHNER, INDEX, BARLEVEL$): Auf den Namen eines Knotens v kann mit der Funktion $label(v) : Node \rightarrow Name$ referiert werden. Die erste Komponente des Viertupels t heißt *Bezeichner*, die zweite, dritte und vierte Komponente von t bezeichnen wir jeweils als *Index* und greifen auf diese mit Hilfe von Projektionsfunktionen $\pi_{4,2}(t) : Name \rightarrow \mathcal{B} : (a, b, i, j) \rightarrow b$, $\pi_{4,3}(t) : Name \rightarrow \mathcal{I} : (a, b, i, j) \rightarrow i$ und $\pi_{4,4}(t) : Name \rightarrow \mathcal{J} : (a, b, i, j) \rightarrow j$ zu. Die Funktion $barlevel^{15}$ liefert die zweite Komponente des Namens von einem Knoten: $barlevel(v) := \pi_{4,2}(label(v)) \in \mathcal{B}$

¹⁵ Der Terminus *barlevel* bezeichnet in GB die Position eines Knotens innerhalb der dazugehörigen Projektionslinie. In der Literatur wird entsprechend ein Index oder entsprechend viele Markierungen notiert, also $X_2 = X''$, $X_3 = X'''$ usw.

PRINZIP 3 (NAMEN IM SYNTAXBAUM):

Die Knotenbeschriftungsfunktion $m_V : V \rightarrow ((\mathcal{T} \cup \mathcal{L} \cup \mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_3) \times \mathcal{B} \times \mathcal{I} \times \mathcal{J})$ bildet einen Knoten auf einen Namen ab. Die vorab definierten sechs Alphabete kommen gemäß Abbildung 4.7 nur bei ausgewählten Komponenten des Namens zur Anwendung. Dies motiviert dazu, die Symbole als getrennte Mengen zu verwalten.

Im Sinne der Relation $<_{max}$ sind die Knoten einer Projektionslinie vom Kopf zur maximalen Projektion hin bei 0 beginnend aufsteigend durchnummeriert. \square

Status des Knotens	$label(v) \in$
Wurzel	$(\{C, I\} \times \{max\} \times \{-\} \times \{-\})$
maximale Projektion	$(\mathcal{T} \setminus \{N\} \times \{max\} \times \{-\} \times \mathcal{J}) \cup (\{N\} \times \{max\} \times \mathcal{I} \times \mathcal{J})$
Blatt lexikalisch	$(\mathcal{L} \times \{0\} \times \{-\} \times \{-\})$
Blatt (Bindungstheorie)	$(\mathcal{G}_1 \times \{max\} \times (\mathcal{I} \setminus \{-\}) \times \{-\})$
Blatt (Wortstellung)	$(\mathcal{G}_2 \times \{max\} \times \{-\} \times (\mathcal{J} \setminus \{-\}))$
Blatt (Satzinformation)	$(\mathcal{G}_3 \times \{0\} \times \{-\} \times \{-\})$
Blattvorgänger	$(\mathcal{T} \times \{1\} \times \{-\} \times \{-\}) \cup (\mathcal{T} \times \{max\} \times \mathcal{I} \times \mathcal{J})$
sonstige	$(\mathcal{T} \times \mathcal{B} \setminus \{max\} \times \{-\} \times \mathcal{J})$

Abbildung 4.7: Knotenpositionen im Baum beschränken erlaubte Namensgebung

Hintergrund: Die Restriktionen bei der Namensvergabe orientieren sich an GB. Weil wir nur ein Fragment der englischen Sprache betrachten, gibt es einige Einschränkungen: (i) Die Wurzel: In GB steht an der Wurzel immer ein Knoten CP . Da dieser aber unter Umständen nur leere Anknüpfungspunkte dominiert, die nicht weiter verwendet werden, läßt unsere Darstellung in solchen Fällen den Syntaxbaum erst mit IP beginnen, was eine kompaktere Darstellung begünstigt. (ii) lexikalische Blattknoten: Da Indizierungen an dieser Stelle verboten sind, kann unser Modell keine Kopfbewegungen modellieren. Die Beschränkung läßt nur Bewegungen von maximalen Projektionen zu.

BEISPIEL 9 (KNOTENNAMEN): Zulässige Tupel als Namen für Knoten sind zum Beispiel $(John, 0, -, -)$, $(N, max, i, -)$, $(PRONOMEN, 0, i, -)$, $(V_1, 1, -, -)$ oder $(V_1, max, -, -)$.

Notation: Wird eine Komponente nicht weiter spezifiziert, schreiben wir dafür das Metasymbol ‘*’, das selber nicht Element der vorab definierten Alphabete ist, wie bei $(N, max, *, *)$. Für einen Bezeichner $X \in \mathcal{T}$ schreiben wir alternativ zu $(X, max, *, *)$ auch kurz X_{max} oder XP .¹⁶

Lexikalische Information befindet sich nur an den Blättern. Gemäß der Ordnung des Baumes von links nach rechts gelesen repräsentieren sie die Wortstellung eines grammatischen Satzes. Um die Eigenschaft der Grammatikalität sicherzustellen, sind allerdings noch weitere Prinzipien zu formulieren, nämlich (i) die korrekte Vergabe von Indizes (Prinzip 4) und (ii) die korrekten Nachbarschaftsbeziehungen von Knoten (Prinzip 5).

¹⁶ Die angegebenen Schreibweisen werden in GB alternativ in der Literatur gebraucht und ermöglichen fallweise eine kompaktere Darstellung.

4.3.2 Typen im Syntaxbaum

Von dem Namen eines Knotens ist sein Typ zu unterscheiden. Diese Differenzierung ermöglicht einfachere Sprechweisen, ist aus formaler Sicht aber nicht zwingend.

DEFINITION 29 (LEXIKALISCHE KATEGORIEN): *Als lexikalische Kategorien definieren wir:*

$$\mathcal{K}_{lex} := \{\mathbf{N}, \mathbf{V}_0, \mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3, \mathbf{V}_4, \mathbf{V}_5, \mathbf{P}, \mathbf{A}\}$$

Hintergrund: Es besteht in unterschiedlichen Versionen von GB kein Konsens bezüglich der hier vorgestellten Auswahl und Anzahl an Kategorien. Tatsächlich werden in der Literatur zum Teil sehr unterschiedliche Thesen zu der Frage diskutiert, welche Kategorien aus linguistischer Sicht fundamental sind und welche eher als Hilfskonstrukt zum Zwecke einer eleganten Beschreibung angesehen werden müssen.¹⁷ Fragen zu diesem Themenbereich werden in der Linguistik in dem Gebiet der *Typologie* untersucht.

DEFINITION 30 (FUNKTIONALE KATEGORIEN): *Eine funktionale Kategorie ist Element der Menge: $\mathcal{K}_{func} := \{\mathbf{I}, \mathbf{C}\}$*

Hintergrund: GB modelliert syntaktische Strukturen dergestalt, daß nicht nur jedem „sichtbaren“ Wort – den sogenannten *lexikalischen Kategorien* – ein Blatt im Syntaxbaum und eine umgebende Struktur zugeordnet wird. Zur Erklärung weniger offensichtlicher Regularitäten und funktionaler Asymmetrien von Satzbestandteilen wird zusätzlich die Existenz einer Reihe sogenannter *funktionaler Kategorien* angenommen. Neben den bereits erwähnten Kategorien \mathbf{I} und \mathbf{C} zählt GB dazu auch weitere Kategorien wie zum Beispiel \mathbf{D} (Determiner/Artikel) für Wörter wie „*der/die/das*“, „*einer/eine/eines*“. Wir umgehen in dieser Arbeit die Notwendigkeit von Determinern, indem wir uns auf Sätze mit Eigennamen („*John*“, „*Mary*“, etc.) beschränken.

DEFINITION 31 (KATEGORIEN): *Kategorien sind Elemente von: $\mathcal{K} := \mathcal{K}_{lex} \cup \mathcal{K}_{func}$*

Die Zuordnung der Kategorien zu einer der Mengen \mathcal{K}_{lex} oder \mathcal{K}_{func} ist in GB von der jeweiligen Sprache abhängig.

Parameter

Hintergrund: Ob beispielsweise Präpositionen als lexikalische oder funktionale Kategorie angesehen werden, hängt von der betrachteten Sprache ab. Im Englischen oder Deutschen ist die Kategorie \mathbf{P} lexikalisch, im Italienischen oder Spanischen hingegen nicht. Da es in GB Regeln gibt, die unmittelbar auf die Art einer Kategorie (funktional oder lexikalisch) Bezug nehmen, hat dies unmittelbaren Einfluß auf die Wortstellungen in unterschiedlichen Sprachen.¹⁸

DEFINITION 32 (TYPEN): *Jeder Knoten hat genau einen Typ. Für den Typ eines Knotens v schreiben wir $type(v)$ mit $type : Node \rightarrow \mathcal{K}$. Der Typ eines Knotens v entspricht dem Typ seines Namens, der durch den Typ des Bezeichners a gegeben ist:*

$$type(v) := type(label(v)) := type(a, b, i, j) := type(\pi_{4,1}(a, b, i, j)) = type(a)$$

¹⁷ Eine extreme Position vertritt zum Beispiel Khalaily [Kha97], der alle Wortarten auf einen einzigen nominalen Grundtypus zurückführt.

¹⁸ [FF87b, S. 213]

Der Typ ist damit letztlich abhängig von a nach folgender Konvention zu bestimmen:

1. Grundprinzip: Alle Knoten einer Projektionslinie haben denselben Typ.
2. Innere Knoten: Den Symbolen des Typenalphabets \mathcal{T} ist per Definition auf kanonische Weise ein Typ zugeordnet, den wir über die Hilfsfunktion *type* bezeichnen:
Beispiel: $\text{type}(N) \rightarrow \mathbf{N}$ und $\text{type}(V_1) \rightarrow \mathbf{V}_1$ etc.
3. Blätter: Es sind vier Typen von Blättern zu unterscheiden:
 - a) Blätter I: Die Operation *type* ist nicht nur auf Knoten sondern auch auf Zeichenketten aus der Menge \mathcal{L} definiert. Auf Elemente von **DICTIONARY** – also Strings – angewendet liefert sie die entsprechende Wortart.
Beispiel: $\text{type}(\text{„John“}) = \mathbf{N}$
Wir vernachlässigen dabei den Umstand, das es in **DICTIONARY** Elemente gibt, die sich nur im Typ, nicht aber von der Zeichenkette her unterscheiden (Beispiel: Gilt $\text{type}(\text{„ROT“}) = \mathbf{N}$ (die Farbe) oder gilt $\text{type}(\text{„ROT“}) = \mathbf{A}$ (die Eigenschaft)?)¹⁹. Statt dessen nehmen wir die Idealisierung vor, die Elemente von **DICTIONARY** als wohlunterschiedene Objekte zu betrachten.
 - b) Blätter II: Für jeden Knoten u mit grammatischem Schlüsselwort $u \in \mathcal{G}_1$ (Pronomen und Anaphern) gilt $\text{type}(u) := \mathbf{N}$.
 - c) Blätter III: Für jeden Knoten u mit grammatischem Schlüsselwort $u \in \mathcal{G}_2$ (Wortstellungsphänomene) existiert ein einziger eindeutiger innerer Knoten v mit demselben Index (siehe folgenden Abschnitt 4.4.1). Es gilt dann $\text{type}(u) := \text{type}(v)$.
 - d) Blätter IV: Für jeden Knoten u mit grammatischem Schlüsselwort $u \in \mathcal{G}_3$ gilt $\text{type}(u) := \mathbf{I}$.

DEFINITION 33 (TYP EINER PHRASE): Als *Typ einer Phrase* ist der Typ des Phrasenkopfes definiert. Eine Phrase vom Typ \mathbf{N} nennen wir *Nominalphrase*, eine Phrase vom Typ \mathbf{V} *Verbalphrase* und entsprechend für die anderen Typen.

4.3.3 Übersicht: Attribute von Knoten und Kanten

Die Prinzipien 1 bis 3 legen fest, daß zu unterscheiden ist zwischen der *Markierung*, dem *Namen*, dem *Bezeichner* und dem *Typ* eines Knotens/einer Kante. Die Markierung ist durch GG_{X-Bar} gegeben. Die Namen werden durch die Prinzipien 2 und 3 den Knoten und Kanten zugewiesen. Als Bezeichner gilt die charakteristische Komponente des Namens, wenn die Indexinformation nicht relevant ist. Mit Hilfe des Typbegriffs schließlich werden von Knoten abhängig von ihrem Bezeichner zusammengefaßt.

¹⁹ Das Beispiel „ROT“ ist noch harmlos: Crystal weist in [Cry92, S. 92] auf das englische Wort „round“ hin. Ist es ein Adjektiv („Mary bought a round table.“), eine Präposition („The car went round the corner.“), ein Verb („The yacht will round the buoy soon.“), ein Adverb („We walked round the shop.“) oder ein Nomen („It is your round.“)?

4.4 Querbeziehungen zwischen Knoten im Syntaxbaum

Ziel: Entwicklung des vierten Prinzips, welches zulässige Verwendungsweisen von Indizes reglementiert.

Über die lokalen Nachbarschaftseigenschaften der Baumstruktur hinaus können auch weit entfernte Knoten im Syntaxbaum in einer regelrelevanten Beziehung zueinander stehen, was durch einen gemeinsamen Index ausgedrückt wird. Wir bezeichnen solche Relationen als *Querbeziehungen*. In diesem Zusammenhang sind die Begriffe Koreferenz und Koindizierung hilfreich:

DEFINITION 34 (KOREFERENT): *Knoten vom Typ \mathbf{N} , die in der dritten Komponente ihres Namens (\mathcal{I}) übereinstimmen, werden (wenn es sich nicht um das Symbol „–“ handelt) als koreferent bezeichnet. Darüber hinaus betrachten wir jeden Knoten als koreferent zu sich selber.*

Hintergrund: Wir betrachten Koreferenz als Beziehung zwischen zwei nominalen Kategorien wie in „*John _{α}* believes that *he _{α}* dreams.“. Es liegt nahe, diesen Begriff auch auf nicht-nominale Kategorien zu erweitern. Aus Gründen der Einfachheit konzentrieren wir uns hier aber auf die Sichtweise, die in der GB-Literatur auch am meisten diskutiert wird.

DEFINITION 35 (KOINDIZIERT): *Knoten, die in der vierten Komponente ihres Namens (\mathcal{J}) übereinstimmen, werden (wenn es sich nicht um das Symbol „–“ handelt) als koindiziert bezeichnet. Darüber hinaus betrachten wir jeden Knoten als koindiziert zu sich selbst.*

Hintergrund: Koindizierung ist eine Beziehung, die in GB bei der Beschreibung von Wortstellungsphänomenen zum Einsatz kommt. Dazu wird eine universelle Bewegungsregel *move α* angenommen, welche Teilstrukturen im Baum bezüglich ihrer Reihenfolge reorganisieren kann, was sich auf die Wortstellung auswirkt. Im Rahmen eines Bewegungsprozesses wird am Startort eine sogenannte *Spur t* (kurz für engl. „*trace*“) hinterlassen, welche die Startposition im Syntaxbaum weiterhin belegt, im resultierenden Satz jedoch morphologisch nicht sichtbar ist. Spur und Wurzel der bewegten Phrase sind koindiziert, tragen also einen gemeinsamen Index, der nicht für andere Markierungen vergeben werden darf. Weiterhin muß *v t* c-kommandieren, es kann also nur nach oben bewegt werden. Historisch gesehen wurden zunächst eine Reihe unterschiedlicher Bewegungsprozesse unterschieden, insbesondere die lokale *NP*-Bewegung und die globale *wh*-Bewegung. Erst später wurden diese Operationen unter das allgemeinere Prinzip *move α* subsumiert.²⁰

Sowohl Koreferenz als auch Koindizierung sind als Relation gesehen *reflexiv*, *symmetrisch* und *transitiv*, bilden also eine Äquivalenzrelation auf eine Teilmenge der Knoten vom Syntaxbaum.

²⁰ [FF87b, S.149f], [Cul97, S.107]

Hintergrund: Die Begriffe Koreferenz und Koindizierung werden in der GB-Literatur nicht derart streng getrennt, wie wir es hier fordern. Analog zur Tupelschreibweise für Knotennamen vermeiden wir mit der Differenzierung auch hier eine Überladung von Begriffen.

Die durch gemeinsame Indizes zum Ausdruck kommenden Querbeziehungen überlagern die zugrundeliegende Baumstruktur eines Syntaxbaums. Wir werden in Kapitel 5 diese Querbeziehungen durch Kanten ersetzen und damit aus dem Syntaxbaum einen Syntaxgraphen machen.

4.4.1 Ketten überlagern die zugrundeliegende Baumstruktur

Auf der Basis der Indizes können nun weitere Subgraphen ausgezeichnet werden, die als *Ketten* bezeichnet werden. Ketten überlagern die zugrundeliegende Baumstruktur und spielen im Zusammenhang mit unterschiedlichen Phänomenen eine Rolle.

Hintergrund: In GB wird der Kettenbegriff in Zusammenhang mit der bereits erwähnten Transformationsoperation *move* α gebraucht. Wir zeigen nachfolgend, daß eine Reihe von Formalismen, die in GB im Zusammenhang mit dem Kapitel Bindungstheorie eine Rolle spielen, sich ebenfalls elegant mit dem Kettenbegriff beschreiben lassen, was in einer kompakteren Notation zum Ausdruck kommt.

DEFINITION 36 (KETTE, ANTEZEDENS, REFERENT, BASIS): *In einem Syntaxbaum (V, E, m_V, m_E) ist eine Kette $K = (V', E')$ ein Teilgraph mit folgenden Eigenschaften:*

1. *Alle Knoten in einer Kette tragen denselben Index. Ketten sind maximal, d.h. es gibt keinen Knoten außerhalb der Kette, der ebenfalls diesen Index trägt.*
2. *$E' = \emptyset$. Die Knoten sind isoliert, der Graph ist nicht zusammenhängend, falls er mehrere Knoten enthält.²¹*
3. *Nur ein Knoten in der Kette K dominiert eine Phrase. Dieser heißt Antezedens von K , $antecedens(k)$. Der Antezedens ist immer ein innerer Knoten, falls die Kette mehr als ein Element enthält.*
4. *Alle Knoten $v \in V'$, $v \neq antecedens(k)$ sind Blätter. Sie heißen Referenten.*
5. *Der im zugrundeliegenden Syntaxbaum am weitesten von der Wurzel entfernte Knoten einer Kette heißt Basis, kurz $base(k)$.*

Hintergrund: Der Begriff Basis ist in GB so nicht üblich. Bezüglich der Beweisungslehre erinnert die Bezeichnung an das, was in GB als *basisgenerierte Position* bezeichnet wird.

6. *Pro Projektionslinie ist höchstens ein Knoten mit einem Index versehen und dieser ist eine maximale Projektion.*

²¹ In Kapitel 5 wird diese Definition dahingehend modifiziert, daß die isolierten Knoten mit Kanten verbunden werden, so daß echte Kettengraphen entstehen.

7. Sei K eine Kette. Für alle Knoten w auf dem Pfad von einem Kettenelement $v \in K$ zur Wurzel gilt: $w \notin K$. Allerdings kann der Antezedens einer anderen Kette auf diesem Pfad liegen.

Eine Kette ist damit ein nicht zusammenhängender Teilgraph eines Syntaxbaums. Die zu einer Kette gehörenden Knoten bilden eine Äquivalenzklasse. Abgesehen vom Antezedens einer Kette sind alle anderen Kettenelemente Blätter. Basis und Antezedens können zusammenfallen, in diesem Fall besteht die Kette nur aus einem Element, enthält dann keinen Referenten.

Wir werden nun zwei Arten von Ketten im Syntaxgraphen definieren, die sich durch unterschiedliche Zusatzeigenschaften unterscheiden: α -Ketten und β -Ketten.

4.4.2 α -Ketten modellieren GB-Bewegungsprozesse

Zusammen mit einigen zusätzlichen Begriffen werden wir in diesem Abschnitt auf der Basis des Kettenbegriffs eine sogenannte α -Kette definieren.

Hintergrund: Was sich nachfolgend kompakt hinter den Begriffen *Subjazen*, *trace* und *Kette* verbirgt, orientiert sich an dem in GB sehr umfangreichen Gebiet der Bewegungstheorie. Dort erlaubt die zentrale *Transformationsregel move α* , sämtliche Permutationen über die Satzbestandteile zu bilden: „*Bewege irgendeine Kategorie in irgendeine Position*.“²² Offensichtlich repräsentiert aber nur eine kleine Teilmenge aller Permutationen über die Wörter eines Satzes einen zulässigen Satz. GB behandelt dieses Auswahlproblem repräsentationell, indem das Ergebnis von *move α* bewertet wird durch eine Reihe von Prinzipien, welche die infolge der Bewegung entstandenen Baumstrukturen anhand von strukturellen Merkmalen für grammatisch oder ungrammatisch erklären (repräsentationelle Methode). Um solche Restriktionen formulieren zu können, werden dort Aussagen über Start- und Zielort einer bewegten Teilstruktur formuliert.

Die Idee eines Transformationsmechanismus *move α* , welcher die Position von Teilbäumen im Baum verändert, impliziert die Existenz einer Vorher- und Nachhersituation. In GB werden diese beiden Repräsentationsebenen als *D-Struktur* („deep structure“/„Tiefenstruktur“) und *S-Struktur* („surface structure“/„Oberflächenstruktur“) unterschieden. Für GB ist die resultierende *S-Struktur* die eigentlich interessante und wird als *basisgeneriert* gegeben angenommen. Wir folgen in dieser Arbeit dieser Sichtweise. Tatsächlich werden in GB darüber hinaus noch weitere Repräsentationsebenen angenommen (logische Form *LF* und phonetische Form *PF*), auf die wir hier aber nicht weiter eingehen.²³

Der in dieser Arbeit verfolgte Ansatz beschreibt allein das Endresultat der von GB unterstellten Bewegungsoperationen (also die S-Struktur) in Form eines Graphen. Aus diesem Grund ist es anders als bei GB nicht nötig, Transformationsregeln auf Graphen zu definieren. Für einen graphentheoretischen Ansatz, welcher sich an der D-Struktur orientiert, siehe [Bör95].

²² [FF87b, S. 150]

²³ [FF87b, S.190,224], [Cul97, S.20]

DEFINITION 37 (BARRIERE): Ein Knoten v mit $\text{type}(v) = \{\mathbf{I}, \mathbf{N}\}$ und $\text{barlevel}(v) = \max$ heißt Barriere.

Hintergrund: Barrieren begrenzen die Reichweite einer Transformation. Die hier angegebene Definition ist aus der Sicht von GB nicht erschöpfend, weil es für die Gesamtheit aller sprachlichen Phänomene nicht ausreicht, Barrierenknoten alleine an Ihrem Typ festzumachen²⁴. Für unser Ziel, nur ein Fragment der englischen Sprache zu modellieren, reicht dieser Ansatz aber aus.

DEFINITION 38 (SUBJAZENZ): Ein Knoten u ist subjazent zu einem Knoten v , wenn es exklusive u und v höchstens eine Barriere auf dem eindeutigen Weg von u nach v gibt und v u c -kommandiert.

DEFINITION 39 (REGIEREN, REKTION, REGENS): Ein Knoten u regiert einen Knoten v , wenn u ein Kopf ist und er zugleich v m -kommandiert. Außerdem darf der eindeutige Pfad zwischen u und v (ohne Knoten v) höchstens eine maximale Projektion enthalten. Wenn u v regiert, heißt u auch Regens von v .

Hintergrund: Der Begriff Rektion ist ein für GB zentraler Begriff („government“), welchem das Modell den ersten Buchstaben seines Namens verdankt.²⁵

DEFINITION 40 (LEXIKALISCHE REKTION): Wenn ein Knoten u v regiert und $\text{type}(u) \in \mathcal{K}_{\text{lex}}$, dann wird v lexikalisch regiert von u .

DEFINITION 41 (ANTEZEDENS REKTION): Wenn ein Knoten u v regiert und beide Knoten koindiziert sind, dann liegt eine Antezedens Rektion vor.

DEFINITION 42 (STRENGE REKTION): Ein Knoten v wird streng regiert (engl. „proper government“), wenn es ein Element u gibt, so daß bezüglich u und v zugleich sowohl lexikalische als auch Antezedens Rektion vorliegt.

DEFINITION 43 (α -KETTE): Sei st ein Syntaxbaum. Eine Menge von Knoten $K \subset V(st)$ heißt α -Kette genau dann, wenn K die folgenden Eigenschaften erfüllt:

1. K ist eine Kette (\rightarrow Def. 36) mit paarweise koindizierten Knoten.

Hintergrund: α -Ketten können unbeschränkt viele Elemente enthalten. In der Praxis tritt dieser Fall aber fast nie auf. Tatsächlich ist schon ein Wert von 3 eher selten. Auch GB sieht keine grundsätzliche universelle Beschränkung vor, wie weit Bewegungen reichen können. Allerdings stellen [FF87b, S.248] fest, daß es wenn auch keine logische, so doch eine stilistische Grenze zu geben scheint: „Entscheidend ist hier, daß Sätze zunehmend schlechter werden, je mehr Barrieren überschritten werden.“

2. Der Antezedens von K hat die Bezeichnung (a, \max, j, i) , $i \in \mathcal{I}$, $j \in \mathcal{J}$, $a \in \{N, C\}$.

Hintergrund: Hierin kommt die Einschränkung zum Ausdruck, daß nur maximale Projektionen bewegt werden, nicht aber Köpfe. Dies wird von GB zwar erlaubt, wir gehen aber zugunsten einer kompakteren Darstellung nicht darauf ein.

²⁴ [Cul97, S.228ff], [FF87b, S.158ff]

²⁵ [FF87b, S.107], [Cul97, S.27]

3. Für jeden Referenten u in K gibt es in K einen Knoten v mit: u subjazent zu v . Im Sinne der Subjazenrelation sind die Kettenelemente von links nach rechts geordnet.

Hintergrund: Das Subjazenprinzip aus GB besagt, daß Bewegungen nie mehr als einen Grenzknoten überschreiten dürfen. Dies schränkt die Reichweite von Bewegungen ein, was jedoch dadurch kompensiert wird, daß Bewegungen zyklisch stattfinden dürfen. Dies ist beispielsweise für manche Topikalisierungsprozesse nötig, wie sie bei Fragesätzen auftreten: „*What_i did you think trace_i John bought trace_i?*“²⁶

4. Alle Referenten der α -Kette tragen die Bezeichnung $(trace, max, -, i), i \in \mathcal{I}$ und müssen streng regiert sein.

Hintergrund: In GB fordert das Empty Category Principle (ECP), daß Spuren streng regiert sein müssen²⁷. Diese Regel ist für die linguistische Forschung dahingehend exemplarisch, als daß es mit ihrer Hilfe gelungen ist, eine Reihe von bis dahin unabhängig voneinander formulierten Prinzipien (*Nominative Island Condition* (NIC), *Subject Condition*, *that-trace-effect*) auf eine einzige Regularität zurückzuführen. Viele Arbeiten bis hin zum Chomskys *minimalist program* widmen sich dem erklärten Ziel, die Anzahl nötiger Prinzipien und Definition durch die Suche nach vergleichbaren Prinzipien weiter zu reduzieren.

5. Bis auf die Basis werden alle Kettenelemente von einem Knoten $(CP, max, *, *)$ unmittelbar dominiert.

Hintergrund: Diese Bedingung sagt etwas darüber aus, wohin eine Bewegung stattfinden darf. Als Landeplätze kommen nur *CP*-Specifier Positionen in Frage. Auch dies stellt eine Vereinfachung gegenüber GB dar.

Prinzip 4 wird eine Aussage über α -Ketten formulieren und ebenfalls über β -Ketten, die im folgenden Abschnitt eingeführt werden.

4.4.3 β -Ketten modellieren bindungstheoretische Beziehungen

Auf Grund seiner Baumstruktur besitzt ein Syntaxbaum eine elementare separierende Eigenschaft: Durch Löschen einer Kante im Syntaxbaum zerfällt dieser in zwei Zusammenhangskomponenten. Wir nutzen diese Eigenschaft aus, um eine Klasse von Subgraphen gesondert auszuzeichnen.

DEFINITION 44 (UNTERSNTAXBAUM): Jede Zusammenhangskomponente, welche wir durch Löschen aller Kanten mit dem Endknoten $(C, max, *, *)$ erhalten, bezeichnen wir als *Untersyntaxbaum*.

Hintergrund: Unser Begriff von Untersyntaxbaum stellt eine grobe Vereinfachung der Konzepte *Complete Functional Complex* (CFC) und *Rektionskategorie* in GB dar.

²⁶ [FF87b, S.156ff], [Cul97, S.195ff]

²⁷ [FF87b, S.170ff]

Während diese in GB dynamische Konzepte darstellen, die sich erst aus dem Kontext des Syntaxbaums heraus ergeben²⁸, wählen wir hier einen statischen Ansatz, können dafür aber den Untersyntaxbaum unmittelbar aus der Graphenstruktur herausrechnen.

Ein Knoten gehört genau einem Untersyntaxbaum an. Syntaxbäume können in Untersyntaxbäume partitioniert werden.

DEFINITION 45 (BINDUNG, BINDEN, GEBUNDEN, FREI): *Knoten u bindet den Knoten v , wenn u v c -kommandiert und sie koreferent sind. v heißt dann gebunden. Ein Knoten, der nicht gebunden ist, heißt frei.*

Bei einer Bindung handelt es sich um eine globale Beziehung, die in ihrer Reichweite nicht beschränkt ist.

Hintergrund: Dem Terminus Bindung (engl. *binding*) verdankt der Name GB seinen zweiten Buchstaben.²⁹

DEFINITION 46 (β -KETTE): *Sei st ein Syntaxbaum. Eine Menge von Knoten $K \subset V(st)$ heißt β -Kette genau dann, wenn K die folgenden Eigenschaften erfüllt:*

1. K ist eine Kette (\rightarrow Def. 36) mit paarweise koreferenten Knoten.
2. Der Antezedens von K hat die Bezeichnung (N, max, j, i) , $j \in \mathcal{J}, i \in \mathcal{I}$.
Hintergrund: Wie bereits dargestellt kann das Konzept der Bindungstheorie auch auf nichtnominale Kategorien erweitert werden. Aus Gründen der Kompaktheit verzichten wir aber darauf, diese Option mit in unser Modell zu integrieren.
3. Alle Referenten von K tragen entweder die Bezeichnung $(PRONOMEN, max, j, -)$ oder $(ANAPHER, max, j, -)$ mit $j \in \mathcal{J}$.
4. Ein Kettenelement $PRONOMEN$ ist innerhalb seines Untersyntaxbaums frei.
5. Ein Kettenelement $ANAPHER$ ist innerhalb seines Untersyntaxbaums gebunden.

Hintergrund: Die β -Ketten sind eine Umsetzung der in GB zur Anwendung kommenden Bindungstheorie. Diese formuliert Prinzipien, die das Auftreten von Pronomen (Personalpronomen wie „*he*“, „*she*“, „*it*“ sowie Possessivpronomen wie „*his*“, „*her*“, „*its*“) und Anaphern (reflexive Fürwörter wie „*himself*“, „*herself*“, „*itself*“ und reziproke Fürwörter wie „*each other*“) im Satz reglementieren. Das „*ABC der Bindungstheorie*“ lautet in der klassischen Form nach [FF87b, S. 108]

1. Prinzip A: Anaphern müssen in ihrer Rektionskategorie gebunden sein.
2. Prinzip B: Pronomina müssen in ihrer Rektionskategorie frei sein.
3. Prinzip C: R-Ausdrücke müssen frei sein.

²⁸ „Eine Kategorie α ist ein CFC, wenn α einen Kopf β und alle mit β verträglichen grammatischen Funktionen enthält.“ [FF87b, S. 103]

²⁹ [FF87b, S.93ff], [Cul97, S.58ff]

Unter Prinzip A wird festgelegt, in welchem Bereich Bindung erfolgen muß, während Prinzip B festlegt, in welchem Bereich Bindung nicht erfolgen darf. Unter A sind alle Koindizierungen verboten, die nicht ausdrücklich spezifiziert sind, während unter Prinzip B alle Koindizierungen erlaubt sind, die nicht ausdrücklich verboten sind.

DEFINITION 47 (R-AUSDRUCK): *Die vom Antezedens einer β -Kettes dominierte Konstituente wird auch als R-Ausdruck bezeichnet.*

Hintergrund: In GB wird der Begriff R-Ausdruck als Abkürzung für „referentieller Ausdruck“ verwendet, um diesen von Pronomen und Anaphern abzugrenzen.³⁰

Aus der Definition einer β -Kette folgt automatisch das dritte Bindungsprinzip von GB, wonach innerhalb einer β -Kette zwei R-Ausdrücke nicht koindiziert sein können und daß R-Ausdrücke in einer β -Kette immer frei sind. Während GB diese Eigenschaft explizit fordert, ist sie in unserer Definition implizit enthalten.

Eine β -Kette ist damit ein nicht zusammenhängender Teilgraph eines Syntaxbaums. Es kann unbeschränkt viele β -Ketten in einem Syntaxbaum geben. Eine β -Kette kann unbeschränkt viele Knoten enthalten. Typischerweise ist die Anzahl der Knoten aber klein. So hat zum Beispiel die β -Kette im Satz „*John _{α} believes that he _{α} shaves himself _{α} .*“ drei Knoten (Anzahl gleicher Indizes) und reizt damit schon sehr aus, was in der Praxis an Koreferenz Verwendung findet.

Hintergrund: Den Begriff β -Kette, den wir hier als Erweiterung des Kettenbegriffes einführen, gibt es in GB so nicht. Der Buchstabe β erinnert an den in GB zentralen Begriff *Bindung*. Unsere Terminologie ermöglicht eine kompakte Darstellung der Regularitäten rund um die Bindungstheorie. Obwohl diese in GB unabhängig von der Bewegungslehre formuliert wird, läßt sie sich aus formaler Sicht ähnlich beschreiben. Dieser unifizierende graphentheoretische Zugang eröffnet leichtere Beweisansätze für den zweiten Teil dieser Arbeit. Zwar finden sich in der GB-Literatur immer wieder Hinweise darauf, daß es sich bei Bewegungs- und Bindungstheorie um verwandte Phänomene handelt, doch wird diese Beobachtung nicht in der Form umgesetzt, als daß eine gemeinsame Beschreibung formuliert wird.

4.4.4 Ein Prinzip für Ketten

Nach diesen Vorbemerkungen können wir nun das vierte Prinzip definieren:

PRINZIP 4 (RESTRIKTIONEN AUF KOINDIZIERTE KNOTEN):

Für ein Blatt v mit Bezeichner $x = \pi_{4,1}(\text{label}(v)) \in \mathcal{G}_1 \cup \mathcal{G}_2$ gilt abhängig von x eine der folgenden Aussagen: $x \in \mathcal{G}_1 : v \in K$, K β -Kette oder $x \in \mathcal{G}_2 : v \in K$, K α -Kette

Dieses Prinzip stellt sicher, daß PRONOMEN, ANAPHERN und *traces* nicht an beliebigen Positionen im Baum vorkommen, sondern nur in ausgewählten Kontexten, die eine grammatische Struktur sicherstellen.

³⁰ [FF87b, S.95]

Insbesondere wird an dieser Art der Präsentation deutlich, daß es sich bei Bewegungslehre und Bindungstheorie zumindest aus struktureller Sicht um verwandte Prinzipien handelt. Beide lassen sich auf der Basis als Verfeinerung eines allgemeiner definierten Kettenbegriffs einführen (Abbildung 4.8).

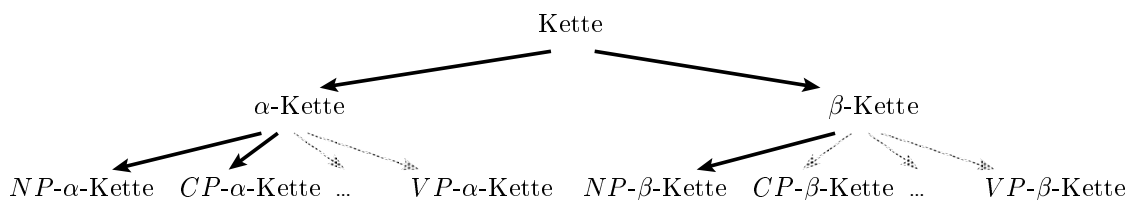


Abbildung 4.8: Zusammenhang der Kettentypen und mögliche Erweiterungen

Die schwarzen Linien erläutern die Verwandtschaft der im Text vorgestellten Begriffe. Die schraffierten Linien deuten denkbare Verfeinerungen des Modells an, die in dieser Arbeit nicht vertieft werden.

Die Reduzierung verschiedener Regularitäten auf einen gemeinsamen Kettenbegriff wird es im Kapitel 5 erleichtern, eine kompakte Modellierung für GB zu finden.

4.5 Geordnete Nachbarschaften im Syntaxbaum

Ziel: Entwicklung des fünften Prinzips, welches die zulässigen Nachbarschaftsbeziehungen von Knotennamen im Syntaxbaum reglementiert.

Verschiedene Sprachen kennen unterschiedliche spezifische Wortstellungen. Zum Vergleich betrachte man die Sätze „*Hans liebt Maria*“ (deutsch), „*John loves Mary*“ (englisch), „*Szereti János Mariát*“ (ungarisch). So bezeichnet in GB zum Beispiel der Terminus *kopffinitial* oder *kopfffinal* den Unterschied, ob ein Kopf links oder rechts vom Komplement steht. Zahlreiche manchmal komplex erscheinende Wortstellungsphänomene lassen sich auf wenige solcher lokalen Strukturregeln zurückführen.

So liegt es zum Beispiel nahe, daß Phrasen nicht beliebig zu einen Syntaxbaum zusammengesetzt werden dürfen. Tatsächlich gibt es in GB eine Reihe von (zum Teil sprachspezifischen) *Restriktionen und Anforderungen an zulässige Typkombinationen* benachbarter Phrasen. Auf dieser Weise kann sichergestellt werden, daß ein Satz mit Subjekt und Objekt auch ein geeignetes Verb enthält. Weiterhin sprachspezifisch zu reglementieren ist die *links-/rechts-Beziehung von Phrasen relativ zum Kopf*. Diese Restriktionen haben besonderen Einfluß auf die Wortstellung des beschriebenen Satzes.

Eine Spezifikation, welche solche Regularitäten zum Ausdruck bringt, bezeichnen wir als *geordnete Nachbarschaftsbeziehung* und unterscheiden dabei zwei Arten:

1. *notwendige Nachbarschaftsbeziehung*: Wenn ein Wort w an einem Blatt verwendet wird, dann *müssen* typabhängig auch eine Reihe von Kontextanforderungen erfüllt sein.

Hintergrund: Verben lassen sich beispielsweise in Typen einteilen, die jeweils einen bestimmten Kontext unbedingt erfordern. Ein Verb wie „love“ benötigt im Satz mindestens eine *NP*, die besagt wer liebt (Subjekt) und eine weitere *NP* (Objekt), die besagt, wer geliebt wird.

2. *optionale Nachbarschaftsbeziehung:* Über die notwendigen Anforderungen an den Kontext hinaus sind manchmal noch zusätzliche Nachbarschaften *erlaubt*, die aber *nicht* notwendig realisiert sein müssen.

Hintergrund: Grammatische Sätze lassen sich stets um zusätzliche Phrasen ergänzen. Allerdings sind nur bestimmte Phrasen an bestimmten strukturellen Positionen im Syntaxbaum zugelassen. Ein Minimalbeispiel hierzu wäre der Satz „Der schöne, große, . . . , alte Baum . . .“. Es ist offensichtlich, daß sich dieser Satz beliebig erweitern läßt.

Die Richtung einer Nachbarschaft wird durch die von der im zweiten Prinzip definierten Funktion m_E vergebenen Kantenbeschriftungen „l“ (für links) und „r“ (für rechts) zum Ausdruck gebracht. Nachbarschaften zu einem Kopf werden als Abzweigungen von der dazugehörigen Projektionslinie realisiert. Mit jeder zusätzlichen Nachbarschaft enthält die Projektionslinie einen zusätzlichen Knoten.

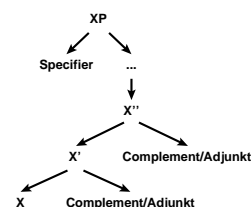
Hintergrund: Die beiden Nachbarschaftsprinzipien leisten das, was Chomsky als eine Grundbedingung für die Grammatikalität eines Satzes fordert: „*every element that appears in a well-formed structure must be licensed in one of a small number of ways*“³¹

Abhängig von der Rolle der Nachbarschaft haben sich für Teilbäume abhängig vom Satzkontext in der Literatur drei Begriffe eingebürgert, nämlich *Complement* (COMP), *Specifier* (SPEC) und *Adjunkt* (ADJ), die wir hier der Vollständigkeit übernehmen, obwohl sie strukturell keine zusätzliche Information beinhalten. Sie ermöglichen kompaktere Sprechweisen.

Hintergrund: Ein vollständiger Syntaxbaum ist in der Regel ein sehr umfangreicher Graph. Die in der linguistischen Literatur diskutierten Phänomene beschränken sich aber häufig auf lokal begrenzte Regularitäten. Deshalb ist es in GB nicht immer nötig, den ganzen Syntaxbaum darzustellen. Das strukturelle Umfeld wird in einem solchen Fall nur umschrieben, nicht aber vollständig angegeben. Zu diesem Zweck haben sich die hier vorgestellten Begriffe wie SPEC, COMP, ADJ (und verwandte Schreibweisen) als nützlich erwiesen und als Konvention durchgesetzt.

Das X-Bar-Schema in GB beschreibt üblicherweise SPEC, COMP und ADJ als Anknüpfungspunkte. In seiner klassischen Form wird es in der Literatur meist ähnlich wie folgt dargestellt [FF87b, S. 54]:

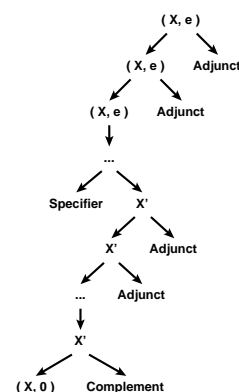
$$X^n \rightarrow \dots X^m \dots; m = n \text{ oder } m = n - 1$$



³¹ [Cho86a, S. 93]

Bezüglich der Anknüpfungspunkte zu benachbarten Phrasen findet sich das X-Bar-Schema in der Literatur immer wieder unterschiedlich ausdifferenziert. Siehe dazu nebenstehend auch Culicover [Cul97]. Der Autor diskutiert dort auch Alternativen zu dem Ansatz, Verzweigungen strikt binär zu modellieren.

Inwiefern linguistische Phänomene aus verschiedenen Sprachen Restriktionen auf Phrasenstrukturen motivieren, untersucht [Spe90]. In [Zam96] wird eine stark verfeinerte Struktur zur Repräsentation von Nominalphrasen demonstriert.



DEFINITION 48 (COMPLEMENT, SPECIFIER, ADJUNKT): Mit *Complement*, *Specifier* und *Adjunkt* werden Phrasen (also (Teil)-Bäume der syntaktischen Struktur) mit folgenden Eigenschaften bezeichnet: *Specifier* (SPEC) modellieren sowohl notwendige als auch optionale Nachbarschaftsbeziehungen. Sie stehen typischerweise links von der Projektionslinie und zweigen unmittelbar bei der maximalen Projektion ab. *Adjunkte* (ADJ) modellieren optionale Nachbarschaftsbeziehungen und können typabhängig links oder rechts von der Projektionslinie abzweigen und dies nur zwischen Specifier und Complement. Die *Complemente* (COMP) modellieren notwendige Nachbarschaftsbeziehungen. Sie stehen typabhängig links oder rechts von der Projektionslinie und zweigen unmittelbar über dem Kopf ab.

Durch diese Definition wird nicht unterstellt, daß es immer Abzweigungen dieser Art gibt. Vielmehr geht es hier darum, sie adäquat zu benennen, falls sie auftreten.

Einer Abzweigung kann in der Regel nicht alleine auf Grund des Typs angesehen werden, ob es sich um Adjunkt oder einen Specifier handelt. Hierzu sind weitere Informationen über Knotennamen und deren Kontext nötig.

BEISPIEL 10 (X-BAR SCHEMA MIT COMP, SPEC, ADJUNCT): Abbildung 4.9 zeigt eine Instanz des X-Bar-Schemas für eine Verbalphrase in einem größeren Satzkontext. Die Kontextstruktur ist jeweils unter den Dreieckssymbolen zusammengefaßt.

Erläuterung zum Sprachgebrauch: Der Kopf (*give*, 0, –, –) projiziert entlang der Projektionslinie zur maximalen Projektion (*V*, *max*, –, –) hin (entgegen der Pfeilrichtung). Rechts zweigen zwei Komplemente ab (zur Grammatikalität notwendig) und ein Adjunkt (optional, nicht notwendig). Ein Specifier einer übergeordneten Phrase auf der linken Seite beinhaltet das Subjekt der Konstruktion. Insgesamt beschreibt diese Verbalphrase die lokale Umgebung der Projektionslinie des Kopfes „*give*“, ohne dabei den gesamten Kontext zu expandieren. Der repräsentierte Satz lautet: „*My uncle Peter from Denmark gives a nice book with a detailed introduction into linguistic secrets to Mary in the afternoon.*“

Offen ist bisher geblieben, warum eigentlich Subjekt und Objekt dort stehen (müssen/dürfen), wo sie sich befinden. Das dazu nötige Prinzip entwickeln wir auf den fol-

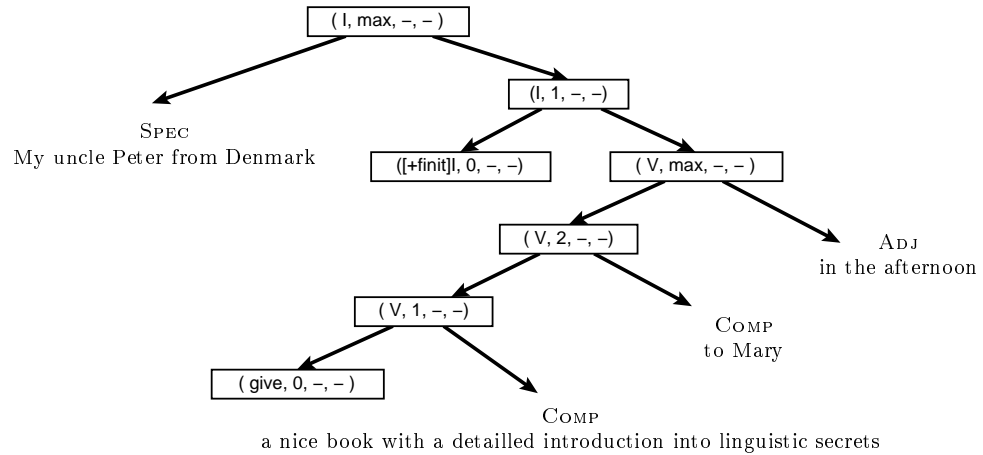


Abbildung 4.9: Eine Instanz des X-Bar Schemas

genden Seiten und geben die Restriktionen für Typ und Richtung von Nachbarschaften jeweils in Form einer Tabelle an:

DEFINITION 49 (NOTWENDIGE NACHBARSCHAFTSBEZIEHUNGEN): *Notwendige Nachbarschaftsbeziehungen sind universelle Anforderungen, die in jedem Syntaxbaum erfüllt sein müssen. Abbildung 4.10 zeigt eine Liste für die englische Sprache.*

Hintergrund: In GB können die im Lexikon enthaltenen Verben typabhängig andere Phrasen *subkategorisieren*, also das Vorhandensein anderer Phrasen im lokalen Kontext fordern. Ein Verb v kann verlangen, daß sich Phrasen vom Typ XP_1, XP_2, \dots, XP_n an bestimmten Positionen im Kontext befinden müssen. Es heißt dann auch: v *subkategorisiert* XP_1, XP_2, \dots, XP_n . So verlangt das Verb „love“ nach einem Subjekt (wer liebt?) und einem Objekt (was wird geliebt?). Dieser Vorgang wird bei uns als *notwendige Nachbarschaftsbeziehung* modelliert. Im Rahmen der sogenannten Θ -Theorie weist GB den subkategorisierten Phrasen X_i zugleich eine von ca. einem Dutzend sogenannter Θ -Rollen (thematische Rollen) zu, welche semantische Aspekte modellieren. So läßt sich der zweifelhafte Status eines Satzes wie „*The stone smashed the window with a ball.*“ mit Hilfe von Θ -Rollen erklären. Θ -Rollen tragen Namen wie zum Beispiel (hervorgehobene Wörter beschreiben Anwendungsfälle): *Agens* („**John** loves Mary.“), *Patiens* (negatives Erleiden wie in „**John** is hit.“). Zwischen einer Θ -Rolle und dem korrespondierenden Kategorientyp gibt es Abhängigkeiten: Die Rollen *Agens* und *Patiens* werden immer an eine *NP* zugewiesen, *Instrument* und *Quelle* an eine *PP* mit Präpositionen „with“ beziehungsweise „from“. Als Θ -Kriterium gilt in GB das Prinzip: „Jede Θ -Rolle muß genau einem Argument zugewiesen werden, und jedes Argument muß genau eine Θ -Rolle erhalten“³². Ein Satz, bei dem das Θ -Kriterium nicht erfüllt ist, betrachtet GB als ungrammatisch. Formalisieren läßt sich die Θ -Theorie im Rahmen eines syntaktischen Modells auf Grund der semantischen Aspekte nur bedingt.³³

BEISPIEL 11 (OBJEKT UND NEBENSATZ): Eine *NP- α* -Kette als Komplement repräsentiert ein Objekt im Satz, wie in „John loves Mary“. Eine *CP- α* -Kette als Komplement

³² [FF87b, S. 83]

³³ [Cul97, S.37ff]

Ein Kopf vom Typ	verlangt als Specifier der dominierenden <i>IP</i> (links)	verlangt als Komplement (rechts) in angegebener Reihenfolge
C*	-	<i>IP</i>
P	-	Element einer <i>NP-β</i> -Kette**
V₁	-	Basis einer <i>CP-α</i> -Kette***
V₂	Entweder Basis einer <i>NP-α</i> -Kette oder Element einer <i>NP-β</i> -Kette	-
V₃	Entweder Basis einer <i>NP-α</i> -Kette oder Element einer <i>NP-β</i> -Kette	Basis einer <i>CP-α</i> -Kette***
V₄	Entweder Basis einer <i>NP-α</i> -Kette oder Element einer <i>NP-β</i> -Kette	Basis einer <i>NP-α</i> -Kette
V₅	Entweder Basis einer <i>NP-α</i> -Kette oder Element einer <i>NP-β</i> -Kette	Sowohl Basis einer <i>NP-α</i> -Kette als auch Basis einer <i>CP-α</i> -Kette***
I****	-	<i>V₁P, V₂P, V₃P, V₄P, V₅P o. V₆P</i>

* Der Kopf einer undominierten *CP* ist leer, wird von keinem Element belegt. Diese Ausnahme resultiert aus einer Abgrenzung gegenüber GB. Dort kann diese Position sehr wohl durch weitere Bewegungsphänomene als Landeplatz für eine Phrase in Anspruch genommen werden.

** Aus einer Präpositionalphrase wird nicht herausbewegt, deshalb ist hier keine *α*-Kette. zugelassen.

*** Der Kopf dieser *CP* trägt den Bezeichner „*that*“.

**** Der Kopf einer *IP* trägt immer den Bezeichner „*[+finit]*“.

Abbildung 4.10: Notwendige Nachbarschaftsbeziehungen

repräsentiert einen Nebensatz wie in dem Beispiel „*It seems, that John loves Mary.*“.

Die Belegung der Tabelle in Abbildung 4.10 ist sprachspezifisch.

Parameter

Hintergrund: Die in der Tabelle angegebenen Richtungsangaben „*links*“ und „*rechts*“ sind wesentlich, weil sie unmittelbar Folgen für die Wortstellung haben. Diese Eigenschaft ist in GB typabhängig parametrisiert. Darüber hinaus kann die Belegung der Tabelle sprachspezifisch unterschiedlich gewählt werden. Beispielsweise folgt aus der Belegung für den Typ **V₄**, daß das Subjekt links vom Verb und das Objekt rechts vom Verb steht wie in „*John loves Mary.*“. Aus der Tabellenbelegung folgt die für eine Sprache typische zugrundeliegende Wortstellung. Entsprechend leicht lassen sich alternative Wortstellungen für andere Sprachen beschreiben, ohne den ganzen GB-Mechanismus neu definieren zu müssen. Kleine Anpassungen reichen aus. Definition 49 setzt eine Reihe von GB-Konzepten um, die in der sogenannten Kasustheorie formuliert werden. Dort stellt der sogenannte *Kasusfilter* ein zentrales Wohlgeformtheitskriterium dar. Dieses erklärt eine syntaktische Struktur dann für ungrammatisch, wenn nicht jeder *NP* genau ein Kasus zugewiesen wird. Kasuszuweiser sind Köpfe von *Kategorien*, die teilweise sprachspezifisch festgelegt sind, und die zu der *NP* in einer Rektionsbeziehung stehen müssen. Ein zusätzlicher *Richtungsparameter* legt fest, ob ein Kasus nach links oder rechts zugewiesen wird, was unmittelbar Folgen für die Wortstellung hat (Objekt steht links oder rechts vom Verb). Ein dritter Parameter kann schließlich verlangen, daß Kasuszuweisung nur unter *Adjazenz* erlaubt ist, die an dieser Relation beteiligten Wörter müssen dann im Satz direkt nebeneinander stehen. Wir führen den Begriff Ka-

sus hier nicht ein und drücken die in der Kasustheorie beinhalteten Restriktionen allein durch graphentheoretische Restriktionen in der Tabelle aus.

KOROLLAR 5: *Da jeder Syntaxbaum einen CP-Knoten oder IP-Knoten an der Wurzel hat, folgt aus den Nachbarschaftsprinzipien, daß jeder Syntaxbaum eine VP enthalten muß. Ein Graph, welcher diese Grundanforderungen nicht erfüllt, ist kein Syntaxbaum.*

In der GB-Literatur ist es üblich, die Optionen COMP, SPEC und ADJ auch dann als Anknüpfungspunkte anzugeben, wenn nicht von ihnen Gebrauch gemacht wird. Unsere Formalisierung erlaubt, diese dann zu ignorieren, was kompaktere Darstellungen erlaubt. Abbildung 4.11 zeigt einen minimal möglichen Syntaxbaum für den Satz „it rains“.

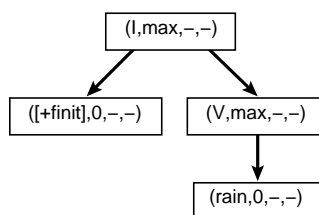


Abbildung 4.11: Ein minimaler Syntaxbaum

Hintergrund: In der klassischen GB-Literatur sieht die entsprechende minimale Basisstruktur umfangreicher aus (Abbildung 4.12): An den Knoten der maximalen Projektionen sind jeweils zwei bis drei alternative Bezeichnungen notiert. Die Notation mit dem hochgestellten Haken V , V' , V'' findet sich oft alternativ zur Indizierung mit natürlichen Zahlen.

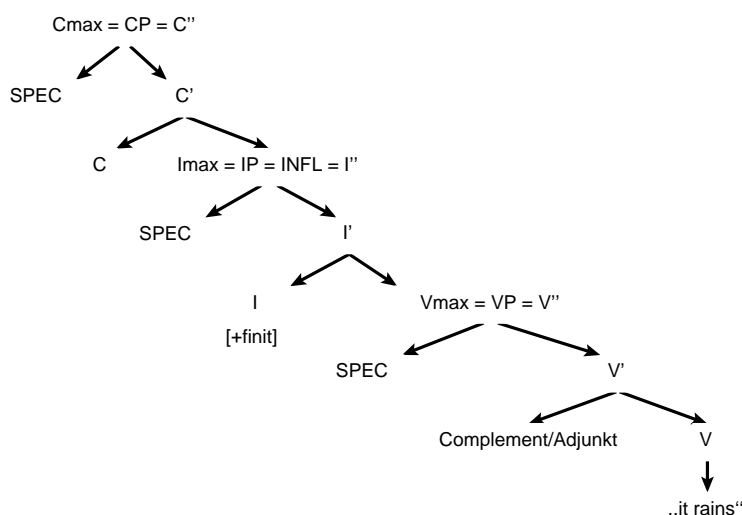


Abbildung 4.12: Minimaler Syntaxbaum in GB: Viele Abzweigungen bleiben ungenutzt.

Das „it“ erscheint im Syntaxbaum nicht als eigenständiger Knoten, was auf die Nachbarschaftsanforderungen des Verbs „rain“ zurückzuführen ist, welches vom Typ V_0 ist. Das „it“ entsteht erst im Rahmen eines Auswertungsprozesses dieser Syntaxbaumstruktur.

Hintergrund: Das „*it*“ steht bewußt nicht an einem Blatt in einer Subjektposition, GB behandelt es als ein sogenanntes *Expletivum* (auch die Bezeichnung *non-A-Ausdruck* ist in GB gebräuchlich), welches vom pronominalen „*it*“ zu unterscheiden ist, das eine θ -Rolle zugewiesen bekommt. Expletiva besetzen bei der morphologischen Auswertung des Syntaxbaums diejenige syntaktische Subjektposition lexikalisch, welche keine θ -Rolle vom Verb bekommt. In der Komplementposition führen sie stets zu ungrammatischen Sätzen. Oft stehen sie im Kontext sogenannter Wetterverben wie „*regnen*“ oder „*scheinen*“. Weitere Beispiele sind Konstruktionen wie „*es friert mich*“, „*es graust mich*“ oder „*it follows*“. Das semantisch weitgehend leere Element „*es*“ existiert im Englischen als „*it*“ oder „*there*“, im französischen als „*il*“. Das expletive „*es*“ ist vom pronominalen „*es*“ streng zu unterscheiden. Letzteres bekommt unbedingt eine Θ -Rolle zugewiesen.³⁴

Neben den notwendigen Nachbarschaftsbeziehungen gibt es noch eine Reihe von Möglichkeiten, wohlgeformte Syntaxbäume durch zusätzliche Phrasen (=Bäume) zu erweitern, die an ausgewählten Positionen eingehängt werden dürfen.

DEFINITION 50 (OPTIONALE NACHBARSCHAFTSBEZIEHUNGEN): Die optionalen Nachbarschaftsbeziehungen sprechen typabhängig eine Beschränkung aus, welche Phrasentypen (beliebig oft) an Adjunkt- oder Spezifierposition verwendet werden dürfen. Sie müssen im Gegensatz zu notwendigen Nachbarschaftsbeziehungen aber nicht zwingend realisiert werden.

Wenn eine Nachbarschaftsbeziehung optional ist, darf von dieser Option sogar beliebig oft Gebrauch gemacht werden. Beispielsweise dürfen dort, wo eine *PP* zulässig ist, unbeschränkt viele *PP*s hinzugefügt werden, ohne daß die Grammatikalität davon berührt wird.

Kopf	Specifier (links)	Adjunkt*
A	-	<i>AP</i> (links)
N	Element einer <i>NP</i> - β -Kette	<i>AP</i> (links), <i>PP</i>
V₀	-	<i>AP</i> , <i>PP</i> , Basis einer <i>CP</i> - α -Kette
V₁	-	<i>AP</i> , <i>PP</i> , Basis einer <i>CP</i> - α -Kette
V₂	-	<i>AP</i> , <i>PP</i> , Basis einer <i>CP</i> - α -Kette
V₃	-	<i>AP</i> , <i>PP</i> , Basis einer <i>CP</i> - α -Kette
V₄	-	<i>AP</i> , <i>PP</i> , Basis einer <i>CP</i> - α -Kette
V₅	-	<i>AP</i> , <i>PP</i> , Basis einer <i>CP</i> - α -Kette
C	Element einer <i>NP/CP</i> - α -Kette**	-
P	-	-

* Wenn nicht anders angegeben, zweigen die Adjunkte rechts von der Projektionslinie ab.

** Landeplatz für bewegte maximale Projektionen oder Zwischenspuren.

Abbildung 4.13: Optionale Nachbarschaftsbeziehungen

Die Richtungsanweisungen in Tabelle 4.13 sind wieder sprachspezifisch zu verstehen. Auf diese Weise wird es auch hier möglich, durch kleine Anpassungen Wortstellungen in anderen Sprachen zu modellieren, ohne das gesamte Modell umformulieren zu müssen.

Parameter

³⁴ [FF87b, 77ff], [Cul97, 101ff]

BEISPIEL 12 (NOMINALPHRASE): Abbildung 4.14 zeigt eine Nominalphrase, bei der alle Nachbarschaftsbeziehungen optional sind. Am Kopf der Phrase steht das Wort „house“.

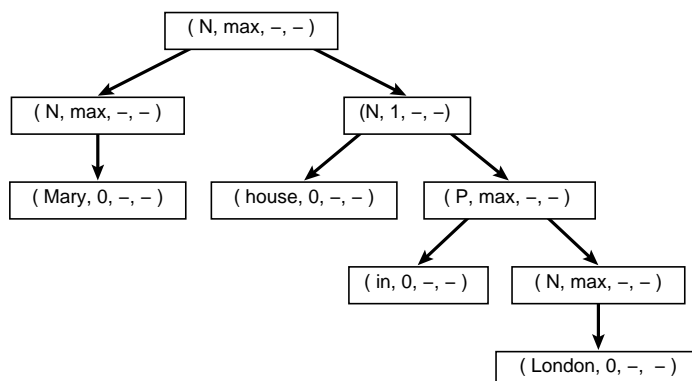


Abbildung 4.14: Nominalphrase für „Marys house in London“

Es trägt keinen Index. Rechts als Adjunkt sehen wir eine Präpositionalphrase „in London“. An der Position der Specifiers der *NP* steht eine *NP* als Genitivattribut. Daß der Knoten mit dem Bezeichner „Mary“ nach der Auswertung mit dem String „Marys“ identifiziert wird, also die passende Endung für den Genitiv angefügt wird, ist durch keine rein syntaktische Regel zu begründen. Hier spielen morphologische Prinzipien - also Regeln, die festlegen, wie der Genitiv bei einem Wort in einer Sprache sichtbar gemacht wird - eine Rolle. Auf morphologische Aspekte gehen wir in dieser Arbeit nicht ein und geben deshalb die Bezeichner an den Blättern in der Grundform an.

Wie das letzte Beispiel andeutet, wird eine *NP* in Specifier Position einer anderen *NP* als Genitivattribut realisiert. Es gibt noch eine Reihe weiterer Regeln dieser Art: Eine *NP* in der SPEC-Position einer *IP*, wird als *Subjekt* des dargestellten Satzes bezeichnet. Eine *NP* in Komplementposition einer *VP* repräsentiert das *Objekt*. Eine *CP* in Adjunkt- oder Komplementposition einer *VP* stellt einen *Nebensatz* dar. Es lassen sich also viele grammatische Begriffe allein auf der Grundlage graphentheoretischer Beziehungen zwischen Knoten definieren.

PRINZIP 5 (NACHBARSCHAFTSPRINZIP):

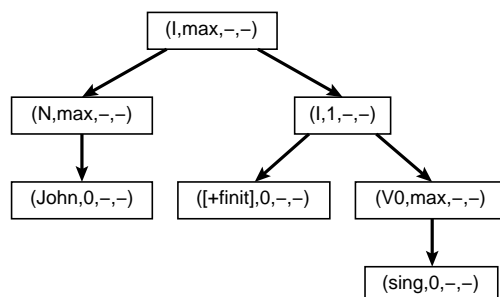
Für jeden Kopf im Syntaxbaum müssen alle notwendigen Nachbarschaftsbeziehungen realisiert sein. Eine Nachbarschaftsbeziehung im Syntaxbaum, die nicht notwendig ist, muß optional sein.

□

4.6 Übersicht: Die fünf Prinzipien in einer Kurzfassung

Damit sind alle fünf Prinzipien ausformuliert, die unser Modell definieren. Wir wiederholen sie hier in einer Kurzfassung:

1. Die Baumstruktur (nicht aber die Namen der Knoten) ist in der Sprache der Graphgrammatik GG_{X-Bar} .

Abbildung 4.15: Syntaxbaum: „*John sings.*“

2. Die Kanten sind geordnet und mit l , r oder p benannt.
3. Die Knotennamen bestehen aus Viertupeln. Die erste Komponente ist ein Bezeichner, die anderen Komponenten bestehen aus Indizes.
4. Indizes sind nur in wohldefinierten Kontexten erlaubt.
5. Die Restriktionen auf Nachbarschaftsbeziehungen zwischen Knoten müssen erfüllt werden. Darüber hinaus muß jede Nachbarschaftsbeziehung erlaubt sein.

Als Metaprinzip gilt schließlich: Jeder Syntaxbaum muß alle Prinzipien erfüllen.

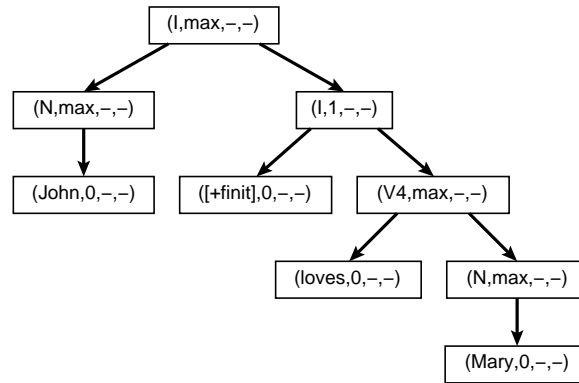
4.7 Einige Satzanalysen als Fallbeispiele

In den vorangehenden Abschnitten haben wir auf der Basis der linguistischen GB-Theorie ein formales Modell definiert, welches sich der Terminologie der Graphentheorie bedient. Dabei ist es gelungen, die vielfältigen Regularitäten von GB nach fünf Gesichtspunkten zu klassifizieren und zusammenzufassen. Um die Expressivität der resultierenden fünf Prinzipien zu beleuchten, demonstrieren wir auf den folgenden Seiten einige Satzanalysen.

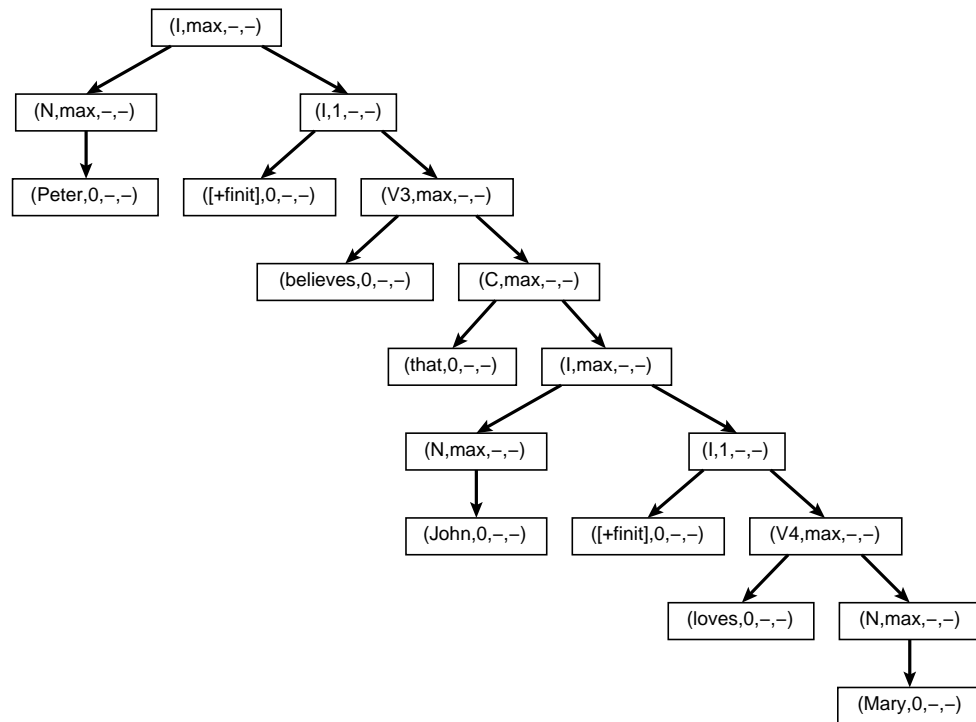
BEISPIEL 13: Der einfache Hauptsatz in Abbildung 4.15 enthält wie jeder grammatische Satz eine *IP*, an welcher GB Modusinformationen des Satzes lokalisiert. An der Position des Specifiers zum Knoten „ $([+finit], 0, -, -)$ “ steht das Subjekt *John*, an der Komplementposition die (finite) *VP*. Ein Objekt gibt es nicht und darf es beim Typ V_0 auch nicht geben.

BEISPIEL 14: Im Syntaxbaum 4.16 steht das für V_4 nötige Objekt *Mary* wie von den Nachbarschaftsanforderungen formuliert in der Komplementposition des Verbs *love*. Das Beispiel macht deutlich, wie Subjekt und Objekt alleine aus ihrer strukturellen Position im Syntaxbaum heraus identifiziert werden können. Aus den lokalen Eigenschaften der Nominalphrasen heraus kann der Unterschied dagegen nicht motiviert werden.

BEISPIEL 15: In Syntaxbaum 4.17 ist ein Nebensatz als Komplement des Verbs realisiert. Würde der Nebensatz fehlen, wäre die Nachbarschaftsanforderung für „*believes*“ verletzt. Deshalb wäre auch das Satzfragment „*John believes*“ ungrammatisch. Würde

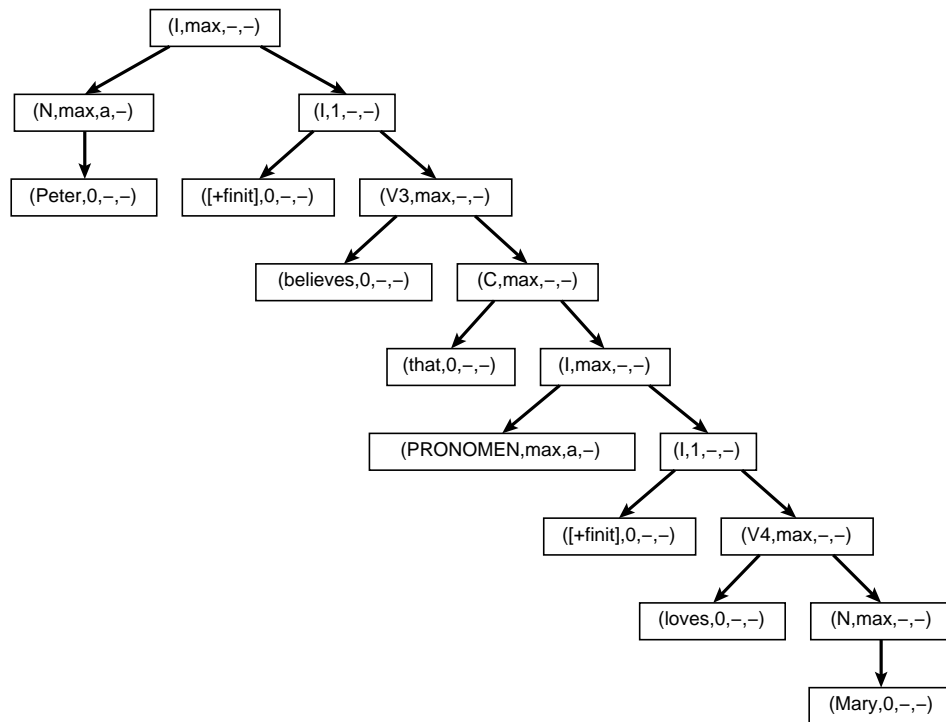
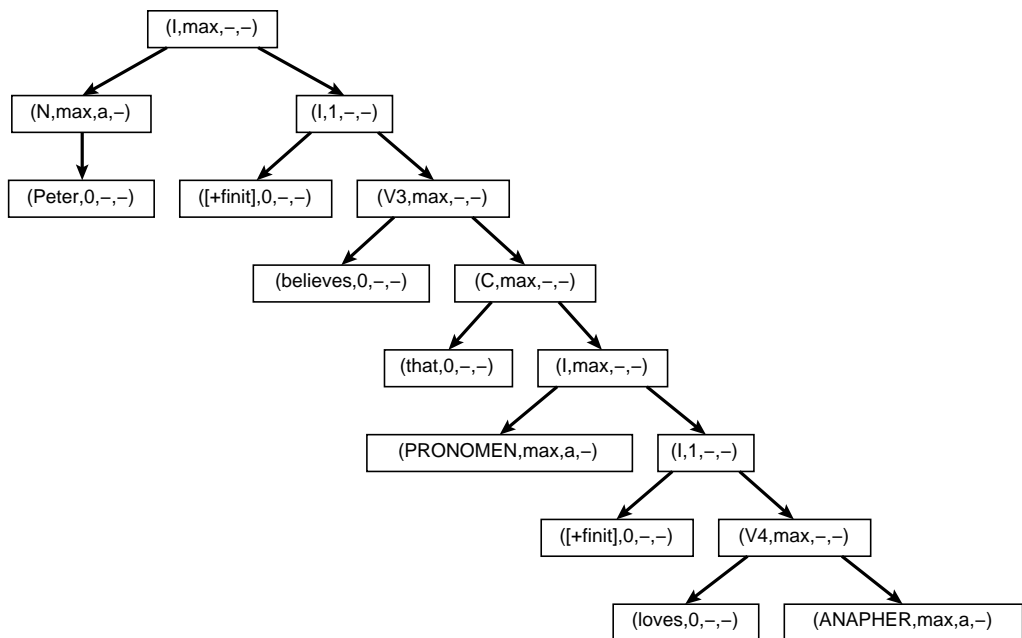
Abbildung 4.16: Syntaxbaum: „*John loves Mary.*“

im Nebensatz das Verb *love* durch *believe* ersetzt, könnte entsprechend rekursiv ein beliebig langer Satz konstruiert werden. Der entsprechende Syntaxbaum wäre dann stark ‘rechtslastig’.

Abbildung 4.17: Syntaxbaum: „*Peter believes that John loves Mary.*“

BEISPIEL 16: Das Pronomen „*he*“ im Nebensatz von Abbildung 4.18 ist koreferent zu *Peter* und bildet mit diesem eine β -Kette. Dieses Pronomen ist innerhalb seines Untersyntaxgraphen nicht gebunden, sondern frei, wie es in GB die Bindungstheorie erfordert.

BEISPIEL 17: In Abbildung 4.19 enthält der Untersyntaxgraph zusätzlich eine (durch das Pronomen *he*) gebundene Anapher *himself*. Die β -Kette enthält jetzt drei Elemente:

Abbildung 4.18: Syntaxbaum: „ $Peter_\alpha$ believes that he_α loves Mary.“Abbildung 4.19: Syntaxbaum: „ $Peter_\alpha$ believes that he_α loves himself $_\alpha$.“

Neben der Anapher sind dies ein R-Ausdruck (dieser ist frei), ein Pronomen (dieses ist frei in seinem Untersyntaxgraphen). Für alle drei sind damit die erforderlichen bindungstheoretischen Anforderungen erfüllt.

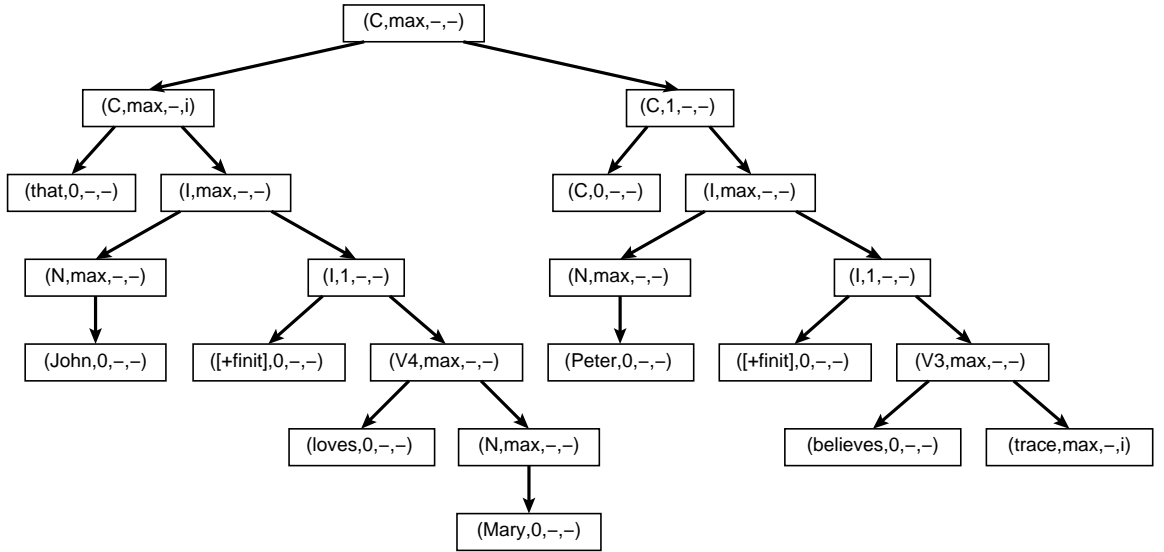


Abbildung 4.20: Syntaxbaum: „*(That John loves Mary)_i, Peter believes trace_i*.“

BEISPIEL 18: Der Nebensatz in Syntaxbaum 4.20 ist an den Satzanfang bewegt (topikalisiert) worden. Landeplatz ist die Specifierposition der dominierenden *CP*. Die Bewegung hinterläßt eine koindizierte Spur (trace). Die Subjazenbedingung ist eingehalten: Der Bewegungsschritt passiert höchstens einen Grenzknoten. Eine Zwischenlandung ist deshalb nicht erforderlich.

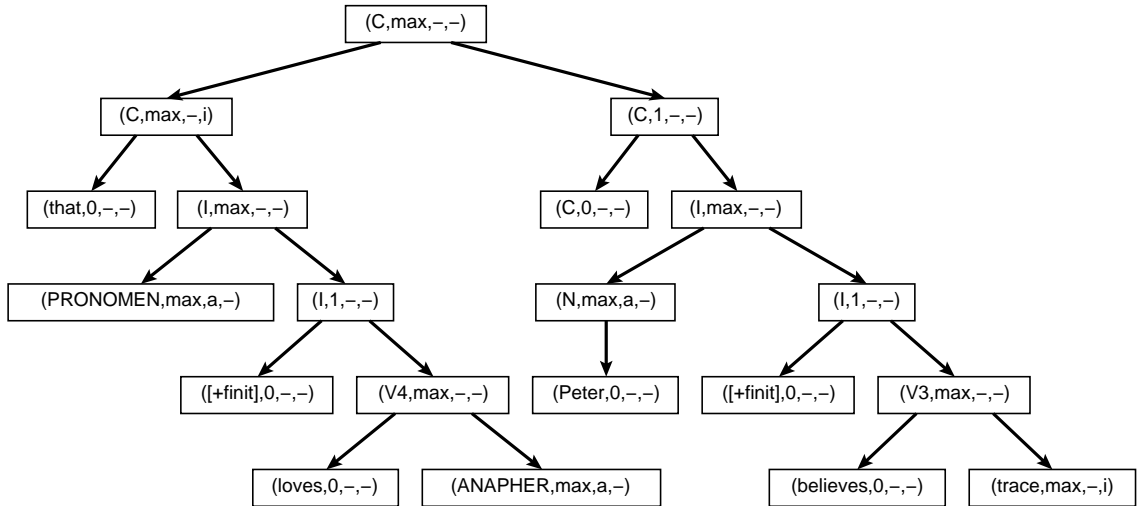


Abbildung 4.21: Syntaxbaum: „*(That he_α loves himself_α)_i, Peter_α believes trace_i*.“

BEISPIEL 19: In Syntaxbaum 4.21 sind die Phänomene der vorangehenden Beispiele kombiniert umgesetzt. Es wird in zwei Schritten topikalisiert. Der Syntaxbaum eine β -Kette und eine α -Kette. Trotz der Bewegung sind die Bindungsprinzipien für R-Ausdruck,

Pronomen und Anaphern nicht verletzt. Alle Anforderungen an eine Kette sind nach wie vor erfüllt.

BEISPIEL 20: Abbildung 4.22 schließlich zeigt einen iterierten Bewegungsprozeß. Es wird in zwei Schritten topikalisiert. Durch die von der Subjazenbedingung erforderte Zwischenlandung wird verhindert, daß zusätzliche illegale Bewegungen an diese sonst freie Landestelle stattfinden. Für weitere Bewegungen steht dieser Landeplatz also nicht mehr zur Verfügung. Die mit *move α* beschrifteten Linien stellen den Bewegungsablauf dar, sind aber *nicht* Bestandteil des Graphen.

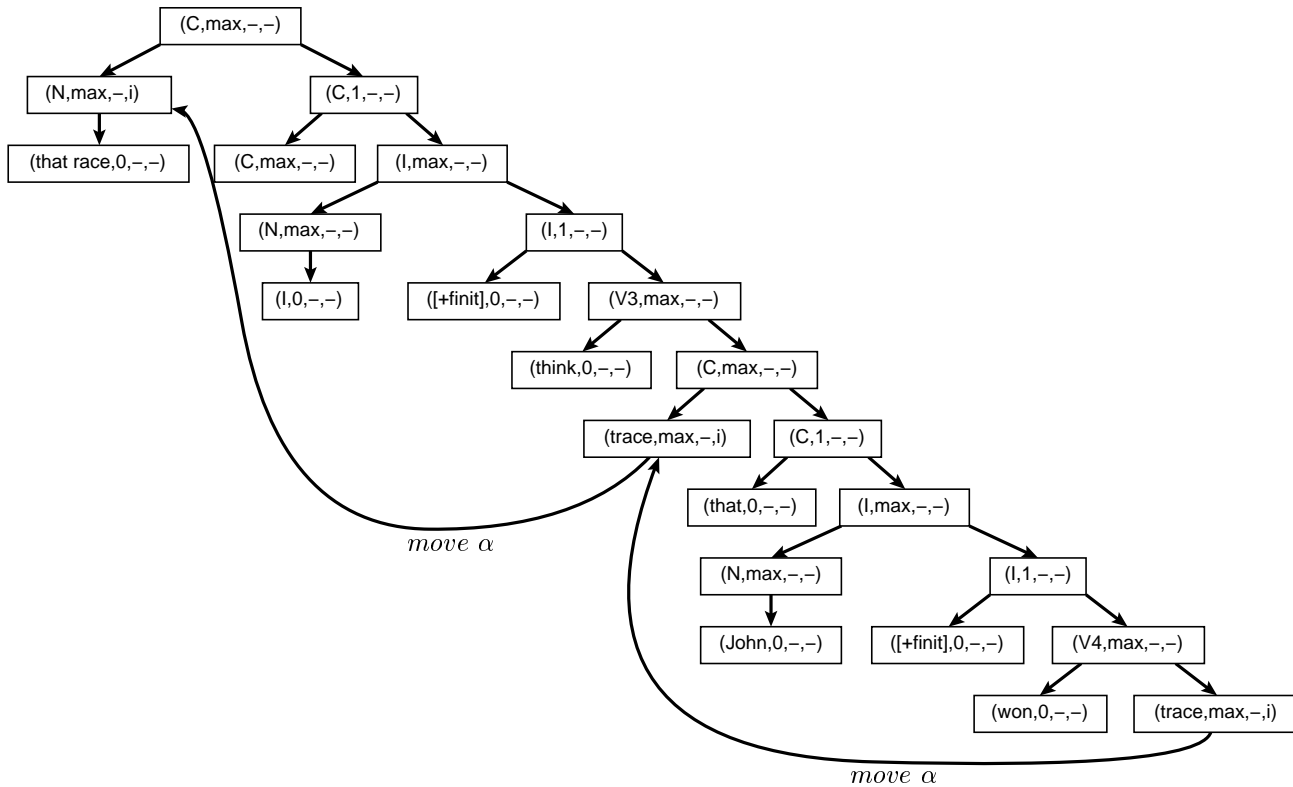


Abbildung 4.22: Syntaxbaum: „ $(That\ race)_i$, $I\ think\ trace_i\ that\ John\ won\ trace_i$ “

4.8 Ungrammatische Sätze

Die eigentliche Leistungsfähigkeit des GB-Modells wird dann besonders deutlich, wenn man es zur Erklärung der Ungrammatikalität von Sätzen heranzieht.

Dazu soll zunächst das Beispiel *,*John, Peter believes, that loves Mary*“ näher betrachtet werden. Daß der Satz ungrammatisch ist, erschließt sich auch dem *nicht-native speaker* unmittelbar, obwohl er dennoch nicht unverständlich ist. Die Ursache dafür ist aber nur schwer zu ergründen. Dem Nicht-Linguisten bleibt in der Regel nur, sich auf das individuelle (eigentlich kollektive) Sprachgefühl des fiktiven *ideal speaker* zu berufen.

Um weitergehende Gründe für die Ungrammatikalität anzubringen, verbleibt zu erklären, woran ein als unzulässig und falsch empfundener Satz scheitert. GB vermag auf diese Frage eine Antwort zu geben, indem nicht über die oberflächlich sichtbare Satzgestalt (Zeichenkette), sondern über seine innere Struktur (Syntaxbaum) gesprochen wird. Ungrammatisch ist ein Satz dann, wenn kein Syntaxbaum existiert, welcher den betrachteten Satz als *yield* besitzt. Jeder Kandidat für einen Syntaxbaum kann anhand der Prinzipien als ungrammatisch verworfen werden.

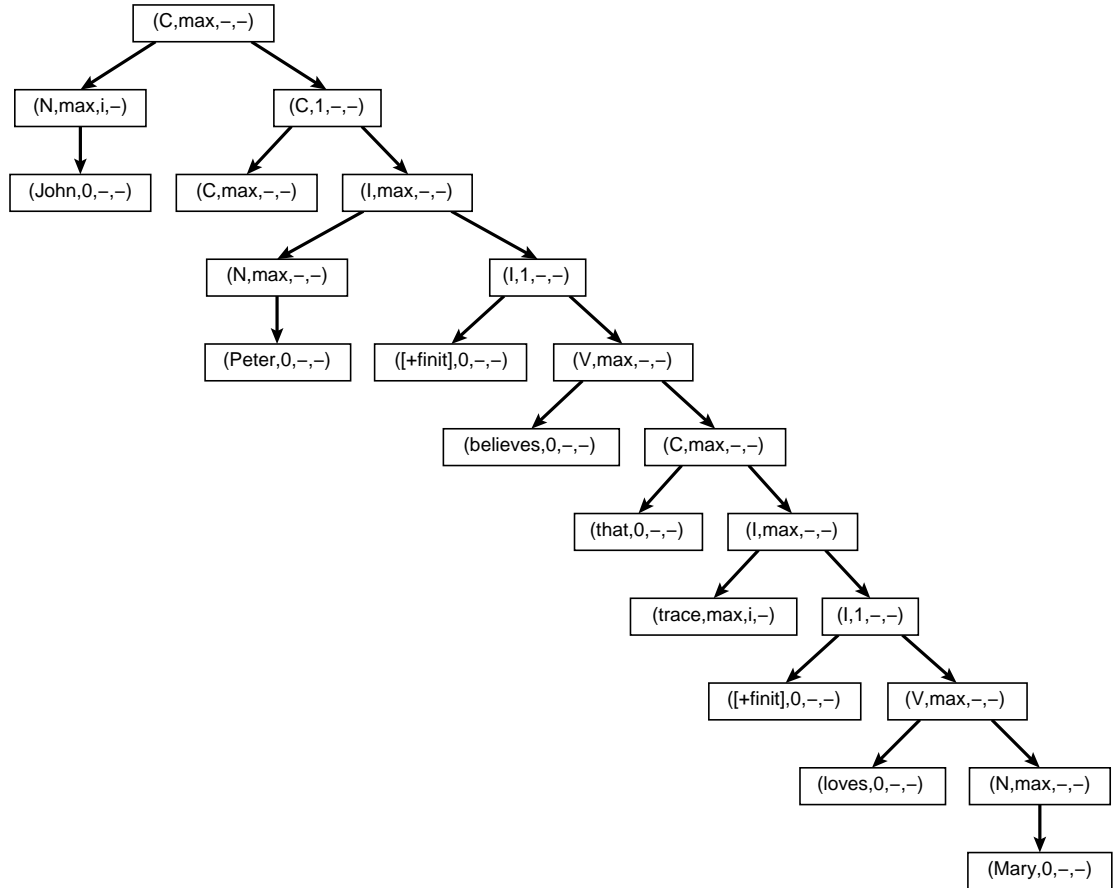


Abbildung 4.23: Eine versuchte Analyse für „*John, Peter believes, that loves Mary.*“

Die Satzanalyse im Beispiel 4.23 unterstellt, daß *John* aus seiner ursprünglichen Position topikalisiert und an den Satzanfang gestellt wurde. Als Resultat ergäbe sich genau die betrachtete Satzstellung. Eine genauere Betrachtung zeigt aber, daß es sich bei diesem Graphen gar nicht um einen Syntaxbaum handelt, weil er gegen das Prinzip 4 verstößt: Als Spur muß der Knoten $(trace, max, i, 0)$ Bestandteil einer α -Kette sein. Dies ist jedoch nicht der Fall, weil die Basis der Kette nicht streng regiert wird, wie es das Prinzip fordert.

Hintergrund: In früheren Versionen von GB wurde die Ungrammatikalität mit dem *that*-Trace Effekt begründet, welcher das Vorkommen einer Spur nach einem „*that*“ verbietet. Später ergab sich diese Regel als Spezialfall der hier eingeführten allgemeineren Sichtweise.³⁵

³⁵ [FF87b, 179ff], [Cul97, 197ff]

Abbildung 4.24 zeigt einen Vorschlag für die Analyse eines Satzes, bei dem die Prinzipien für die Konstruktion von β -Ketten verletzt werden. Eine Anapher wie „*himself*“ muß im Untersyntaxgraphen gebunden werden, ist hier aber frei.

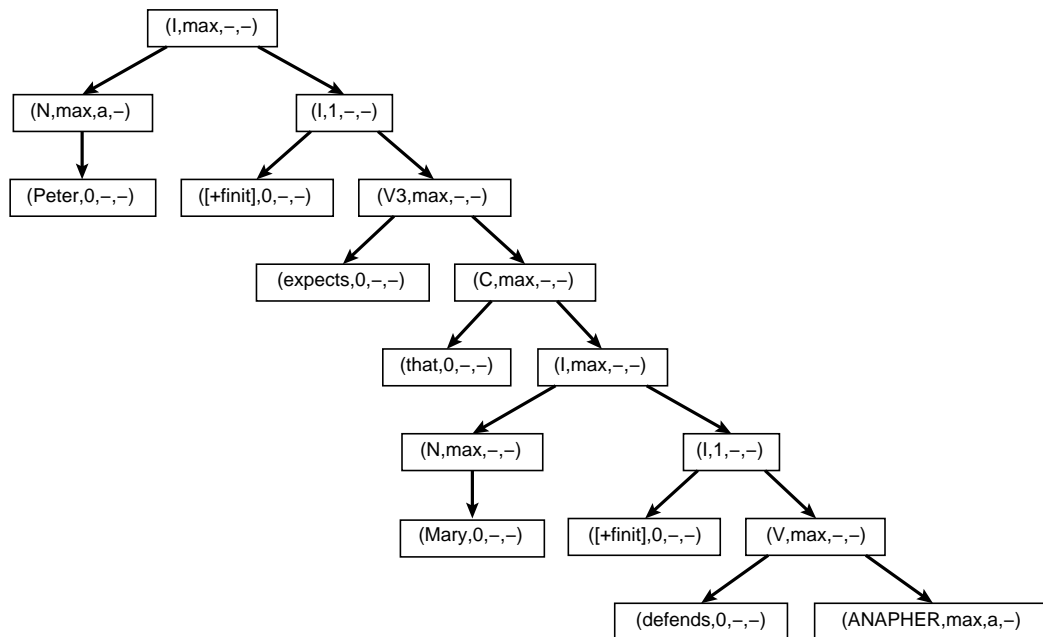


Abbildung 4.24: Versuchte Analyse: *„ $John_\alpha$ expects, that Mary defends himself $_\alpha$.“

Da es für den Satz keine Analyse gibt, in der alle Prinzipien erfüllt sind, ist er ungrammatisch. Daß er dem Leser dennoch nicht unsinnig erscheint, ist eine Motivation dafür, daß die in der Linguistik übliche Unterscheidung von Syntax und Semantik sinnvoll ist.

Auch die folgenden Sätze sind so gewählt, daß sich dem Leser ein Sinn erschließt, obwohl sie ungrammatisch sind. Wir erläutern die Gründe für die Ungrammatikalität ohne Syntaxgraph (weil kein solcher existiert).

1. *„John loves Mary smiles.“

Die Nachbarschaftsanforderungen sind nicht erfüllt. Ausgehend von dem korrekten Satz „John loves Mary“ gibt es keine Möglichkeit, den Syntaxgraphen um eine Struktur für „smiles“ zu erweitern. Umgekehrt kann die korrekte Struktur für „Mary smiles“ in keiner zulässigen Weise dem Fragment „John loves“ untergeordnet werden.

2. *„John hit herself“

Eine Anapher wie „herself“ darf nur in der Umgebung einer β -Kette vorkommen. Eine solche läßt sich in diesem Satz aber nicht (re-)konstruieren. „herself“ muß entweder durch „John“ gebunden sein, was wegen des unterschiedlichen Geschlechts nicht zulässig ist, oder durch „hit“, was wegen der Wortart unmöglich ist.

3. *„The mouse the cat catches“

Zwar sind hier für das Verb Kandidaten für Subjekt und Objekt vorhanden, allerdings kollidiert die Wortstellung mit den Nachbarschaftsanforderungen.

Die angeführten Beispiele für Analysen ungrammatischer Sätze machen den Filtermechanismus der Prinzipien deutlich. In Kapitel 5 wird dieser repräsentationelle Ansatz zu einem derivationellen Verfahren weiterentwickelt.

4.9 Zusammenfassung und Bewertung der Vorgehensweise

Das vorangehende Kapitel demonstriert eine Möglichkeit, unter Verwendung graphentheoretischer Terminologie einen Syntaxbaum für ein Fragment der englischen Sprache zu spezifizieren, wie er im Rahmen von GB eine zentrale Rolle spielt. Abbildung 4.25 illustriert noch einmal die Vorgehensweise.

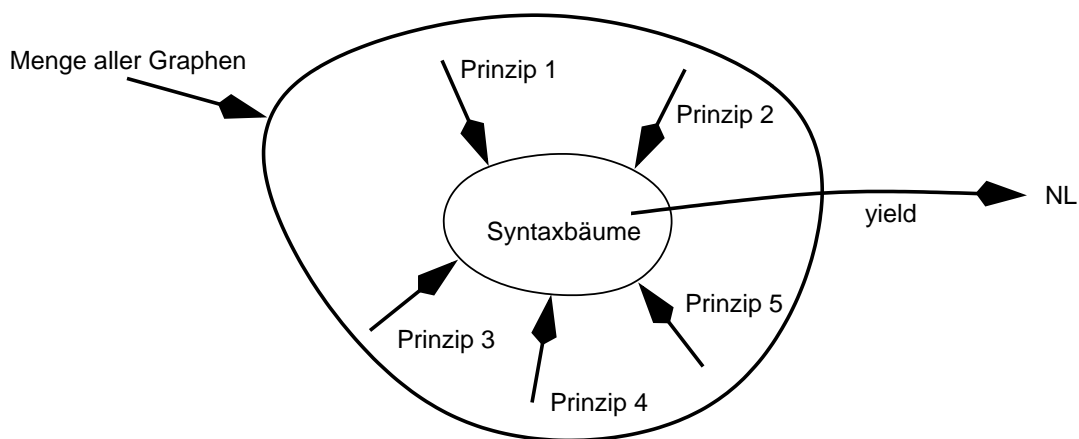


Abbildung 4.25: Prinzipien für Graphen beschreiben Zeichenketten.

4.9.1 Selbstbeschränkung und Auswahlkriterien

Die aus unserer Beschreibung resultierenden Syntaxbäume setzen bei weitem nicht alle Prinzipien um, die GB einführt, sondern nur ein Fragment davon. Auch sind durch die Schwerpunktsetzung auf graphentheoretische Schreibweisen Darstellungen bedingt, die sich von denen der GB-Literatur unterscheiden. Davon werden wir aber sehr profitieren, wenn wir uns in Kapitel 5 formale Betrachtungen von Syntaxbäumen zuwenden.

Es gibt also sehr wohl englische Sätze, die grammatisch sind, die aber mit den hier bereitgestellten Mitteln nicht modelliert werden können. Zu den nicht modellierbaren Sätzen zählen zum Beispiel:

1. Fragesätze wie „*Who loves Mary?*“
2. Befehlssätze wie „*Sit down!*“

3. Relativsätze wie „*The man, who loves Mary, sings a song.*“
4. Sätze mit Junktoren „*John loves Mary and Mary loves John.*“
5. Sätze mit Quantoren: „*Every man loves a woman.*“
6. ECM-Konstruktionen: „*John seems to love Mary.*“

Trotz dieser Beschränkungen ist die Auswahl der bei der Modellierung umgesetzten GB-Prinzipien sinnvoll: Entscheidend ist, daß unser Modell bei allen Restriktionen expressiv genug bleibt, um an die beweisbaren Grenzen zu stoßen, die wir im nächsten Kapitel aufzeigen werden. Bezüglich dieser Grenzen erweist sich die Menge der modellierten Phänomene als umfangreich genug. Die Modellierung der aufgezählten ausstehenden Phänomene würde in erster Linie die Baumstruktur der Syntaxbäume verfeinern, aber keine zusätzlichen Querbeziehungen einführen. Denen gilt aber (in formaler Hinsicht) unser primäres Interesse: Nicht die zugrundeliegende Baumstruktur macht Syntaxbäume zu einer Herausforderung, sondern die darüberliegenden Querbeziehungen.

4.9.2 Die fünf Prinzipien beschreiben nicht zu viel

Bei allen Einschränkungen sollte es nicht passieren, daß unser Modell eine Satzstruktur als grammatisch markiert, für die dies nicht der Fall ist. Diese Eigenschaft läßt sich nicht formal beweisen, weil der Beschreibungsgegenstand (die Menge der grammatischen Sätze in *NL*) nicht formal gegeben ist.

Statt dessen verweisen wir darauf, daß wir mit GB ein wohluntersuchtes linguistisches Modell zur Grundlage unserer Prinzipien gemacht haben. Sollte sich in Einzelfällen zeigen, daß durch die fünf Prinzipien zu viele Sätze – also auch ungrammatische Konstruktionen – als grammatisch zugelassen werden, dann wäre dies ein Indiz für konzeptionelle Schwächen von GB hinsichtlich des Anspruchs an die explanatorische Adäquatheit.

Tatsächlich werden wir im nächsten Kapitel im Zusammenhang mit Separatoren in diesen Grenzbereich zwischen grammatisch und ungrammatisch vorstoßen.

4.9.3 Die Prinzipien sind konsistent formuliert

Weiterhin müssen wir uns die Frage stellen, inwiefern unser Modell frei von Widersprüchen ist. Bezüglich der Konsistenz ist also sicherzustellen, daß die Prinzipien und Regularitäten in dem Sinne zusammenpassen, daß ein Satz nicht auf eine Weise als grammatisch und auf eine andere Weise als ungrammatisch ausgewiesen wird. Diese Gefahr kann aber durch das gewählte Definitionsprinzip ausgeschlossen werden: Eine syntaktische Struktur wird im repräsentationellen Mechanismus nur dann als grammatisch ausgewiesen, wenn sie *alle* Prinzipien erfüllt, wobei jedes Prinzip wie ein Filter wirkt. Wenn jedes Prinzip für sich genommen widerspruchsfrei ist, dann gilt dies automatisch auch für den gesamten Mechanismus. Damit müssen wir lediglich prüfen, daß wir nicht eine leere Menge spezifiziert haben. Die angeführten Beispiele in Abschnitt 4.7 (die den Prinzipien genügen) beweisen, daß dies nicht der Fall ist.

4.9.4 Redundanz im Modell kann nicht ausgeschlossen werden

Trotz der Beschränkung auf fünf Prinzipien ist nicht klar, inwieweit unsere Darstellung frei von Redundanzen ist, zumal wir sie nicht von Grund auf neu entworfen haben, sondern uns bei der Konzeption an einem bereits vorhandenen Modell orientieren. Es ist denkbar, daß sich eine Reihe von Prinzipien zugunsten allgemeinerer und kompakterer Regeln zusammenfassen lassen. Der weiteren Untersuchung unseres Modells steht dies jedoch nicht im Wege. Bezüglich GB hat Chomsky selber mit der Initiierung des „*minimalist program*“ das Ziel formuliert, GB auf weniger und dafür allgemeinere Prinzipien zurückzuführen. Den repräsentationellen Ansatz selbst stellt er dabei aber nicht grundsätzlich in Frage.

4.9.5 Weiteres Vorgehen

Genau hier – und darin zielen wir in eine andere Richtung als Chomsky – suchen wir nach einer alternativen Lösung. Es geht uns darum, einen derivationellen Mechanismus zu entwerfen, mit welchem sich eine Menge NL approximieren läßt. Davon erwarten wir eine kompaktere Darstellung von syntaktischen Strukturen und darüber hinaus Ansätze für effiziente Algorithmen.

The need for a formal grammatical description of specific languages arises in various scientific disciplines. This makes formal language theory a really interdisciplinary area of science.

SALOMAA IN [SAL81]

5 Formale Analyse syntaktischer Strukturen

Die Ergebnisse des vorangehenden Kapitels liefern eine formale Grundlage, um linguistische Fragestellungen aus dem Bereich der Syntax mit Methoden aus der Theorie der formalen Sprachen zu untersuchen. Nun kann über wohldefinierte Zeichenketten und Graphenstrukturen diskutiert werden.

5.1 Vom Syntaxbaum zum Syntaxgraph

Zunächst eliminieren wir einige aus graphentheoretischer Sicht störende Eigenschaften von Syntaxbäumen. Diese lassen sich durch wenige Modifikationen beheben. Als Ergebnis ergibt sich aus einem Syntaxbaum ein *Syntaxgraph*. Da die in Kapitel 4 eingeführten Begriffe für eine Baumstruktur definiert worden sind, ist nicht klar, daß sie auf der modifizierten Struktur Syntaxgraph immer noch sinnvoll angewendet werden können. Tatsächlich haben wir diese Definitionen aber so formuliert, daß sie nach wie vor wohldefiniert sind. Vereinzelte Ausnahmen und Erweiterungen zählen wir explizit auf. Die entsprechenden Stellen sind mit dem Stichwort *Modifikation* gekennzeichnet.

5.1.1 Problem – Der Baum hat zu viele Knotenbezeichner

Der in Kapitel 4 definierte Syntaxbaum besitzt unter anderem folgende Eigenschaften:

1. Unbeschränkt viele Knoten können gemeinsame Indizes besitzen.
2. Die Indexmengen zur Knotenbenennung sind unbeschränkt.

Beide Eigenschaften führen zu Schwierigkeiten, die eng miteinander zusammenhängen und sich auf ähnliche Weise lösen lassen.¹ In der Graphentheorie ist es üblich, endliche Alphabete zur Knoten- und Kantenbenennung zu verwenden. Zwar sind unbeschränkte Alphabete – ob zur Knoten- oder Kantenbenennung verwendet – im Zusammenhang mit formalen Sprachen nicht grundsätzlich verboten. Sie machen aber die Anwendung vieler gut erforschter Mechanismen unmöglich, denn diese setzen in der Regel beschränkte Alphabete voraus.

5.1.2 Lösung – Mit zusätzlichen Kanten reicht ein endliches Alphabet

Eine formale Analyse setzt ein Graphenmodell voraus, welches mit endlichen Alphabeten auskommt. Eine künstliche Beschränkung der Indexalphabete wäre eine unbefriedigende Lösung, weil damit die zugrundeliegenden Mechanismen nicht mehr sichtbar wären.² Tatsächlich führt GB empirische Argumente dafür an, daß diese Mengen nicht beschränkt sein dürfen. Eine Reihe von Sätzen, die sehr wohl grammatisch sind (wenn auch unter Umständen nicht praxisrelevant), könnten sonst nicht modelliert werden.

Zur Lösung bieten sich zwei Eigenschaften des Graphenmodells an: Während wir mit den Alphabetsymbolen „sparsam“ umgehen müssen, bekommen wir einerseits beliebig viele Kanten und andererseits einen beliebig hohen Verzweigungsgrad quasi „umsonst“. Wie wir zeigen werden, ist es möglich, mit Hilfe dieser Konzepte auf die unendlichen Alphabete \mathcal{B} , \mathcal{I} und \mathcal{J} zu verzichten, ohne an Expressivität zu verlieren.

5.1.2.1 Elimination der Alphabete \mathcal{I} und \mathcal{J}

Wie bei deren Definition erwähnt, bilden die Knoten einer Kette eine Äquivalenzklasse. Verschiedene Möglichkeiten bieten sich an, diesen Sachverhalt graphentheoretisch auszudrücken. Von den folgenden beiden Lösungsansätzen werden wir den ersten wieder verwerfen:

1. Ansatz: „*Identifiziere Knoten mit gleichem Index.*“ Da die Länge von Ketten nicht beschränkt ist, könnte der Grad eines Knotens beliebig groß werden. Wenn in einem erweiterten Modell auch benachbarte Knoten in einer Kette enthalten sein dürften, würde dieser Lösungsansatz zudem ein Graphenmodell mit Mehrfachkanten benötigen. Für die Modellierbarkeit mit Graphgrammatiken sind das problematische Voraussetzungen. Wir können diese aber mit folgender alternativer Strategie umgehen:
2. Ansatz: „*Verbinde Knoten mit gleichem Index durch eine Kante.*“ Die neuen Kanten verbinden Knoten auf der Basis der Ordnung, die auf Kettenknoten definiert

¹ Zur Erinnerung: Alphabet \mathcal{I} : α -Ketten können beliebig lang sein, es kann *beliebig viele* α -Ketten geben. Alphabet \mathcal{J} : β -Ketten können beliebig viele Knoten beinhalten, es kann *beliebig viele* β -Ketten geben. Alphabet \mathcal{B} : *Rekursion kann eine Phrase beliebig groß machen*, die Länge der Projektionslinie ist nicht beschränkt.

² Als Analogie betrachte folgende Situation: Wird die Sprache $a^n b^n$ nur für den Fall $n \leq k \in \mathbb{N}$ betrachtet, dann ist diese nicht nur kontextfrei, sondern sogar regulär.

ist. Der maximale Knotengrad bleibt dann beschränkt und Mehrfachkanten können nicht entstehen. Diese Strategie erweist sich als erfolgreich:

Das Alphabet zur Kantenbenennung erhält neben „l“, „r“ und „p“ ein viertes Symbol, die Kantenmarkierungsfunktion wird entsprechend angepaßt:

MODIFIKATION 1 (QUERKANTE): Bei allen Ketten im Syntaxbaum werden die Knoten verbunden mit dem jeweils unmittelbaren Vorgänger in der Kette. Die verbindende Kante wird mit „q“ bezeichnet, was für *Quer*kante steht. Die Kantenmarkierungsfunktion wird entsprechend umdefiniert zu $m_E : E \rightarrow \{l, p, r, q\}$. Die Menge aller Querkanten wird bezeichnet mit Q . Für die Kantenmenge $E(g)$ eines Syntaxgraphen $g \in \mathcal{SG}$ gilt nun: $E = L \cup R \cup P \cup Q$.³

□

Im Falle einer linguistischen Transformation (*move* α) einer Phrase zeigt die Querkante die Bewegungsrichtung an. Bezüglich der Bindungstheorie weisen die Querkanten von einem Pronomen oder einer Anapher weg in Richtung zu dem dazugehörigen R-Ausdruck hin.

DEFINITION 51 (REFERENZKANTE, INDEXKANTE): *Eine Querkante, die eine Koreferenz ausdrückt, heißt Referenzkante. Wenn sie eine Koindizierung ausdrückt, dann heißt sie Indexkante. Wenn nötig, unterscheiden wir diese mit den Namen q_r und q_i .*

Querkanten überlagern die zugrundeliegende Baumstruktur. Die vormaligen Referenzen einer α -Kette oder β -Kette sind nach dem Einfügen von Querkanten keine Blätter mehr. Die Struktur des Graphen verändert sich dahingehend grundlegend, als daß die Baumstruktur nun nicht mehr garantiert ist.

DEFINITION 52 (KETTE): *Aufeinanderfolgende Querkanten definieren eine Folge von Knoten im Syntaxbaum. Der so entstehende zusammenhängende Teilgraph heißt Kette und ersetzt den Kettenbegriff aus Kapitel 4. Die Indizes zur Auszeichnung zusammengehörender Knoten werde dann nicht mehr benötigt. Die übrigen Ketteneigenschaften bleiben sinngemäß erhalten.*

Zwei Ketten haben nie einen Knoten gemeinsam, sie sind in diesem Sinne *disjunkt*. Zur Unterscheidung zweier Ketten ist dann kein differenzierendes Indexsymbol mehr nötig. Deshalb werden mit der Einführung von Querkanten die unendlichen Alphabete \mathcal{I} und \mathcal{J} überflüssig.

5.1.2.2 Elimination der Alphabets \mathcal{B}

Auch das Alphabet für die zweite Komponente \mathcal{B} enthält unendlich viele Elemente. Zwei alternative Lösungen bieten sich an, um auch dieses Problem zu lösen:

³ Zur Erinnerung: Es stehen die Symbole für *Linkskanten*, *Rechtskanten*, *Projektionskanten*.

1. Als erste Variante liegt es nahe, die Beschränkung aufzugeben, daß nur binäre Verzweigungen zugelassen sind. Statt dessen können unbeschränkte Verzweigungen das Nebeneinander gleichberechtigter Teilbäume modellieren, indem eine Projektionslinie zu einem Knoten zusammengefaßt wird, wodurch in der Summe weniger Knoten und Kanten benötigt werden:

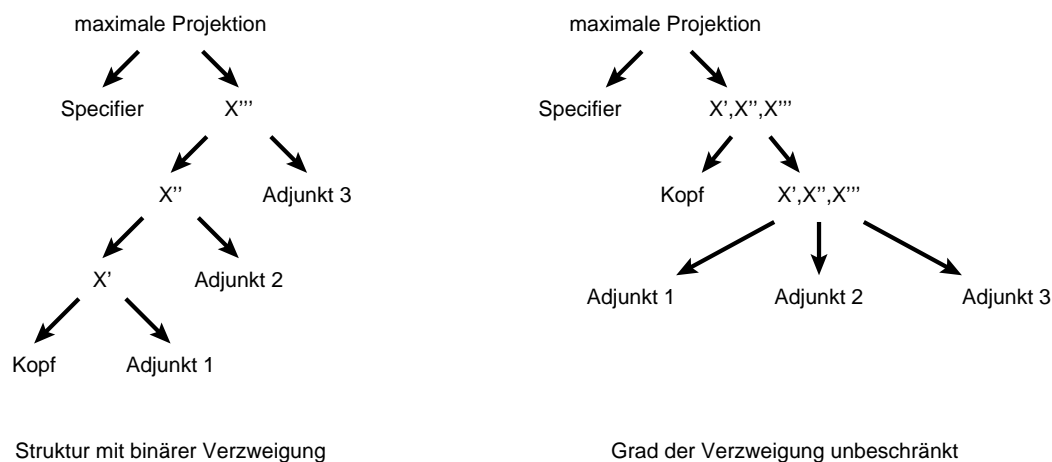


Abbildung 5.1: Eine vormals schmale und hohe Struktur wird flach und breit

Der in Abbildung 5.1 illustrierte Lösungsansatz bringt zwei Nachteile mit sich: (i) Für eine formale Betrachtung ist es nützlich, einen beschränkten Verzweigungsgrad vorauszusetzen. Außerdem ist (ii) bei einer mehrfachen Verzweigung explizit sicherzustellen, daß die Ordnungsinformation der Kanten nicht verlorengeht. Mit einem endlichen Kantenalphabet bei unbeschränktem Verzweigungsgrad ist das *nicht* zu lösen, auch wenn die links-rechts-Ordnung in der Zeichnung einen anderen Sachverhalt suggeriert. Aus diesem Grund verwerfen wir diesen Ansatz.

2. Eine Analyse der Rolle, welche die Menge \mathcal{B} in GB spielt, führt zu einer einfacheren Lösung. Im Gegensatz zu den beiden Indexalphabeten \mathcal{I} und \mathcal{J} trägt das Alphabet \mathcal{B} keine Information, die nicht auch strukturell im Graphen enthalten ist. Da Köpfe als Blätter eindeutig bestimmbar sind, können die Ebenennummern bei Bedarf ausgerechnet werden. Sie sind redundant und können deshalb im Graphenmodell ignoriert werden.

Aus linguistischer Sicht ist die zweite Lösung auch deshalb zu bevorzugen, weil sie sich an der in der Literatur üblichen Notation orientiert, die diese Art der Verzweigung über die Analyse von Konstituentenstrukturen motiviert.

5.1.2.3 Vereinfachung des Knotenalphabets

In Folge der vorangehenden Betrachtungen erweisen sich für das Namenstupel (a, b, i, j) die letzten drei Komponenten als redundant, weshalb wir die Namensgebung vereinfachen können.

MODIFIKATION 2 (VEREINFACHUNG DER KNOTENBEZEICHNER): Der Name eines Knotens v im Syntaxgraph wird gegenüber dem Namen eines Knotens w im Syntaxbaum auf den Bezeichner beschränkt. Die Indizes entfallen: $label(v) := \pi_{4,1}(label(w))$. Die Knotenbeschriftungsfunktion aus Prinzip 3 wird dann umdefiniert zu $m_V : V \rightarrow (\mathcal{T} \cup \mathcal{L} \cup \mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_3)$. □

Man beachte, daß durch diese Modifikation die Information verlorengeht, welcher Index für die Elemente einer Kette verwendet wurde. Lediglich die Information darüber, welche Knoten zu einer Äquivalenzklasse gehören, bleibt erhalten. Für die Mächtigkeit des Modells ist diese Beschränkung aber irrelevant, denn auch GB macht in vergleichbaren Fällen keinen Unterschied: Die Sätze „ $John_i$ tells $Mary_j$, that he_i loves her_j .“ und „ $John_j$ tells $Mary_i$, that he_j loves her_i .“ werden dort – was sinnvoll ist – gleich behandelt und analysiert.

5.1.3 Weitere Begriffe und Definitionen

DEFINITION 53 (GB-KOMPATIBEL): Ein Graph der alle fünf Prinzipien unter Berücksichtigung der Modifikationen aus Abschnitt 5.1.2 (keine unendlichen Alphabete und Querkanten statt Indizes) erfüllt, heißt GB-kompatibel.

DEFINITION 54 (SYNTAXGRAPH, SG): Ein GB-kompatibler Graph heißt Syntaxgraph. Die Menge aller Syntaxgraphen bezeichnen wir mit SG .

DEFINITION 55 (GERÜST): Jeder Syntaxgraph besitzt eine unterliegende Baumstruktur, die dem Syntaxbaum aus Kapitel 4 entspricht. Diese heißt Gerüst.

Im Syntaxbaum wird die repräsentierte Wortstellung durch eine Funktion *yield* bestimmt, welche die Blätter gemäß einer *inorder*-Strategie besucht. Es ist nun sicherzustellen, daß dies auch ohne Baumstruktur im Syntaxgraphen funktioniert. Tatsächlich braucht die *yield*-Funktion nicht umdefiniert zu werden, weil die Querkanten für die Wortstellung irrelevant sind, also beim systematischen Durchlaufen der Struktur ignoriert werden können.

Bezüglich der Wortstellung ist der Anpassungsaufwand an Syntaxgraphen also minimal.

5.1.4 Beispiele für die Transformation Syntaxbaum \leftrightarrow Syntaxgraph

An zwei Beispielen soll demonstriert werden, wie sich die Modifikationen auf die Darstellung der Graphen auswirkt. Dazu betrachten wir noch einmal Sätze, für die wir schon Beispiele kennengelernt haben. Die linguistischen Hintergründe zu diesen Sätzen wiederholen wir nur kompakt und verweisen für Details auf Kapitel 4:

BEISPIEL 21: In dem Satz „(That race)_i I think trace_i that John won trace_i“ wird die Phrase „(that race)“ unter Einhaltung der Subjazenrestriktionen in zwei Schritten an den Satzanfang bewegt. Dieser Prozeß wird auch als Topikalisierung bezeichnet. Abbildung 5.2 zeigt den dazugehörigen Syntaxbaum und den entsprechenden Syntaxgraphen.

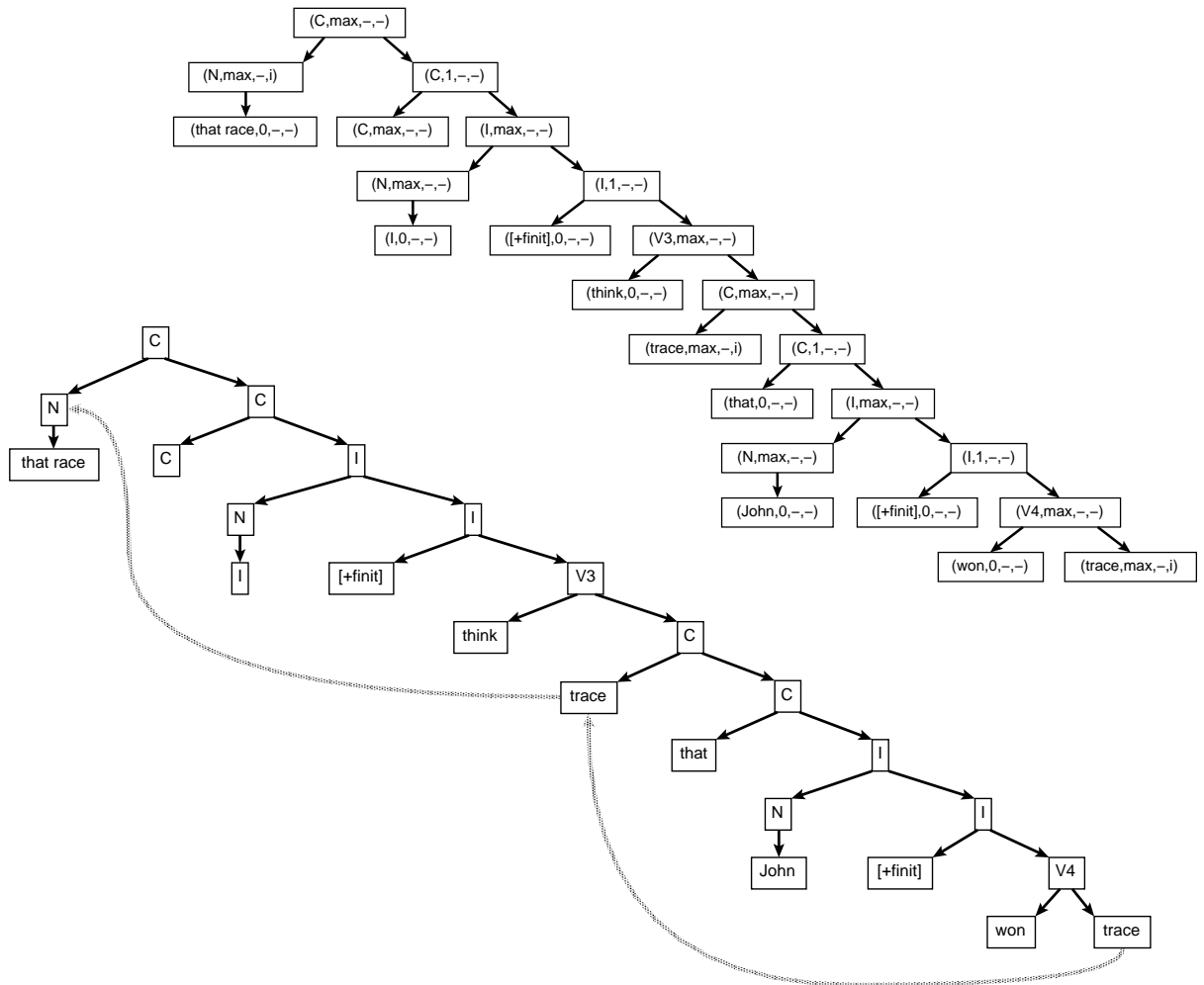


Abbildung 5.2: Zu Beispiel 21: Ein Syntaxbaum und der entsprechende Syntaxgraph

BEISPIEL 22: Im Satz „*That he_a loves himself_a, Peter_a believes.*“ wird ein Nebensatz topikalisiert. Interessant ist dabei, daß dieser ein Pronomen und eine Anapher enthält. Insgesamt enthält der Syntaxbaum also zwei Ketten. Im Syntaxbaum kommt dies durch Indizes, im Syntaxgraphen durch Querkanten zum Ausdruck. (Abbildung 5.3)

Die beiden Beispiele machen die Vereinfachung von Knotennamen deutlich. Gemeinsame Indizes sind durch Index- und Referenzkanten ersetzt. Diese bringen die in den Strukturen beinhalteten Querbeziehungen zum Ausdruck. (im Syntaxgraph grau dargestellt).

5.1.5 Zusammenfassung und Formulierung der weiteren Aufgabenstellung

Durch die beiden Maßnahmen (i) Elimination unendlicher Alphabete und (ii) Einfügen zusätzlicher Kanten wird ein wohldefiniertes Graphenmodell gewonnen. Auf kanonische Weise ergibt sich so ohne Informationsverlust aus jedem Syntaxbaum ein Syntaxgraph.

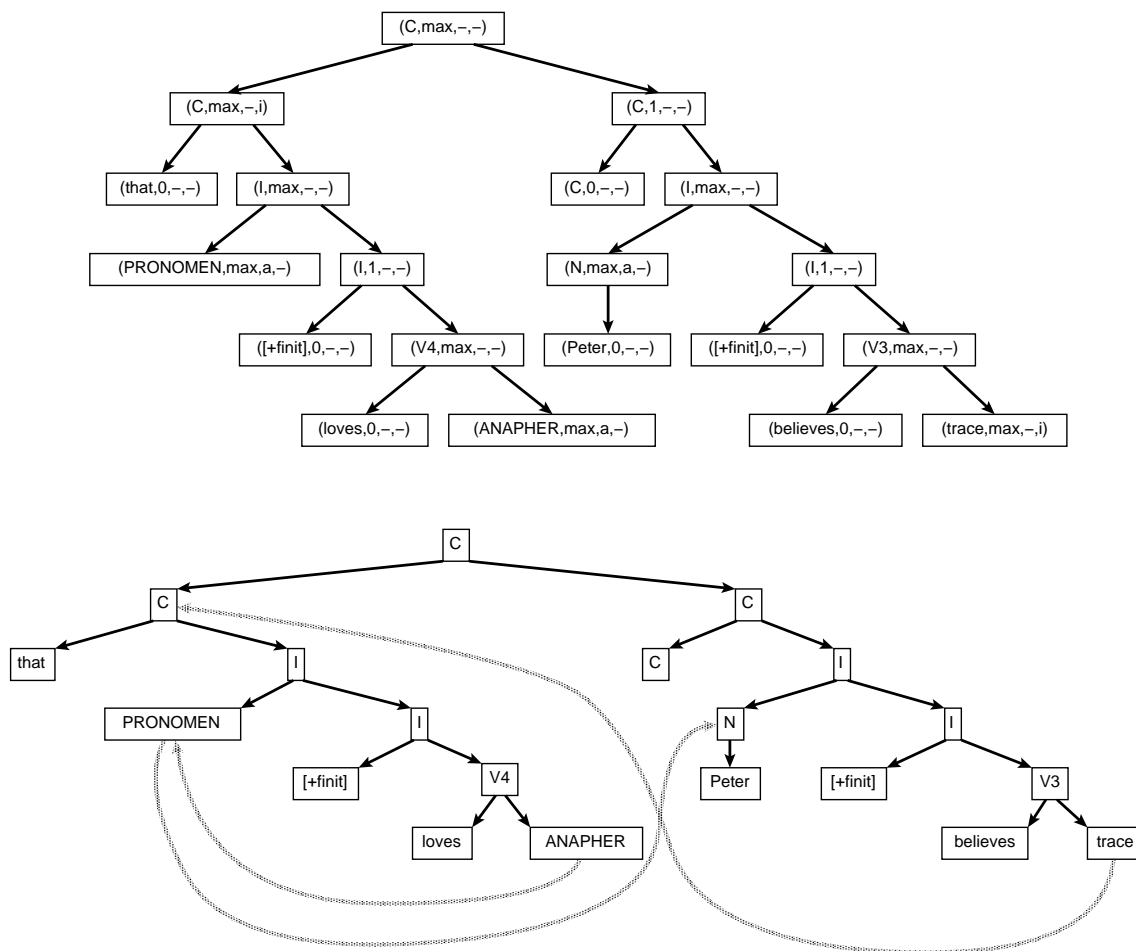


Abbildung 5.3: Zu Beispiel 22: Ein Syntaxbaum und der entsprechende Syntaxgraph.

Diejenigen Definitionen, die von den Begriffen *Koindizierung* und *Koreferenz* Gebrauch machen (Subjanz, Bindung etc.) sind von der Terminologie her angepaßt zu verstehen: Die Information über die Zugehörigkeit eines Knotens zu einer Äquivalenzklasse kommt nicht mehr über Indizes, sondern über Querkanten zu Ausdruck.

Es wird deutlich: Grundsätzlich handelt es sich bei einem Syntaxgraphen um eine Struktur, welche aus einem zugrundeliegenden Baum besteht, der von Querkanten überlagert wird. Letztere erschweren eine kompakte (derivationelle) Beschreibung. Die sich daraus ergebende *Aufgabenstellung dieses Kapitels* besteht nun darin, das durch GB repräsentationell gegebene Modell daraufhin zu untersuchen, ob es auch als Ganzes derivationell beschrieben werden kann. Interessant ist dabei, ob es Invarianten gibt, welche die Klasse *SG* charakterisieren. Zu diesem Zweck haben wir alle graphentheoretischen Begriffe in Erwägung gezogen. Für die interessanten Eigenschaften untersuchen wir im folgenden Abschnitt ihre Anwendbarkeit und stellen die Ergebnisse vor.

5.2 Eigenschaften von Syntaxgraphen

5.2.1 Syntaxgraphen: gradbeschränkt, zusammenhängend und zyklensfrei

In Abschnitt 4.1 haben wir bereits eine Reihe von Eigenschaften des Syntaxbaums gezeigt. Durch die zusätzlich eingeführten Querkanten ändern sich einige davon, andere bleiben erhalten:

LEMMA 6: *Ein Syntaxgraph ist zyklensfrei.*

BEWEIS: Das zugrundeliegende Gerüst des Syntaxgraphen enthält auf Grund der Baumstruktur keinen Zyklus. Zu zeigen bleibt, daß auf Grund zusätzlicher Querkanten kein Zyklus entstehen kann. Zunächst schließt die Definition 36 für eine Kette aus, daß eine Querkante einen Knoten mit einem Vorgänger auf dem Weg zur Wurzel verbindet. Kanten verbinden lediglich benachbarte Teilbäume miteinander. Zu zeigen bleibt dann, daß eine Konstruktion wie in Abbildung 5.4 *nicht* möglich ist.

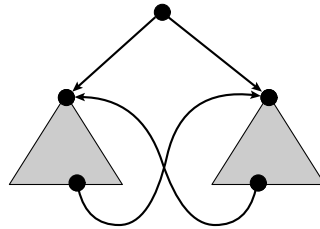


Abbildung 5.4: Querkanten lassen keine Zyklen entstehen

Tatsächlich stellen die Definitionen von α -Ketten und β -Ketten sicher, daß dies nicht passieren kann: Bis auf den Antezedens einer Kette sind alle Kettenknoten Blätter. Blattknoten sind bezüglich Zyklusbildung unbedenklich, da Ketten immer disjunkt sind (ein Knoten kann nur einer Kette angehören). Querkanten von α -Ketten verweisen stets nur auf linke Teilbäume. Lediglich Querkanten von β -Ketten können auch auf einen rechten Teilbaum verweisen, wenn eine Bewegung stattgefunden hat.⁴ In diesem Fall ist - wegen der Transformation - immer auch notwendig eine α -Kette vorhanden. Für diesen Fall ist abschließend sicherzustellen, daß kein Zyklus entsteht: Nach Definition 46 kann eine Querkante, die von links nach rechts weist, nur auf eine NP verweisen oder auf einen Blattknoten. In einer NP kann gemäß den Nachbarschaftsprinzipien keine CP enthalten sein, welche wiederum notwendige Voraussetzung für das Vorhandensein eines Knotens einer α -Kette ist. Dies schließt einen Zyklus aus.

□

LEMMA 7: *Sei SG ein Syntaxgraph. Dann ist SG ein einfach und ein schwach zusammenhängender zyklensfreier Graph. Der Graph SG beinhaltet einen einzigen Knoten, der nur auslaufende Kanten besitzt.*

BEWEIS: *Einfacher Zusammenhang* folgt aus der zugrundeliegenden Baumstruktur und wird von den eventuell hinzugefügten Kanten nicht beeinträchtigt. Die zugrundeliegende

⁴ Siehe zum Beispiel Abbildung 5.3

Baumstruktur garantiert *schwachen Zusammenhang*. Starker Zusammenhang kann auf Grund der *Zyklenfreiheit* nicht vorliegen. Im zugrundeliegenden Baum ist die Wurzel der einzige Knoten, der nur auslaufende Kanten besitzt. Da sich durch Hinzufügen von Querkanten der Grad eines Knotens nur erhöhen kann und zugleich sichergestellt ist, daß keine Querkante zum Wurzelknoten hinläuft, ist dieser Knoten, die einzige *Quelle* im Graphen. □

Da keine Querkanten vorkommen müssen, sind Syntaxgraphen in der Regeln nicht mehrfach zusammenhängend.

LEMMA 8: *Der Grad der Knoten eines Syntaxbaums ist nach oben beschränkt durch 4. Eingangsgrad und Ausgangsgrad sind dabei nach oben jeweils durch 2 beschränkt.*

BEWEIS: Im Syntaxbaum beträgt der Grad eines Knotens v maximal 3 ($\text{indeg}(v) \leq 1$, $\text{outdeg}(v) \leq 2$). An v kann höchstens eine Querkante einlaufen. Ist v ein Blatt, dann kann zugleich eine Querkante einlaufen und eine weitere Querkante auslaufen. Daraus folgt die Behauptung. □

LEMMA 9: *Ein Syntaxgraph hat $O(|V|)$ Blätter.*

BEWEIS: Nach Satz 1 hat ein Syntaxbaum $O(|V|)$ Blätter. Da bei einem Syntaxgraph gegenüber einem Syntaxbaum durch Hinzufügen von Querkanten höchstens aus einem Blatt ein innerer Knoten entstehen kann, nicht aber ein zusätzliches Blatt, folgt daraus die Behauptung. □

LEMMA 10: *Sei $G \in \mathcal{SG}$. Die Anzahl der Querkanten Q in G ist durch die Anzahl der Blätter im Gerüst von G beschränkt.*

BEWEIS: Eine Querkante verbindet einen Blattknoten des zugrundeliegenden Gerüsts mit einem anderen Knoten, wobei wir Referenz- und Indexkanten voneinander unterscheiden. Pro Knoten kann höchstens eine solche Kante wegweisen. Daraus folgt die Behauptung. □

SATZ 11: *Der Syntaxgraph hat linear viele Kanten gemessen an der Anzahl der Knoten:*

$$|E| = O(|V|)$$

BEWEIS: Da Gerüst und Querkanten bezüglich Kantenzahl in $O(|V|)$ sind, folgt die Behauptung. □

Diese Eigenschaft ist insofern interessant, als sie eine echte Beschränkung gegenüber allgemeinen Graphen darstellt. Diese können bis zu $O(|V|^2)$ viele Kanten besitzen. Ein Syntaxgraph ist damit eine vergleichbar dünne Struktur. Dieses Ergebnis läßt sich konkretisieren. Wie folgendes Lemma zeigt, kann auch für die Konstante der O -Notation eine (kleine) Schranke angegeben werden.

SATZ 12: Für die Anzahl der Kanten im Syntaxgraph SG gilt: $|E(SG)| \leq \frac{3}{2}|V(SG)|$.



BEWEIS: Das Gerüst eines Syntaxgraphen besitzt $|V| - 1$ Kanten. Wegen Lemma 10 und Satz 1 gibt es höchstens $\frac{1}{2}|V|$ Querkanten. Daraus folgt die Behauptung. \square

5.2.2 Syntaxgraphen sind nicht planar

Planare Graphen sind solche Graphen, die sich ohne Kreuzungen von Kanten darstellen lassen. Euler (1750) formulierte für einen planaren Graphen mit n Knoten und m Kanten die Ungleichung: $m \leq 3n - 6$. Planare Graphen haben also linear viele Kanten. Sie sind somit dünne Strukturen, was für die Verarbeitung und Verwaltung effiziente Algorithmen in Aussicht stellt.

Satz 11 besagt, daß Syntaxgraphen nur linear viele Kanten besitzen. Allerdings ist dies nur ein notwendiges, nicht aber ein hinreichendes Kriterium für die Planarität, wie der folgende Satz illustriert:

SATZ 13: *Syntaxgraphen sind nicht planar.*

BEWEIS: Da nach Kuratowski jeder nicht planare Graph einen zum K_5 () oder $K_{3,3}$ ()⁵ homöomorphen⁶ Subgraphen enthält, reicht es für den Beweis aus, ein entsprechendes Beispiel zu finden. Betrachte dazu die folgende Satzkonstruktion: „*John_i gave the book_j to Mary_k, because he_i promised it_j to her_k, when he_i showed it_j to her_k.*“ Abbildung 5.5 zeigt den dazugehörigen Syntaxgraphen. Aus Platzgründen wählen wir eine komprimierte Darstellung des Syntaxgraphen, welche die Phrasen nicht vollständig expandiert. Da hierbei nur große Bäume durch kleinere ersetzt werden, bleiben die Planaritätseigenschaften von dieser Vereinfachung unberührt.

Wie in GB geben Indizes die koreferenten Elemente an: i , j und k markieren die drei β -Ketten. Antezedens ist dabei jeweils „John“, „book“ und „Mary“. Die gemeinsamen Indizes werden im Syntaxgraph durch eine gemeinsame Kante zusammengefaßt.

Der Syntaxgraph enthält einen zum $K_{3,3}$ homöomorphen Subgraphen und ist damit nicht planar. Daraus folgt die Behauptung. Abbildung 5.6 zeigt ein anderes Layout desselben Graphen, in dem der $K_{3,3}$ besser zu erkennen ist. Die entsprechenden Knoten sind fett markiert. \square

Würde man unser Modell so verfeinern, daß auch Sätze mit Junktoren (wie „und“, „oder“, etc.) modelliert werden, ließe sich noch ein anderer Beweis entwickeln, dessen Idee wir hier kurz skizzieren: Mit Hilfe von Junktoren kann jeder boolesche Ausdruck über die

⁵ $K_{n,n}$ bezeichnet einen bipartiten Graphen mit $2n$ Knoten, bei dem jeder Knoten aus einer Menge mit allen Knoten der anderen Menge verbunden ist.

⁶ Ein Graph g_1 enthält dann einen zu einem Graphen g_2 homöomorphen Subgraphen, wenn g_1 durch iterative Anwendung der folgenden beiden Operationen in den Graphen g_2 umgeformt werden kann: (i) Ersetze einen Knoten vom Grad 2 durch eine Kante und (ii) lösche einen Knoten (und damit natürlich alle inzidenten Kanten). Diese Operationen vereinfachen die Graphstruktur.

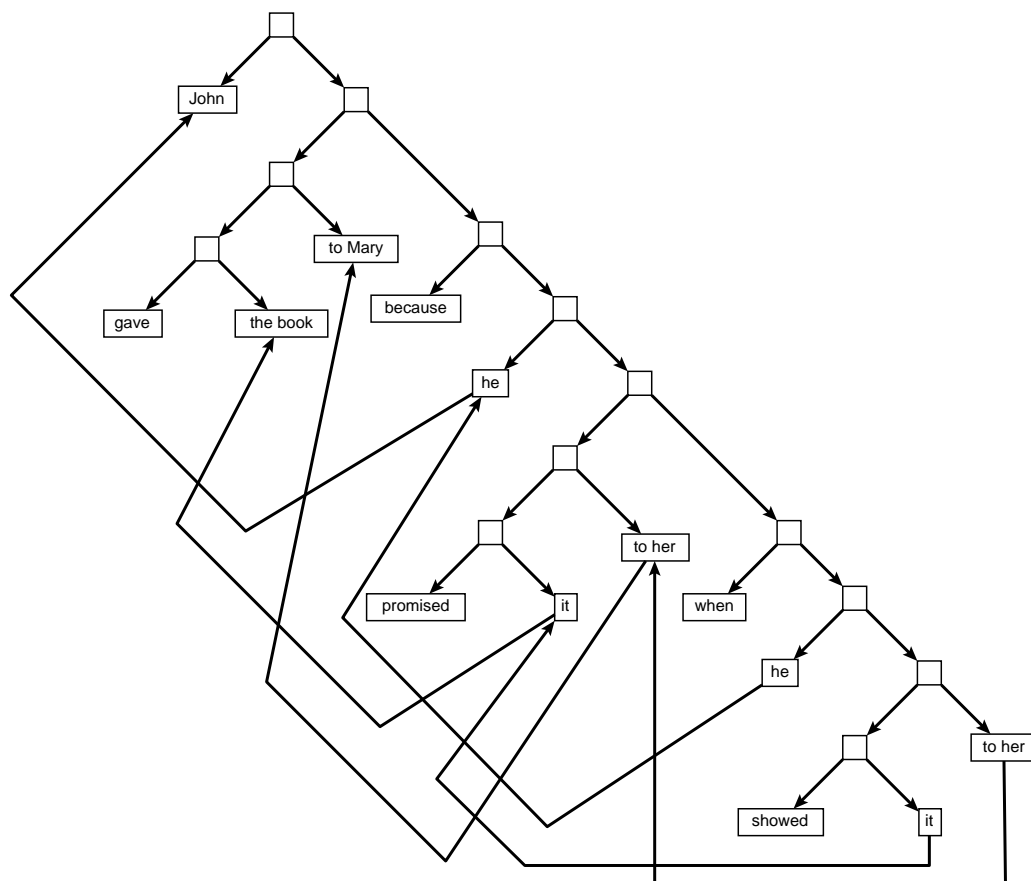


Abbildung 5.5: Syntaxgraphen sind nicht planar

Operatoren \vee und \wedge auf kanonische Weise in koordinierte Nominalphrasen transformiert werden, wobei die Variablen des booleschen Ausdrucks sich durch einen nominalen Lexikonentry modellieren lassen. Da Variablen im booleschen Ausdruck mehrfach vorkommen dürfen, lassen sich vielfältige syntaktische Strukturen „nach Wunsch“ zusammenzubauen, insbesondere solche, die zu beliebigem $K_{n,n}$ homöomorph sind.

5.2.3 Syntaxgraphen haben einen unbeschränkten Separator

Starke Separatoren ermöglichen das Partitionieren eines Graphen in zwei annähernd gleich große Teile. Wenn diese Teilungsstrategie rekursiv fortgesetzt werden kann, liegt es nahe, auf dieser Grundlage effiziente Divide & Conquer Strategien für Algorithmen zu entwickeln.⁷ Weiter folgen aus den Separatoreigenschaften Aussagen über die Anwendbarkeit von Graphgrammatiken für Graphsprachen.⁸ Aus diesem Grund ist es von Interesse, auch für Syntaxgraphen diesen Aspekt zu untersuchen. Grundsätzlich werden Knoten- von Kantenseparatoren unterschieden. Wir untersuchen zunächst Kantenseparatoren:

⁷ [LT79], [LT80], [Ull84]

⁸ [Sch87]

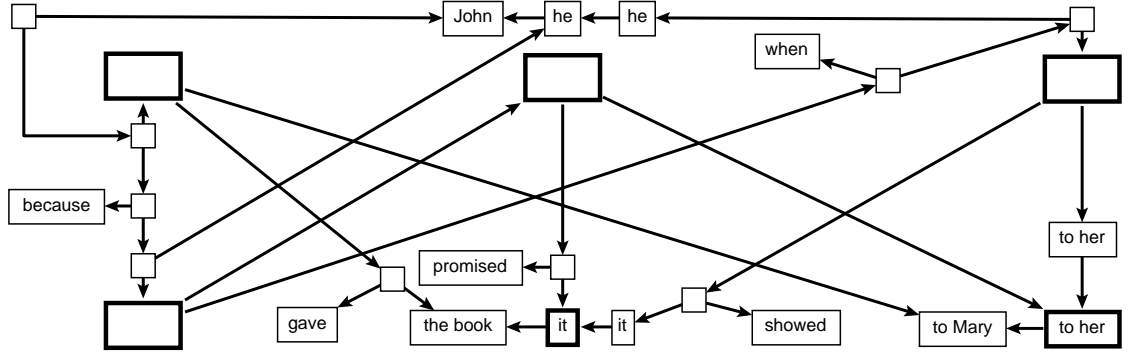


Abbildung 5.6: Der Graph aus Abbildung 5.5 anders dargestellt

DEFINITION 56 (KANTENSEPARATOR): Ein Kantenseparator $sep(G)$ für einen zusammenhängenden Graphen $G = (V, E)$ ist eine Kantenmenge $E' \subset E$, die gemäß der Vorschrift $(V, E \setminus E')$ den Graphen G in zwei Zusammenhangskomponenten $G_1 = G|_{V_1} = (V_1, E_1)$ und $G_2 = G|_{V_2} = (V_2, E_2)$ zerfallen läßt (mit $V_1, V_2 \subset V$ disjunkt), so daß für den Kantenseparator gilt: $sep(G) = E \setminus (E_1 \cup E_2)$.

DEFINITION 57 ((r, c) - $f(n)$ -KANTENSEPARATOR): Ein Graph G hat einen (r, c) - $f(n)$ -Kantenseparator oder G ist (r, c) - $f(n)$ -separierbar ($1 \leq f(n) \leq n^2$) genau dann, wenn für r ($0 \leq r \leq \frac{1}{2}$) der Graph G rekursiv in zwei disjunkte Untergraphen der Größe $|G_1| = \alpha_1|G|$, $|G_2| = \alpha_2|G|$ ($r \leq \alpha_1, \alpha_2 \leq 1 - r$) zerlegbar ist. Dabei dürfen höchstens $c * f(|V|)$ Kanten entfernt werden.

BEISPIEL 23 (CHARAKTERISTISCHE SEPARATOREN): Binär**ä**ume haben einen konstanten $O(1)$ Separator, denn es ist rekursiv möglich, einen Binärbaum durch Löschen einer einzigen Kante in zwei annähernd gleich große Teile zu zerlegen ($r = \frac{1}{3}$, $f(n) = 1$, $c = 1$).⁹ Quadratische Gitter besitzen einen \sqrt{n} -Separator ($r = \frac{1}{2}$, $f(n) = \sqrt{n}$, $c = 1$) und $n \times m$ -Gitter haben einen $\min\{n, m\}$ -Separator ($r = \frac{1}{2}$, $f(n) = \min\{n, m\}$, $c = 1$). Vollständige Graphen schließlich haben einen n^2 -Separator ($r = \frac{1}{2}$, $f(n) = n^2$, $c = 1$).

Wir sind an solchen Separatoren interessiert, die einen Graphen in möglichst gleich große Teile zerlegen. Diese heißen *starke Separatoren*.¹⁰

DEFINITION 58 (STARKER KANTENSEPARATOR): Ein $(\frac{1}{2}, c)$ - $f(n)$ -Separator heißt *starker Separator*.

Für die praktische Anwendbarkeit von Graphgrammatiken auf eine Klasse von Graphen ist ein starker Separator vom Typ $O(1)$ erstrebenswert, wie ihn beispielsweise Bäume besitzen. Dies motiviert, nach den Separatoreigenschaften von Syntaxgraphen zu fragen.

Von Syntaxgraphen ist bekannt, daß sie gradbeschränkt sind und linear viele Kanten besitzen. Das allein reicht nicht aus, um einen konstanten Separator zu garantieren, wie

⁹ Daß für Bäume eine einzige Kante ausreicht, um die Graphenstruktur zu separieren, wurde für Syntaxgraphen in Kapitel 4 für eine kompakte Definition von Barrieren benutzt.

¹⁰ Mehr zum Begriff des starken Separators bei [Ull84]

das Beispiel der Gittergraphen zeigt. Tatsächlich können wir Gitter dazu verwenden, um zu zeigen, daß auch Syntaxgraphen keinen beschränkten Separator besitzen.

Dazu benötigen wir den Begriff des Quasigitters, bei denen kein Knoten einen Grad größer als 3 besitzt. Dennoch besitzen diese eine gitterähnliche Struktur:

DEFINITION 59 ($n \times m$ -QUASIGITTER): Ein $n \times m$ -Quasigitter ist eine gitterähnliche Struktur mit $2 * n * m$ Knoten, die gemäß Abbildung 5.7 miteinander verbunden sind.

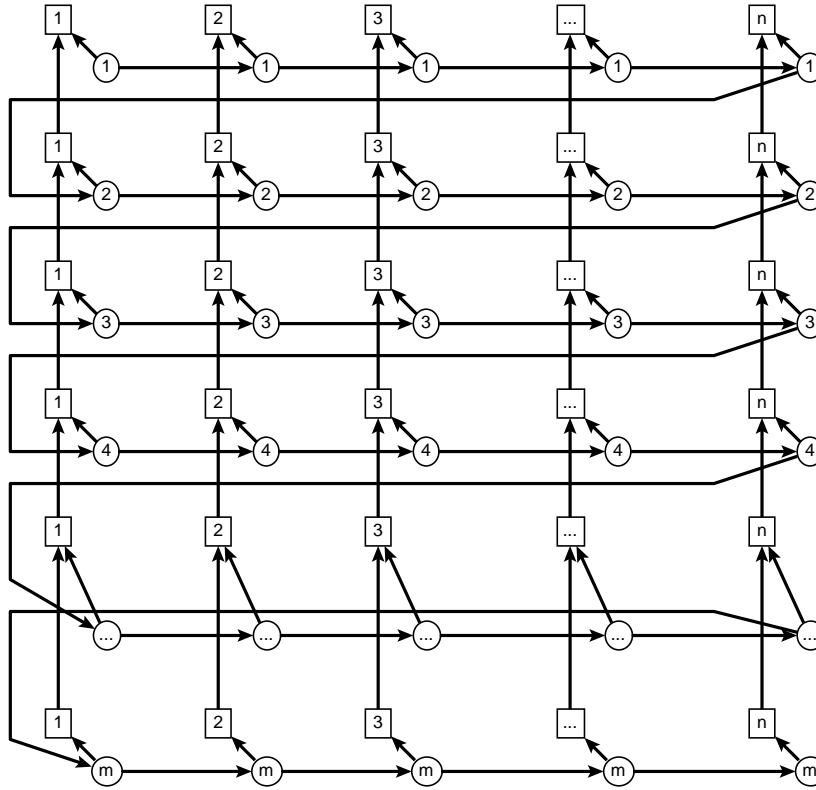


Abbildung 5.7: Aufbau von einem $n \times m$ -Quasigitter

Im Gegensatz zu einem einfachen Gitter (innere Knoten besitzen Grad 4) ist der Grad eines Quasigitters maximal 3. Abbildung 5.7 macht deutlich, daß aber auch hier eine Gitterstruktur vorliegt.

LEMMA 14: Ein $n \times m$ -Quasigitter hat einen starken $\min\{n+1, 2m-1\}$ -Kantenseparator.

BEWEIS: Zu zeigen: $\min\{n+1, 2m-1\}$ Kanten reichen immer, um ein Quasigitter zu separieren. Dabei sind zwei Fälle zu unterscheiden: (i) Sei $n > 2m-2$. Dann reichen $2m-1$ Kanten aus, um den Graphen zu halbieren (senkrecht separieren). (ii) Sei $m \geq (n+2)/2$. Dann reichen $n+1$ Kanten aus, um den Graphen zu halbieren (waagrecht separieren). Diese Strategie läßt sich rekursiv fortsetzen. \square

Eine Aussage über die untere Schranke ist schwieriger zu formulieren. Dazu beweisen wir vorab folgende Aussagen:

LEMMA 15: Sei $G = (V, E)$ ein Graph mit konstantem $(\frac{1}{3}, c)$ -1-Separator. Dann besitzt G einen starken $O(\log|V|)$ -Separator.

BEWEIS: Abbildung 5.8 illustriert, wie ein $(\frac{1}{3}, c)$ -1-Separator einen Graphen G rekursiv in Teilgraphen zerlegt. Die inneren Knoten des Baumes illustrieren jeweils einen Separator S_i , also eine Kantenmenge, deren Größe nach Voraussetzung durch eine Konstante c beschränkt ist. Jeder Separator S_i erzeugt einen Graphen G_{i1} und G_{i2} , die die Blattknoten bilden.

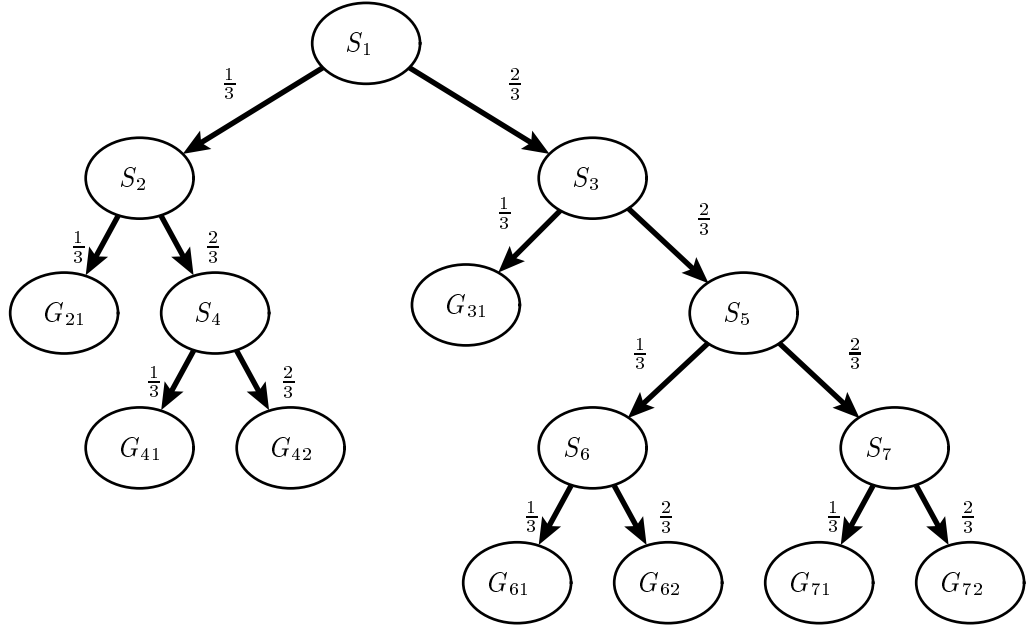


Abbildung 5.8: Konstruktion eines starken Separators

Durch geeignete Auswahl mehrerer G_{ij} läßt sich ein Graph H zusammensetzen, der halb so groß ist, wie G . Von jeder Ebene des Separatorbaumes kommt dabei höchstens ein S_i zur Anwendung. Auf Grund der mit $\log|V|$ beschränkten Höhe des Separatorbaumes sind höchstens $\log|V|$ Separatoren S_i nötig, um H vollständig von $G \setminus H$ zu separieren.

Da die Größe der S_i durch c beschränkt ist, folgt, daß G einen $(\frac{1}{3}, c)$ - $(\log|V|)$ -Separator besitzt. Dies ist nach Definition ein starker $O(\log|V|)$ -Separator. \square

LEMMA 16: Es existiere ein starker $(\frac{1}{2}, c)$ - $O(f)$ -Separator für $n \times m$ -Quasigitter. Dann existiert ein $(\frac{1}{3}, 2 * c)$ - $O(f)$ -Separator für ein $n \times m$ -Gitter.

BEWEIS: Sei S ein starker $O(f)$ -Separator für $n \times m$ -Quasigitter G_1 . Wir bauen in zwei Schritten G_1 zu einem einfacheren Quasigitter G_2 und dann zu einem normalen Gitter G_3 um. Dabei beobachten wir, was mit den jeweiligen Separatoren geschieht:

1. $n \times m$ -Quasigitter \rightarrow vereinfachtes $n \times m$ -Quasigitter: Die Kanten, die ähnlich einem 'Zeilenumbruch' den in einer Zeile am weitesten rechts stehenden runden

Knoten mit dem am weitesten links stehenden runden Knoten der nächsten Zeile verbinden, werden gelöscht.

Der neue Separator S_2 für G_2 kann infolge dieser Operation nicht größer werden: Es gilt $S_2 \leq S_1$. Da die Anzahl der Knoten nicht verändert wird, ist S_2 immer noch ein starker Separator.

2. vereinfachtes $n \times m$ -Quasigitter $\rightarrow n \times m$ -Gitter: Die beiden Knoten, die entlang einer Diagonalkante δ miteinander verbunden sind, werden verschmolzen. Die Diagonalkante verschwindet, die inzidenten Kanten bleiben erhalten wie in Abbildung 5.9 illustriert.

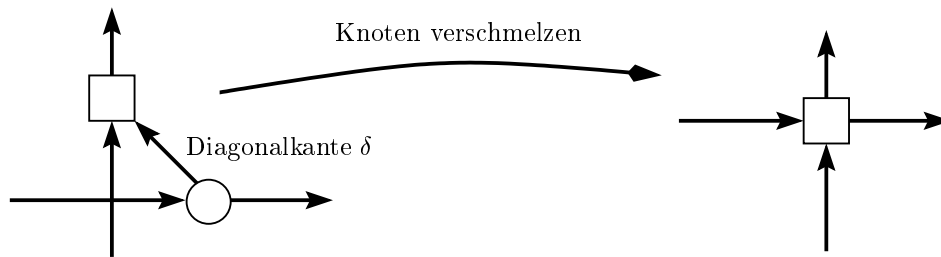
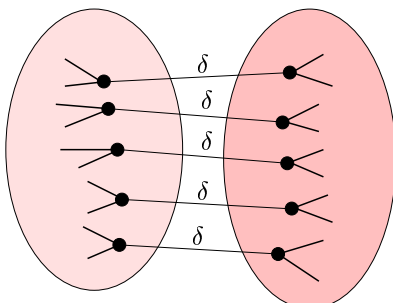


Abbildung 5.9: 'diagonal' benachbarte Knoten werden verschmolzen

Wir unterscheiden nun zwei Extremfälle:

- a) Der Separator S_2 besteht nur aus Diagonalkanten (Abbildung 5.10 links). Dann gibt es einen Separator S_3 für G_3 der statt der Diagonalkante jeweils die beiden Nachbarschaftskanten durchläuft. Der neue Separator kann infolge dieser Operation schlimmstenfalls doppelt so groß werden wie S_2 . Es gilt damit $S_3 \leq 2 * S_2 \leq 2 * S_1$. Die starke Separatoreigenschaft kann nicht mehr garantiert werden, wenn die Knoten entlang der ehemaligen Diagonalkanten einseitig auf einen der beiden separierten Teilgraphen verteilt werden. Mit einer *Reisverschlußstrategie* (Abbildung 5.10 rechts), welche die Knoten abwechselnd dem linken und dann dem rechten Graphen zuordnet, kann sichergestellt werden, daß das Verhältnis der Knoten in beiden Teilgraphen nach der Separierung annähernd ausgeglichen ist.

Situation vor der Verschmelzung



Separation nach Reisverschlußmethode

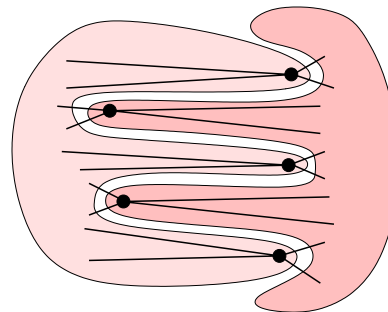


Abbildung 5.10: Verschmolzene Knoten werden abwechselnd verteilt

- b) Der Separator enthält keine Diagonalkante und im Extremfall befinden sich alle Diagonalkanten im (o.B.d.A) linken Teilgraphen. Dann kann sich dort die Zahl der Knoten halbieren. Das neue Knotenverhältnis lautet dann $\frac{1}{3} : \frac{2}{3}$.

Zusammenfassung: Das in zwei Schritten konstruierte $n \times m$ -Gitter besitzt auf der Basis des starken Separators S_1 einen $(\frac{1}{3}, 2 * c)$ - $O(f)$ -Separator S_3 . □

LEMMA 17: *Ein $n \times m$ -Quasigitter Q besitzt keinen beschränkten Kantenseparator.*

LEMMA 18: *Ein $n \times m$ -Quasigitter Q besitzt einen $s(n)$ -beschränkten Kantenseparator mit $s(n) \leq \min\{\frac{n}{\log n}, \frac{m}{\log n}\}$.*

BEWEIS: (durch Widerspruch) Angenommen, Q besitzt einen starken $O(1)$ -Separator. Dann hat ein $n \times m$ -Gitter nach Lemma 16 einen $(\frac{1}{3}, c)$ - $O(1)$ -Separator und mit Lemma 15 einen starken $O(\log(n * m))$ -Separator. Dies ist ein Widerspruch zu dem bekannten Ergebnis, daß Gittergraphen einen starken $O(\min\{n, m\})$ -Separator besitzen. □

Diese Eigenschaft von Quasigittern nutzen wir aus, um eine Aussage über die Separatoreigenschaften von Syntaxgraphen zu formulieren.

SATZ 19: *Syntaxgraphen haben einen unbeschränkten Separator.*

BEWEIS: Zu jedem $n \times m$ -Quasigitter QG läßt sich eine Syntaxgraph SG konstruieren, so daß gilt: SG enthält QG als homöomorphen Subgraphen. Der diesem Syntaxgraphen zugrundeliegende Satz besteht aus einem Hauptsatz und $(n * m) - 1$ -Nebensätzen:¹¹

„*John₁ dreams, that Mary₂ dreams, that John₃ dreams, . . . , that Mary_n dreams, {that he₁ dreams, that she₂ dreams, that he₃ dreams, . . . , that she_n dreams}*“^{*m*}“

Abbildung 5.11 zeigt den (vereinfachten) Bauplan für Syntaxgraphen zu dieser Menge von Sätzen und illustriert, daß es sich dabei um eine Repräsentation eines $n \times m$ -Quasigitters handelt: Da die Menge der Syntaxgraphen unendlich viele Graphen vom Typ Quasigitter enthält, die bezüglich n und m unbeschränkt sind, ist sichergestellt, daß auch der Separator für Syntaxgraphen nicht beschränkt ist. □

GB akzeptiert einen solchen Satz als wohlgeformt. Alle Anforderungen an einen Syntaxgraphen sind erfüllt, es wird gegen kein Prinzip verstoßen. In diesem Sinne ist er grammatisch, auch wenn er die Toleranz eines „*ideal speaker*“ stark strapaziert. Die Einhaltung aller syntaktischen Vorgaben sagt also – wie zu erwarten – nichts über die Semantik des Satzes aus. Der Beweis läßt sich auch verstehen als Hinweis darauf, daß GB zu mächtig ist, also zuviel beschreibt. Es muß für praktische Anwendungen also kein

¹¹ Die Klammern und der hochgestellte Index m sind zu lesen wie in einem regulären Ausdruck.

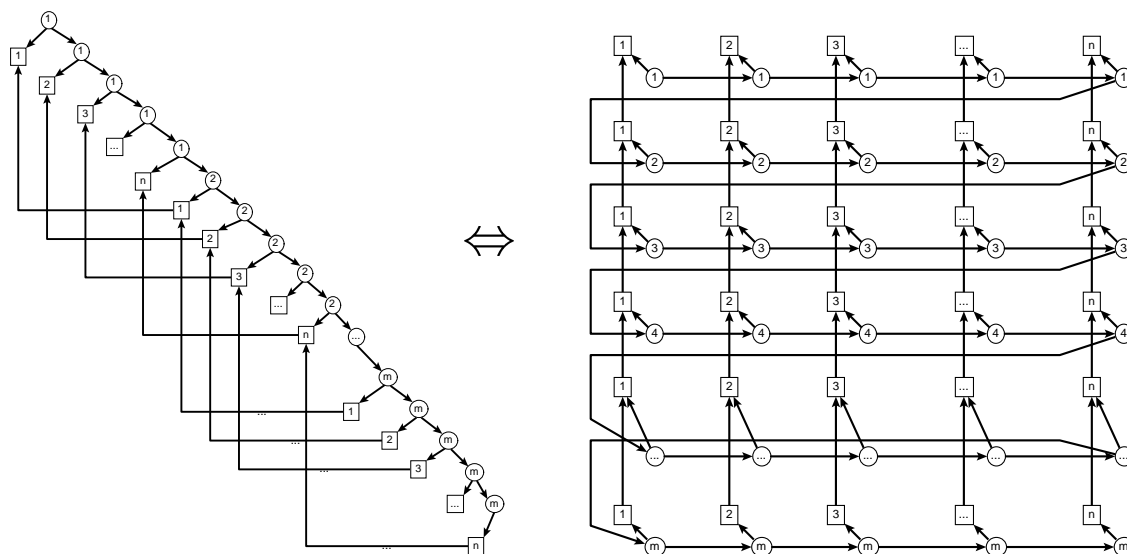


Abbildung 5.11: $n \times m$ -Quasigitter lassen sich als verdichtete Syntaxgraphen ansehen

Mangel sein, wenn eine Graphgrammatik nicht alle Syntaxgraphen im Sinne von GB erzeugt.

Grund dafür, daß Syntaxgraphen wie in dem Beispiel unbeschränkt groß werden können, ist die Art und Weise, wie in natürlichen Sprachen auf nominale Konstrukte immer wieder Bezug genommen werden kann. Unter Berücksichtigung der Bindungsprinzipien¹² können sie beliebig weit entfernt über den Satz verteilt werden, wie in dem folgendem Beispiel „John“ und „he“:¹³

„John_i believes, that Mary_j dreams{, that she_j dreams}* , that he_i loves her_j.“

Trotz der in Kapitel 4 vorgenommenen Einschränkungen bezüglich GB ist es also unmöglich, den Separator auf einen konstanten Wert zu beschränken.

Dieses Ergebnis läßt sich zur Formulierung eines Gütekriteriums für den Stil (gerade noch) grammatischer Sätze nutzen. Die linguistischen Beispiele für Graphen mit hohem Separator sind stets Beispiele für „stilistisch schlechtere“ Sätze, was sich als Grundlage weiterer empirischer Untersuchungen anbietet. Da es hier aber nicht darum geht, ein formales Maß für den Stil eines Satzes, sondern vielmehr für seine Grammatikalität zu entwickeln, vertiefen wir diesen Aspekt nicht weiter.

¹² Bindungsprinzipien beschränken nicht die Reichweite von Referenzen auf Nominalphrasen. Sie bestimmen nur, ob abhängig von der Entfernung ein Pronomen oder eine Anapher an einer Position stehen muß.

¹³ Die Symbole „{“, „}“ und „*“ im Beispiel sind wie in einem regulären Ausdruck zu lesen.

5.2.4 Die Baumbreite von Syntaxgraphen ist unbeschränkt

Betrachtet man **NP**-vollständige Probleme für Graphen, werden diese im Spezialfall von Bäumen oft effizient lösbar. Diese Beobachtung läßt sich verallgemeinern: Auch beliebige Graphen können eine baumähnliche Struktur besitzen. Der Begriff Baumbreite formalisiert diese Idee. Für Graphen mit beschränkter Baumbreite gilt dann, daß viele **NP**-vollständige Probleme in der Klasse **P** sind. Die Begriffe wurden erstmals diskutiert in [RS83] und [RS86]. Wir zitieren nachfolgend aus [Klo93] und [Bod92].

DEFINITION 60 (BAUMZERLEGUNG): Eine Baumzerlegung eines Graphen $G = (V, E)$ ist ein Paar $(\{X_i | i \in I\}, T = (I, F))$ wobei $\{X_i | i \in I\}$ eine Menge von Teilmengen von V ist. Jedes X_i repräsentiert einen (Super)-Knoten des Baumes T mit folgenden Eigenschaften:

- $\cup_{i \in I} X_i = V$
- Für jede Kante $(v, w) \in E$ existiert ein $i \in I$ mit $v \in X_i$ und $w \in X_i$
- Für alle $i, j, k \in I$ gilt: Ist j auf dem Pfad von i nach k im Baum T , dann gilt: $X_i \cap X_k \subseteq X_j$.

Eine Baumzerlegung ist nicht eindeutig. Deshalb ist folgender Begriff sinnvoll:

DEFINITION 61 (BAUMBREITE, TREEWIDTH): Als Baumbreite (Treewidth) einer Baumzerlegung $(\{X_i | i \in I\}, T = (I, F))$ bezeichnen wir $\max_{i \in I} |X_i| - 1$. Die Baumbreite eines Graphen G ist die minimale Baumbreite aller möglichen Baumzerlegungen von G .

BEISPIEL 24: Graphen mit Baumbreite 0 sind die leeren Graphen. Graphen mit Baumbreite 1 sind Wälder, also alle kreisfreien Graphen. Die Graphen mit Baumbreite 2 zeichnen sich dadurch aus, daß sie sich durch Mehrfachanwendung der Operationen „lösche Knoten mit Grad 1“ und „Kontrahiere Knoten mit Grad 2 zu einer Kante“ zu einem isolierten Knoten umformen lassen.

Das vorangehende Beispiel zeigt, daß Syntaxgraphen nicht die Baumbreite 1 oder 2 besitzen können, denn für manche in dieser Arbeit untersuchten Syntaxgraphen war der entsprechende Wert bereits größer. Es bleibt die Frage: Läßt sich die Baumbreite für Graphen aus \mathcal{SG} prinzipiell durch eine Konstante beschränken?

Zur Beantwortung dieser Frage hilft der Begriff des Knotenseparators (nicht Kantenseparator).

DEFINITION 62 (KNOTENSEPARATOR): Ein zusammenhängender Graph $G = (V, E)$ besitzt einen $(f(n), r)$ -Knotenseparator genau dann, wenn $V(G)$ in drei Mengen A, B und C partitioniert werden kann, so daß $|C| \leq f(n)$ und $|A|, |B| \leq r * n$. Es darf dann keine Kanten zwischen A und B geben.

Folgender Satz stellt einen Zusammenhang zwischen Knotenseparatoren und Baumbreite herstellt:¹⁴

¹⁴ Der Beweis findet sich bei Kloks [Klo93, S.22]

SATZ 20: Sei G ein Graph mit n Knoten und Baumbreite k . Dann existiert eine Menge S mit $k+1$ Knoten, so daß jede zusammenhängende Komponente von $G[V \setminus S]$ höchstens $\frac{1}{2}(n-k)$ Knoten besitzt.

Wir können diesen Satz so interpretieren, daß ein Graph mit Baumbreite $\leq k$ einen konstanten $(k+1, \frac{1}{2})$ -Knotenseparator besitzt.

SATZ 21: Quasigitter haben einen unbeschränkten Knotenseparator.

BEWEIS: Angenommen es gäbe einen beschränkten Knotenseparator S mit k Knoten. Unabhängig von der Größe des Quasigitters G haben alle Knoten von G nach Konstruktionsprinzip einen beschränkten Grad ≤ 3 . Der Separator S löscht dann höchstens $3k$ Kanten aus dem Graphen. Betrachte nun Quasigitter mit der Dimension $((k+1) * 3) \times ((k+1) * 3)$. Dieses Quasigitter ist so beschaffen, daß es nicht durch Löschen von k Knoten separiert werden kann, die Annahme ist also falsch. \square

KOROLLAR 22: Quasigitter haben keine beschränkte Baumbreite und damit gilt unmitelbar: Syntaxgraphen haben keine beschränkte Baumbreite.

Die Ergebnisse zeigen, daß der Begriff Baumbreite keinen Ansatz für die effiziente Beschreibung von Syntaxgraphen bietet.

5.2.5 Zusammenfassung

Die Elemente der Klasse \mathcal{SG} besitzen folgende strukturelle Eigenschaften: Als DAGs sind sie zyklensfrei und einfach zusammenhängend. Die Richtung der Kanten garantiert schwachen, nicht aber starken Zusammenhang. Außerdem ist der Grad beschränkt auf maximal 4. Es existieren keine langen Knotensequenzen ohne Abzweigungen. Es gibt eine Quelle und unbeschränkt viele Senken. Weder Separator noch Baumbreite sind beschränkt. Schließlich besitzen Syntaxgraphen linear viele Kanten, sind aber nicht planar.

Daß die „böartigen“ Fälle und die Gegenbeispiele stets mit Sätzen schlechten Stils zu korrelieren scheinen, halten wir nicht für einen Zufall. Leichter zu verarbeitende Sätze lassen sich durch Bäume mit weniger Querkanten repräsentieren. Diesen Zusammenhang weiter zu untersuchen zielt aber an der Aufgabenstellung dieser Arbeit vorbei, weil wir uns zum Ziel gesetzt haben, syntaktische Strukturen formal zu beschreiben. Weiterführende Zusammenhänge zwischen Struktur und Stil zu betrachten, würde auch empirische Untersuchungen erfordern, die den Rahmen dieser Arbeit sprengen.

5.3 Syntaxbäume und konfluente Graphgrammatiken

Bis jetzt wurden ausgewählte Eigenschaften von Syntaxgraphen auf der Basis einer repräsentationellen Definition untersucht. Eine umfassende vollständige und gleichzeitig kompakte derivationelle Darstellung von Syntaxgraphen steht noch aus. Grundsätzlich

erscheinen als Technik für derivationelle Fragestellungen formale Sprachen als ein geeignetes Instrument. Im Falle von Graphsprachen, wie wir sie hier mit der Menge der Syntaxgraphen vorliegen haben, sind das insbesondere Graphgrammatiken.¹⁵

Inwieweit es möglich ist, die Mächtigkeit von repräsentationellen Prinzipien auf derivationelle Produktionen abzubilden, hängt vor allem von der Beschaffenheit der Prinzipien ab. Wir werden deshalb in mehreren Schritten eine Graphgrammatik entwickeln, indem wir zunächst kompliziertere Prinzipien unberücksichtigt lassen und die Graphgrammatik nach und nach vervollständigen.

5.3.1 Graphgrammatik, Konfluenz und Separatoren

Für eine nicht nur vollständige, sondern aus algorithmischer Sicht auch effiziente Lösung ist der Begriff der Konfluenz wichtig:¹⁶ Wenn Konfluenz vorliegt, dann lassen sich viele Beweistechniken und Eigenschaften von den kontextfreien Wortgrammatiken auf Graphgrammatiken übertragen. Aus den vorangehenden Ergebnissen läßt sich folgern, daß keine Graphgrammatik mit dieser Eigenschaft existieren kann. Dies ergibt sich insbesondere aus den Separatoreigenschaften von Syntaxgraphen, wobei wir ein Theorem von Schuster verwenden.¹⁷

SATZ 23 (SEPARATOR-THEOREM): *Sei GG eine Graphgrammatik. Ist GG konfluent und $L(GG)$ gradbeschränkt und $G \in L(GG)$, dann besitzt G einen $O(1)$ -Kantenseparator, der nicht von der Größe des Graphen abhängig ist.*

KOROLLAR 24: *Sei M eine Menge von Graphen, die eine Teilmenge $M' \subset M$ mit unendlich vielen Graphen mit beschränktem Grad und unbeschränktem Kantenseparator enthält. Dann gibt es keine konfluente Graphgrammatik GG mit $M = L(GG)$.*

SATZ 25: *Die Menge aller Syntaxgraphen kann nicht mit einer konfluenten Graphgrammatik generiert werden.*

BEWEIS: Nach Satz 19 existiert für den Kantenseparator von Syntaxgraphen keine obere Schranke. Da weiter nach Lemma 8 garantiert ist, daß Syntaxgraphen von beschränktem Grad sind, folgt mit Korollar 24 die Aussage. □

5.3.2 Eine konfluente Graphgrammatik GG_{deriv} für Syntaxgraphen

Die vorangehenden Ergebnisse machen deutlich, daß die Menge \mathcal{SG} nicht vollständig mit konfluenten Graphgrammatiken beschrieben werden kann. Aus zwei Gründen ist dies aber unproblematisch, weil praktische Anwendungsfälle von diesen Einschränkungen unberührt bleiben:

¹⁵ Graphgrammatiken wurden bereits in Abschnitt 2.2 eingeführt.

¹⁶ Definition 7 führt den Begriff der Konfluenz formal ein.

¹⁷ [Sch87]

1. GB ist ‘nur’ ein Modell für \mathcal{NL} . In der Einleitung wurde bereits darauf hingewiesen, daß ein Modell für \mathcal{NL} grundsätzlich nie mehr sein kann, als eine Approximation. Es muß deshalb keinen Schaden darstellen, wenn ausgewählte Aspekte von GB sich nicht mit konfluenten Graphgrammatiken modellieren lassen. Dies ist insbesondere dann der Fall, wenn es sich um dermaßen irrelevante Satzkonstruktionen handelt, wie wir sie hier aufgezeigt haben.
2. Eine Menge \mathcal{SG} zur Beschreibung syntaktischer Strukturen muß nicht zwingend auf der Basis von GB definiert werden. Wir orientieren uns hier an GB und verwenden es zur Bewertung eines eigenen Modells. Indem wir zeigen können, welche Regularitäten sich umsetzen lassen, können wir unser Modell an linguistischen Ergebnissen messen. Es zeigt sich, daß Einschränkungen nur bei solchen syntaktischen Konstruktionen auftreten, die in der Praxis sowieso keine Rolle spielen.

Wir stellen nun explizit eine konfluente Graphgrammatik vor, die ausschließlich GB -kompatible Graphen erzeugt. In Anlehnung an die im ersten Prinzip vorgestellte Graphgrammatik GG_{X-Bar} füllen wir die mit den Markierungen *NewPhrase*, *RefinePhrase*, *LexInsertion*, *max*, *project*, *head* und *grammar* benannten Produktionen in einer Graphgrammatik GG_{deriv} mit Inhalt. Die zusätzlichen Knoten- und Kantenbezeichnungen orientieren sich an den im zweiten und dritten Prinzip zugelassenen Typen. Die realisierten Kombinationen von Typen und die Benennung der Kanten setzen die im fünften Prinzip zum Ausdruck kommenden Restriktionen und Anforderungen um.

Die Umsetzung dieser Prinzipien in eine konfluente Graphgrammatik ist in erster Linie eine mechanische Arbeit und stößt auf keine grundsätzlichen formalen Schwierigkeiten. Wir erhalten als Ergebnis eine endliche Beschreibung einer unendlichen Menge von Syntaxgraphen, die alle Bäume sind.

Dies ändert sich, wenn auch das vierte Prinzip in eine konfluente Graphgrammatik integriert werden soll. Die dort formulierten Restriktionen für die Verteilung von Indizes lassen sich trivialerweise einfach dadurch berücksichtigen, daß keine Indizes modelliert werden, wie wir das im nun folgenden ersten Lösungsansatz tun. Erst in einem anschließenden Schritt wenden wir uns der Aufgabenstellung einer verfeinerten Graphgrammatik zu, welche weitere Phänomene beherrscht und diese dann durch Querkanten modelliert.

Das repräsentationelle Prinzip schließlich ist implizit im Graphgrammatik Formalismus enthalten. Wenn die anderen Prinzipien richtig in Produktionen umgesetzt wurden, kann die Graphgrammatik gar nicht anders, als derivationell nur solche Graphen zu generieren, die repräsentationell auch erlaubt sind.

5.3.2.1 Startkonfiguration für die Graphgrammatik GG_{deriv}

Die Graphgrammatik GG_{deriv} umfaßt zahlreiche Symbole und Produktionen, die wir nachfolgend sortiert nach Ihrer Bedeutung für den Syntaxgraphen in mehreren Abschnitten vorstellen. Wir beginnen mit dem Startsymbol:

Nichtterminalsymbol S

Startsymbol S

Die weiteren Symbole (und die Produktionen) folgen auf den nächsten Seiten. Dort wird nacheinander die Modellierung von *Nominalphrasen*, *Adjektivphrasen/Adverbialphrasen*, *Präpositionalphrasen*, *funktionalen Kategorien C und I* sowie *Verbalphrasen* mit geeigneten Produktionen demonstriert.

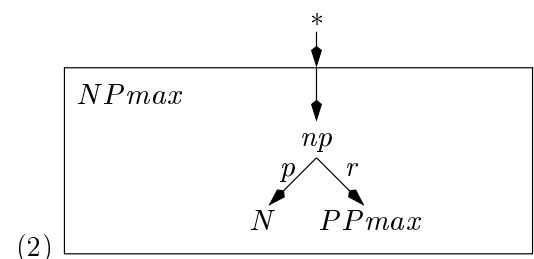
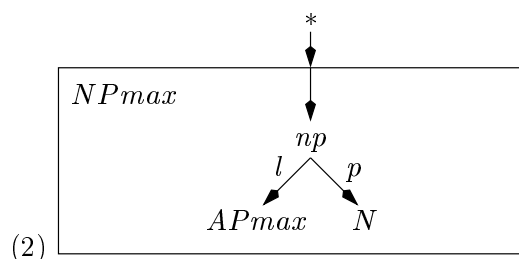
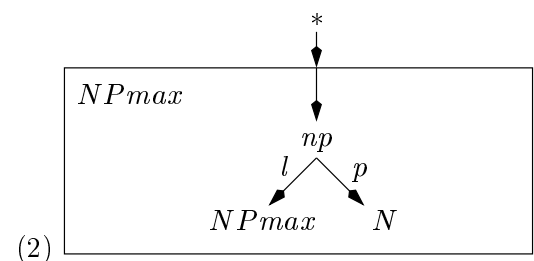
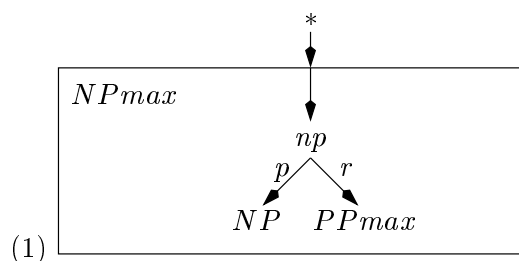
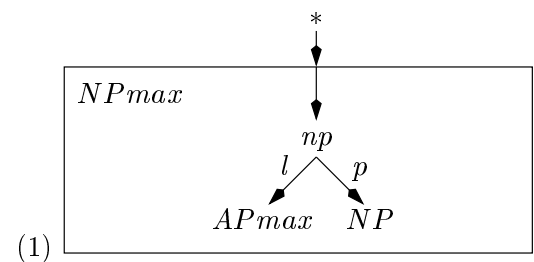
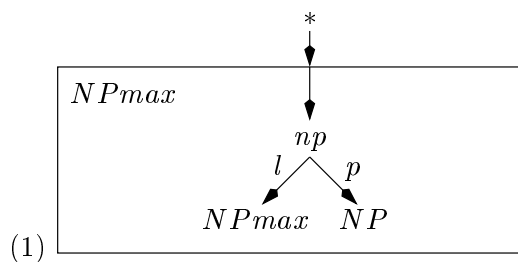
5.3.2.2 Nominalphrasen

Mit zwölf Produktionen lassen sich GB-kompatible Nominalphrasen generieren:

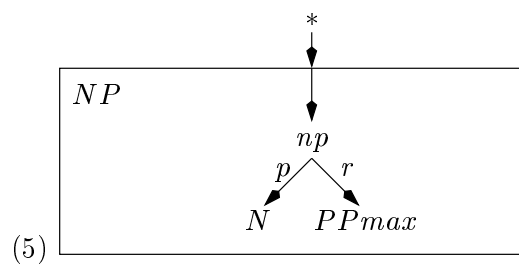
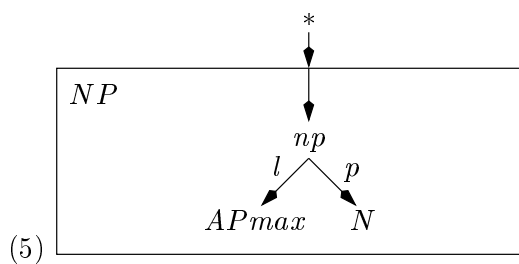
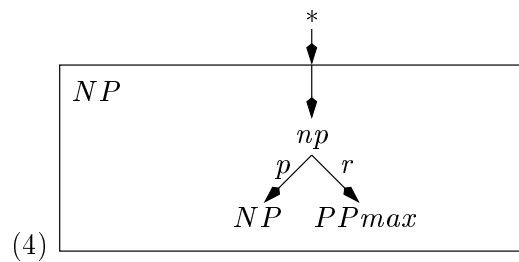
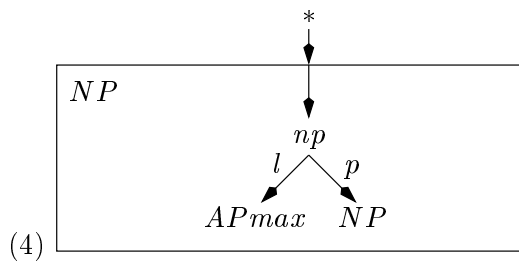
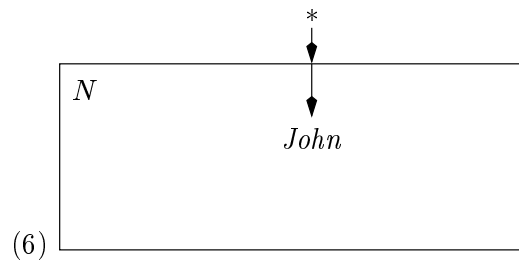
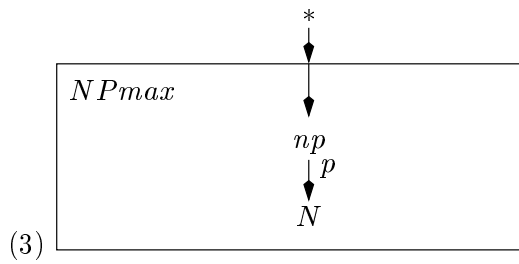
zusätzliche terminale Kantenbezeichner : l, r, p

zusätzliche Nichtterminalknoten : $NP_{max}, NP, AP_{max}, PP_{max}, N$

zusätzliche Terminalknoten : $np, John$ ¹⁸



¹⁸ Zusätzlich zu „John“ kann das Alphabet um endlich viele Substantive erweitert werden.



5.3.2.3 Adjektivphrasen/Adverbialphrasen

Adjektivphrasen und Adverbialphrasen unterscheiden sich nicht in ihrer internen Struktur, sondern durch die Art und Weise, wie sie im Satz verwendet werden. Eine Adjektivphrase erläutert Eigenschaften einer Nominalphrase, während eine Adverbialphrase Eigenschaften einer Verbalphrase oder rekursiv einer weiteren *AP* umschreibt.

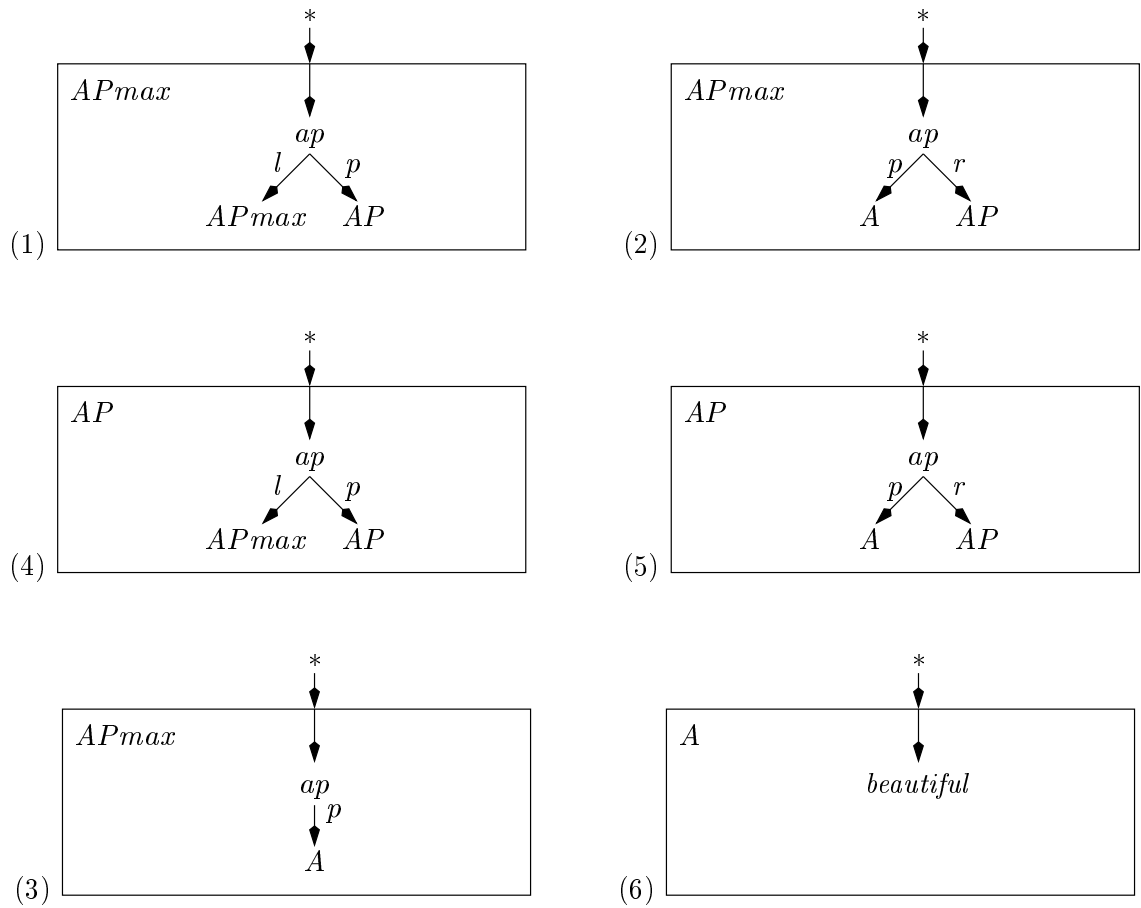
Der Unterschied wird deutlich in den Sätzen „*The beautiful girl is dancing*“ und „*The girl is dancing beautifully*“ und wird in englischen Sätzen markiert durch den Suffix *-ly*.¹⁹

zusätzliche Nichtterminalknoten : *AP*, *A*

zusätzliche Terminalknoten : *ap*, *beautiful*²⁰

¹⁹ In der deutschen Sprache ist dieser Unterschied morphologisch nicht sichtbar.

²⁰ Zusätzlich zu „*beautiful*“ kann das Alphabet um endlich viele Adjektive erweitert werden.

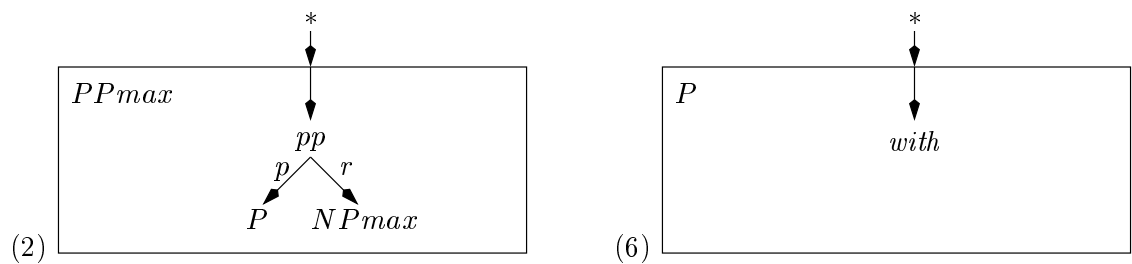


5.3.2.4 Präpositionalphrasen

Eine Präpositionalphrase verknüpft eine Präposition mit einer Nominalphrase. Da das fünfte Prinzip keine rekursive Verfeinerung zulässt, reichen in diesem Falle zwei Produktionen aus.

zusätzliche Nichtterminalknoten : P

zusätzliche Terminalknoten : $pp, with^{21}$



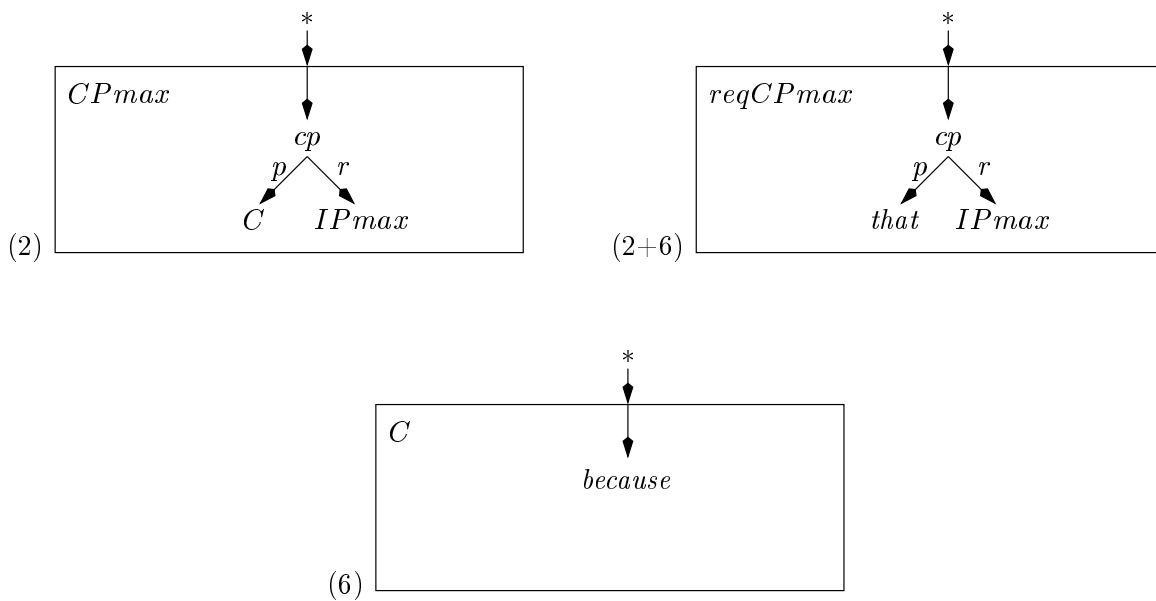
²¹ Zusätzlich zu „*with*“ kann das Alphabet um endlich viele Präpositionen erweitert werden.

5.3.2.5 Die funktionale Kategorie C

Aufgrund der Nachbarschaftsprinzipien ist hier zwischen obligatorischen und optionalen CP s zu unterscheiden. Auch hier ist wie bei den PP s keine rekursive Verfeinerung von Phrasen zugelassen. Deshalb reichen wenige Produktionen aus:

zusätzliche Nichtterminalknoten : $CPmax$, $reqCPmax$, C , $IPmax$

zusätzliche Terminalknoten : cp , $that$, $because$ ²²

5.3.2.6 Die funktionale Kategorie I

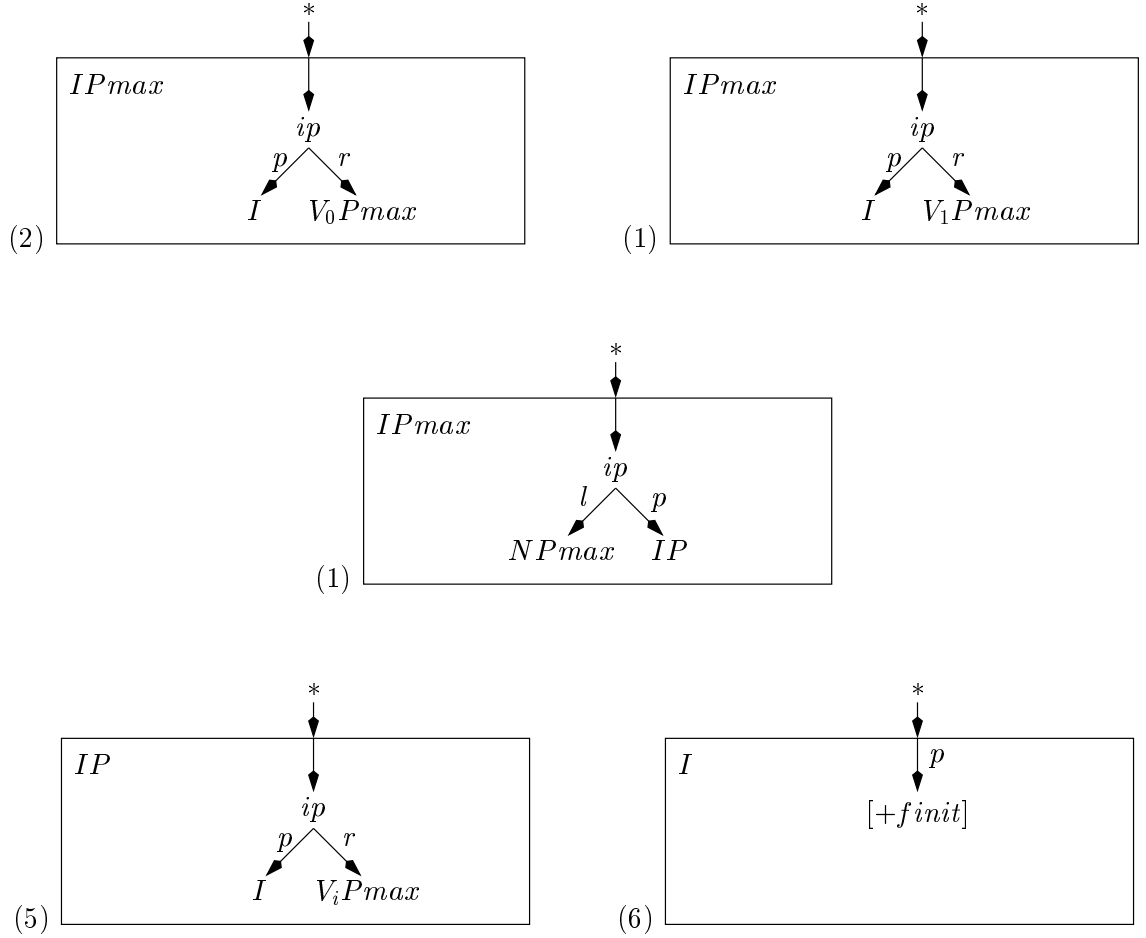
Prinzip 5 lässt auch hier keine rekursive Verfeinerung zu, Produktionen vom Typ (3), (4) und (7) kommen also nicht vor.

zusätzliche Nichtterminalknoten : I , IP , V_0Pmax , V_1Pmax , V_2Pmax , V_3Pmax , V_4Pmax , V_5Pmax

zusätzliche Terminalknoten : ip , $[+finit]$ ²³.

²² Zusätzlich zu „*because*“ kann das Alphabet um endlich viele Konjunktionen erweitert werden.

²³ GB sieht hier noch mehr Bezeichner vor. In dem in Kapitel 4 vorgestellten Modell beschränken wir uns nur auf diesen einen Eintrag. In einem erweiterten Modell würde an dieser Stelle mit zusätzlichen Parametern entschieden, ob zum Beispiel ein Infinitiv-/ oder Fragesatz repräsentiert wird.



Der Index i bei der linken Produktion nimmt Werte aus der Menge $\{2, \dots, 5\}$ an.

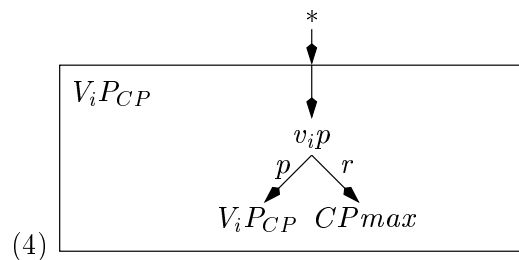
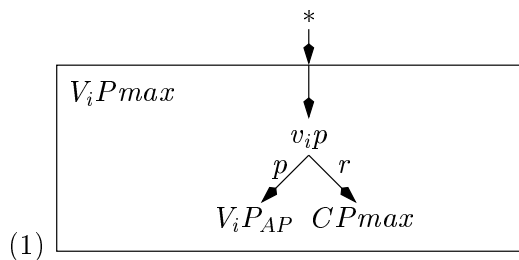
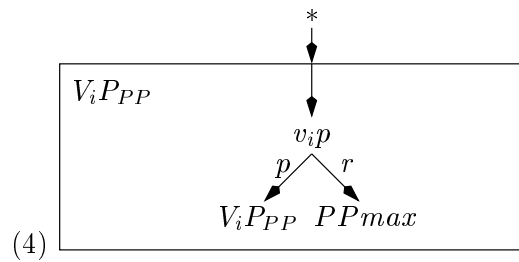
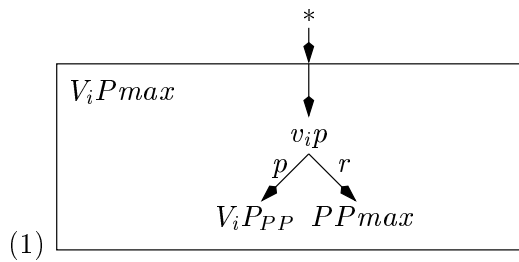
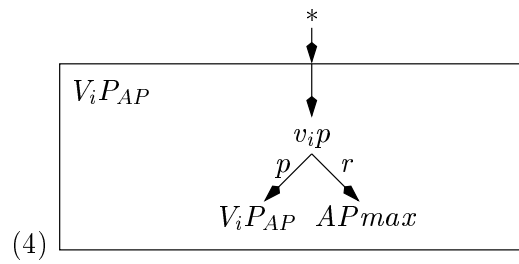
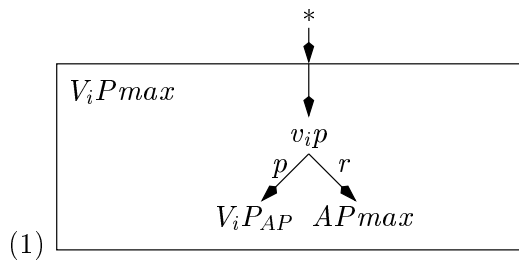
5.3.2.7 Verbalphrasen

Wegen den vielfältigen Restriktionen und Anforderungen sind die Verbalphrasen für das Syntaxgraphgerüst von besonderer Bedeutung. Um die Tabelle für die Nachbarschaftsanforderungen (fünftes Prinzip) umzusetzen, sind mehr Produktionen nötig, als bei den anderen Phrasentypen.

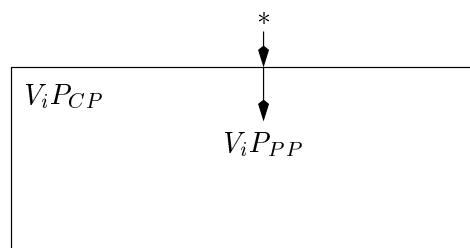
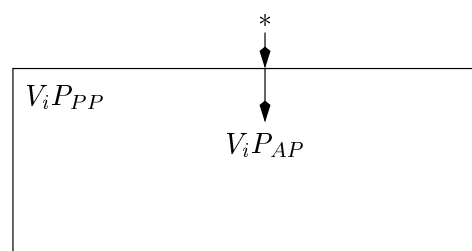
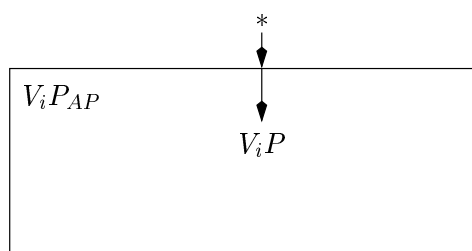
zusätzliche Nichtterminalknoten : $CP_{max}, reqCP_{max}, C, IP_{max}$

zusätzliche Terminalknoten : $v_0p, v_1p, v_2p, v_3p, v_4p, v_5p$

Der tiefgestellte Index i repräsentiert nachfolgend die Ziffern $0, \dots, 5$. Die entsprechenden Produktionen kommen also mehrfach in verschiedenen Varianten vor.

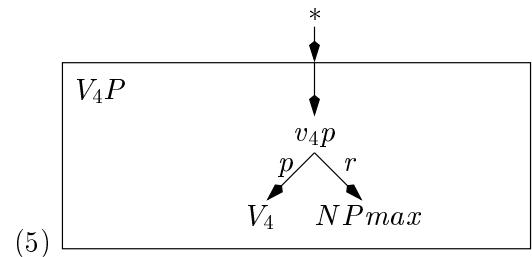
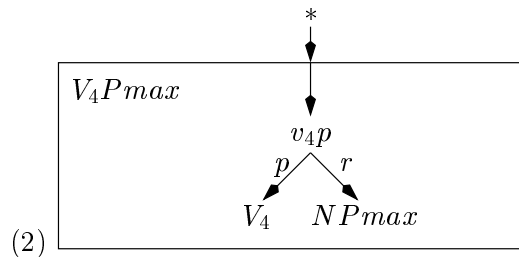
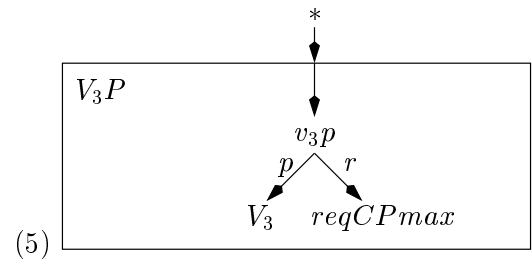
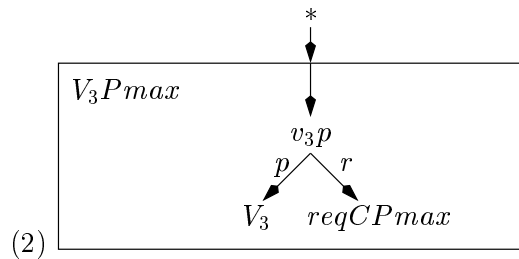
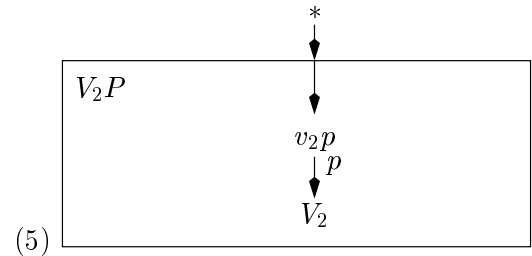
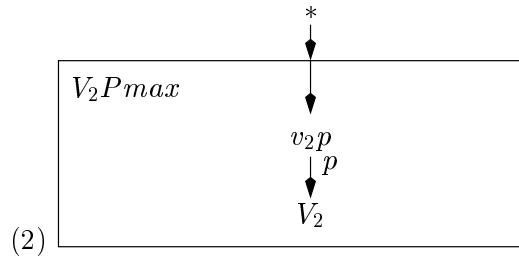
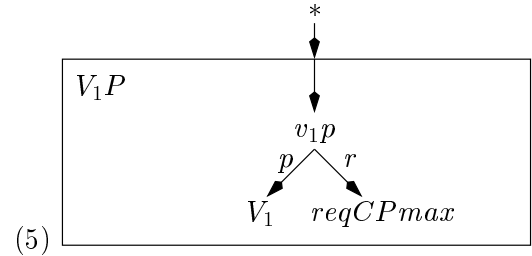
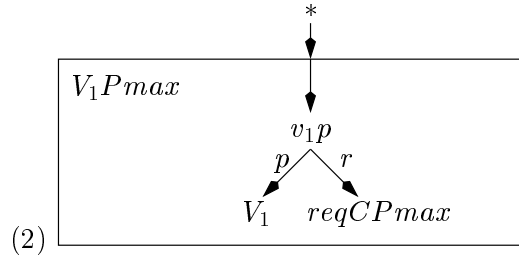


Die vorangehenden Produktionen stellen sicher, daß die Adjunkte zu einer Verbalphrase so oft wie erlaubt und in der richtigen Reihenfolge in die Graphenstruktur integriert werden. Eine VP kann durch AP s, PP s und eine CP (in dieser Reihenfolge) ergänzt werden. Da die Struktur von der Wurzel zu den Blättern hin aufgebaut wird, werden die rechts abzweigenden Adjunkte in umgekehrter Reihenfolge generiert.

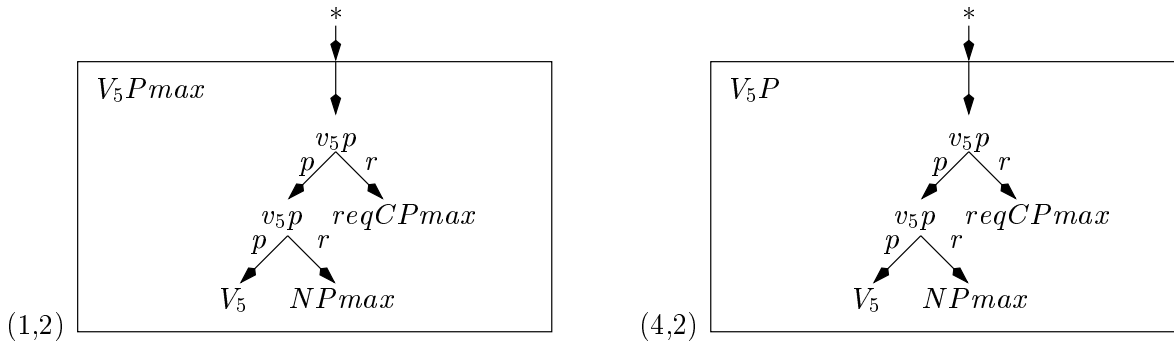


Da Adjunkte zwar realisiert werden dürfen, aber nicht realisiert werden müssen, ermöglichen es die vorangehenden Produktionen, deren Generierung zu überspringen.

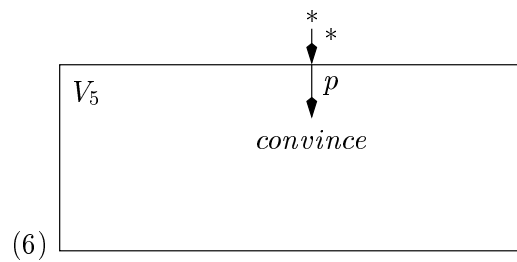
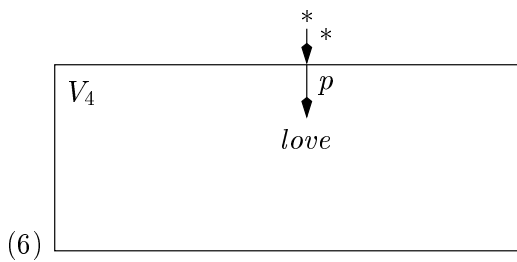
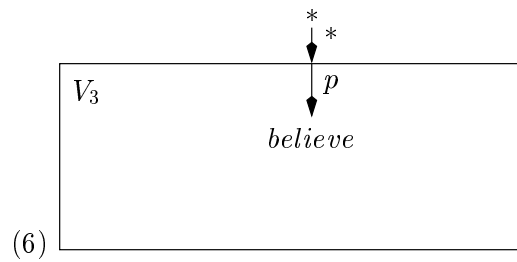
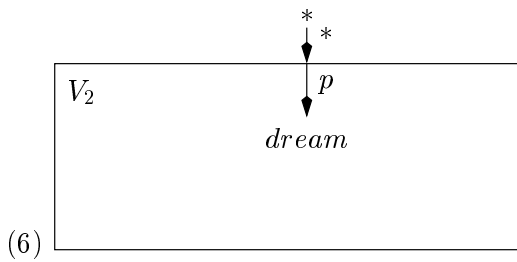
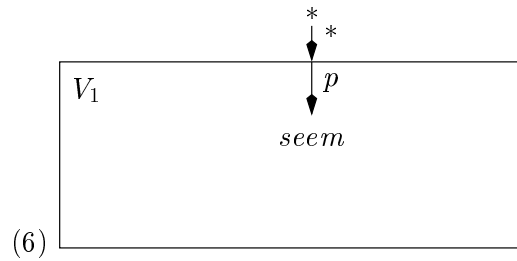
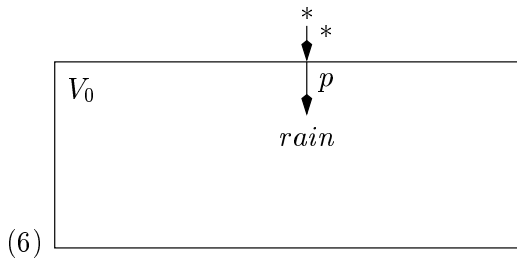
Abhängig vom Typ des Verbs wird unterschiedlich expandiert²⁴:



²⁴ Da der Typ V_0 keine Komplemente subkategorisiert, brauchen wir dafür auch keine Produktionen vorzusehen.



Schließlich ist noch die *lexikalische Einsetzungsregel* typabhängig zu spezifizieren. Jeder Typ ist durch eine andere Stelligkeit charakterisiert, subkategorisiert andere Komplemente. Die folgenden Produktionen sind als Repräsentant für (endlich) viele Produktionen zu verstehen.



5.4 Eigenschaften von GG_{deriv}

Alle von der Graphgrammatik GG_{deriv} beschriebenen Graphen sind Syntaxgraphen:

SATZ 26: $L(GG_{deriv}) \subseteq \mathcal{SG}$.

BEWEIS: Für einen Graphen $G \in L(GG_{deriv})$ sind alle fünf Prinzipien erfüllt:

1. Prinzip 1: Alle Produktionen sind vom Typ der Graphgrammatik GG_{X-Bar} bis auf die umfangreichen Produktionen bei den Verbalphrasen auf Seite 125 und den Umbenennungen auf Seite 123. Erstere sind unproblematisch, da sie sich als Komposition zweier Produktionen aus GG_{X-Bar} unter Anwendung eines Refinement-Schrittes ansehen lassen. Die Umbenennungen sind dahingehend unproblematisch, als daß sie nichts an der Struktur des Graphen verändern.
2. Prinzip 2: Die Kanten der terminalen Graphen haben entweder die Bezeichnung l , p oder r (von dem Symbol q wird nicht Gebrauch gemacht, es kann also gar nicht falsch verwendet werden).
3. Prinzip 3: Die terminalen Knotenbezeichnungen sind alles zugelassene Namen, die einem Typbezeichner oder einem Element aus dem Lexikon entsprechen. Auch treten sie nur an zulässigen Positionen auf. Die in Prinzip 3 ausgesprochenen Restriktionen sind erfüllt.
4. Prinzip 4: Indizes sind durch Querkanten ersetzt worden. Die entsprechenden Restriktionen sind trivialerweise erfüllt, weil in der Graphgrammatik GG_{deriv} gar keine Querkanten generiert werden. Es kann also keine falschen Querkanten geben.
5. Prinzip 5: Alle realisierbaren Nachbarschaften genügen den Restriktionen der Tabellen zum fünften Prinzip. Die Anforderungen für die Reihenfolge sind dahingehend umgesetzt, daß die Kantenbeschriftungen entsprechend vergeben sind. Die von GG_{deriv} generierten Graphen sind zwar ungeordnet, enthalten aber auf Grund der Kantenbeschriftung die für eine korrekte Wortstellung notwendige Reihenfolgeinformation.

□

LEMMA 27: GG_{deriv} ist boundary (und damit auch konfluent).

BEWEIS: Es ist einfach zu sehen, daß die Produktionen von GG_{deriv} keine adjazenten nichtterminalen Knoten besitzen. Damit ist GG_{deriv} boundary woraus unmittelbar die Konfluenzeigenschaft folgt.

□

Die Menge $L(GG_{deriv})$ enthält unendlich viele GB-kompatible Graphen. Allerdings werden eine Reihe von Phänomenen, die GB-kompatibel sind, von der Graphgrammatik nicht modelliert. So generiert die vorgestellte Grammatik nur Syntaxgraphen ohne Querkanten. Dies läßt sich unmittelbar aus der Gestalt der Produktionen ableiten, bei denen die rechten Seiten aus expandierenden Baumstrukturen bestehen. Die Graphen aus $L(GG)$ sind ohne Ausnahme Bäume.

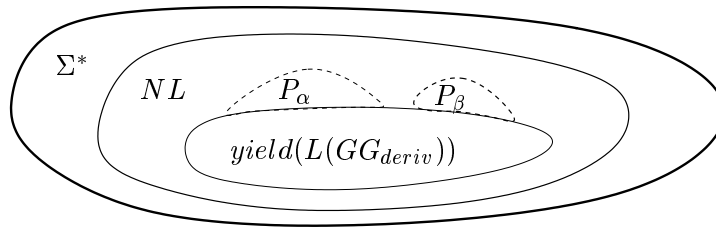


Abbildung 5.12: Die Graphgrammatik GG_{deriv} approximiert Teilmengen von NL

5.5 Zusätzliche Produktionen modellieren Querkanten

Die Graphgrammatik GG_{deriv} modelliert nur eine Teilmenge von NL , wie es das Inklusionsdiagramm 5.12 deutlich macht.

In der Zeichnung wird die weitere Vorgehensweise dahingehend deutlich, daß zusätzliche Produktionen P_α für α -Ketten und P_β für β -Ketten die Menge der beschreibbaren Syntaxgraphen vergrößern sollen.

Zur Erinnerung der Rolle von Ketten und Querkanten seien hier noch einmal wesentliche Aspekte zusammengefaßt, die ausführlich im Abschnitt 4.4.1 eingeführt wurden.

1. Eine Querkante kommt nur in einem wohldefinierten strukturellen Kontext vor, der als Kette bezeichnet wird.
2. Eine Kette kann unbeschränkt viele Querkanten enthalten. Ketten können beliebig lang sein.
3. Es kann unbeschränkt viele Ketten in einem Syntaxgraphen geben. Diese Eigenschaft hat einen Einfluß auf die Anzahl der Querkanten im Syntaxgraphen.
4. Unbeschränkt viele Ketten können „parallel“ im Syntaxgraphen laufen. Diese Eigenschaft hat Einfluß auf den Separator des Syntaxgraphen.

Die Menge der Querkanten ist also in mindestens dreierlei Hinsicht unbeschränkt. Aus der vierten Eigenschaft folgt (Satz 25), daß keine konfluente Graphgrammatik existiert, welche die Menge \mathcal{SG} aller Syntaxgraphen mit Querkanten erschöpfend modelliert. Diese Beschränkung ist jedoch praktisch irrelevant und wie sich zeigt, lassen sich Syntaxgraphen vernünftig mit konfluenten Graphgrammatiken approximieren.

Einige der denkbaren Erweiterungen von GG_{deriv} werden nachfolgend getrennt demonstriert, um die jeweils zur Anwendung kommenden Mechanismen besser zu verdeutlichen. Eine gemeinsame Beschreibung aller Phänomene in einer einzigen (konfluenten) Graphgrammatik ist möglich, würde die Mechanismen aber zu sehr verschleiern, weil die Grammatik dann wegen der notwendigen zusätzlichen Symbole sehr groß und unübersichtlich würde. Die Produktionen für einige exemplarisch umgesetzte linguistische Regularitäten auf den folgenden Seiten machen das deutlich.

Folgende Erweiterungen werden diskutiert:

1. Querkanten für Pronomen und Anaphern (β -Ketten) (Abschnitt 5.5.1)
2. Querkanten für Bewegungsphänomene (α -Ketten) (Abschnitt 5.5.2)
3. Beide Typen von Querkanten gemischt (α -Ketten und β -Ketten) (Abschnitt 5.5.3)
4. Kontextsensitive Erscheinungen (Abschnitt 5.5.4)

Die β -Ketten werden vor den α -Ketten erläutert, weil ihre Modellierung mit weniger Aufwand verbunden ist.

5.5.1 Erweiterung 1: Querkanten für bindungstheoretische Regularitäten

Um auch solche Syntaxgraphen zu beschreiben, die eine korrekte Verwendung von Pronomen und Anaphern mit β -Ketten sicherstellen, ergänzen wir GG_{deriv} um die Menge der Produktionen P_β . Die bestehenden Produktionen aus GG_{deriv} bleiben alle erhalten.

Die Ergänzungen betreffen die Produktionen für CP s, IP s und VP s. Sie müssen sicherstellen, daß die lokalen Bedingungen und Restriktionen für β -Ketten, welche Prinzip 4 formuliert, auch tatsächlich immer eingehalten werden.

Grundidee: Bereits bei der Generierung einer Nominalphrase wird entschieden, ob auf diese später im Ableitungsprozeß noch einmal referiert werden soll. Die aktuelle Information über die Entfernung zur NP während der Ableitung, die letztlich über die Realisierung einer Referenz als Pronomen oder Anapher entscheidet, wird in das Alphabet hineincodiert. Zwei Phasen sind dabei zu unterscheiden:

1. Die Entfernung zur NP ist „kurz“. Die Referenz ist dann als ANAPHER zu realisieren. Dieser Zustand kommt durch das Indexsymbol β zum Ausdruck.
2. Die Entfernung zur NP ist „lang“. Die Referenz ist dann als PRONOMEN zu realisieren. Dieser Zustand kommt durch das Indexsymbol β' zum Ausdruck.

Im Rahmen des Ableitungsprozesses wird die Indexinformation über den aktuellen Zustand gleichsam „nach unten“ weitergegeben. Gleichzeitig wird mit der Ableitungsinformation auch eine Referenz auf eine Querkante mittransportiert. Wird schließlich ein Knoten als PRONOMEN oder ANAPHER terminal, dann ergibt sich gleichzeitig eine Querkante, die den Anforderungen von Prinzip 4 genügt.

5.5.1.1 Neue Produktionen für die funktionale Kategorie I

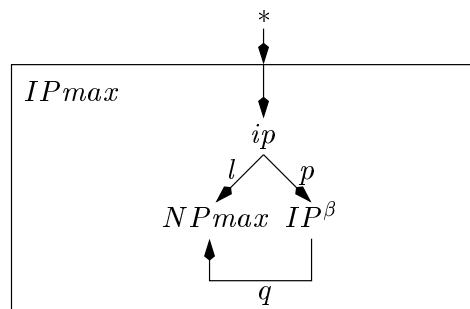
An dieser Erweiterung wird bereits deutlich, daß die Modellierung zusätzlicher Restriktionen Folgen für den Umfang der Symbolmenge hat:

zusätzliche Nichtterminalknoten : $IPmax^\beta, V_0Pmax^\beta, V_1Pmax^\beta, V_1Pmax^\beta,$
 $V_2Pmax^\beta, V_3Pmax^\beta, V_4Pmax^\beta, V_5Pmax^\beta, IPmax^{\beta'}, V_0Pmax^{\beta'}, V_1Pmax^{\beta'},$
 $V_1Pmax^{\beta'}, V_2Pmax^{\beta'}, V_3Pmax^{\beta'}, V_4Pmax^{\beta'}, V_5Pmax^{\beta'}, IP^\beta, IP^{\beta'}$

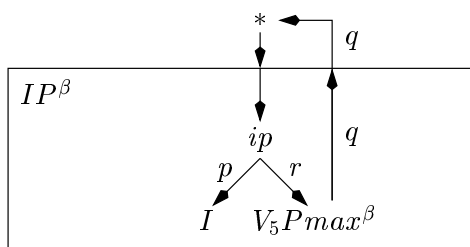
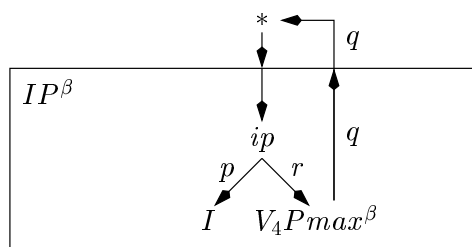
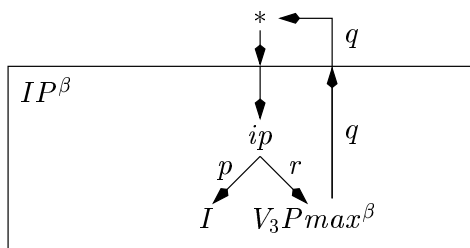
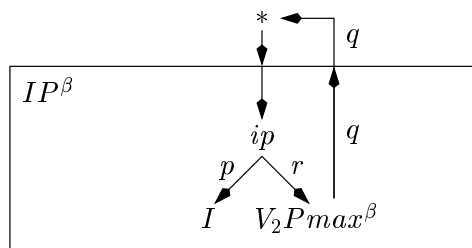
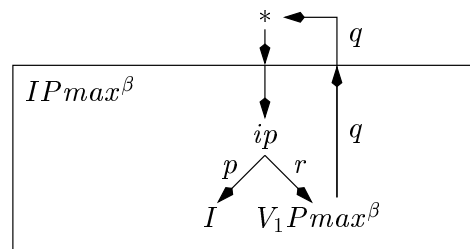
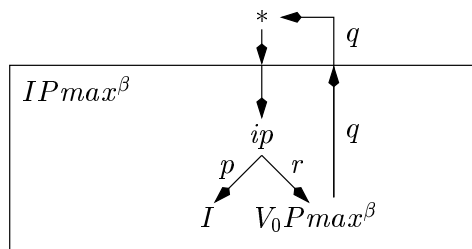
zusätzlicher Terminalknoten : PRONOMEN

zusätzliche Terminalkante : q

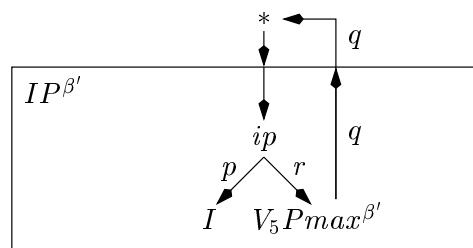
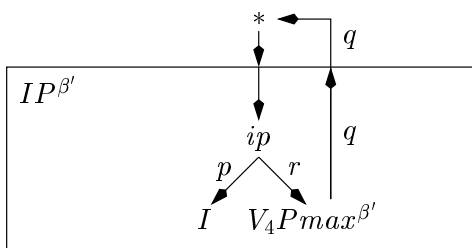
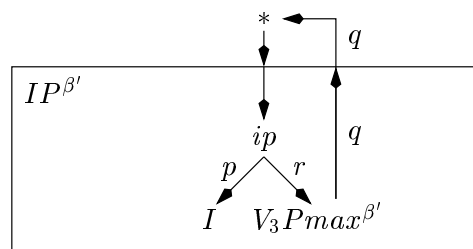
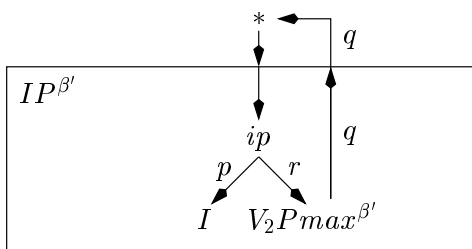
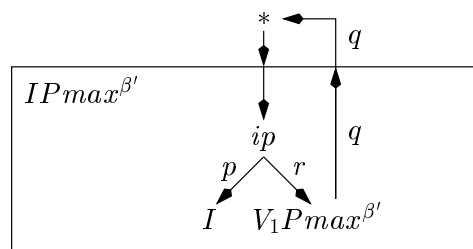
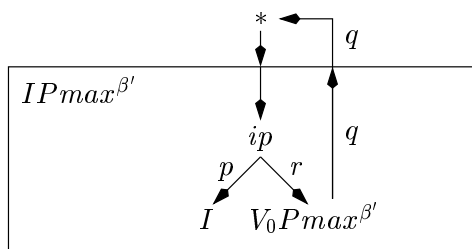
Die erste Produktion generiert die maximale Projektion einer Nominalphrase und erzeugt darüber hinaus eine Querkante, über die im weiteren Ableitungsprozeß auf die NP referiert werden kann:



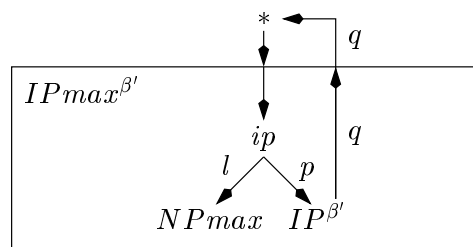
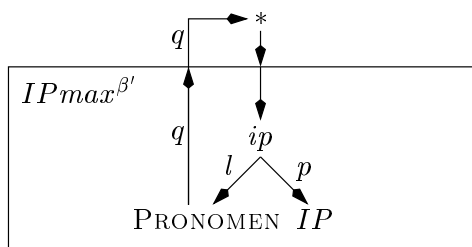
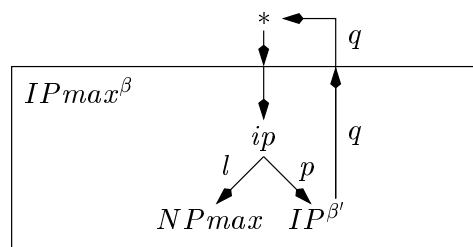
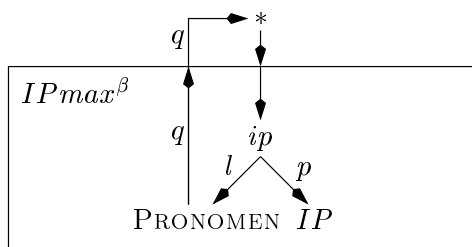
Weitere Produktionen bereiten die Expansion eine VP vor:



Noch einmal dieselben sechs Produktionen mit dem Index β' :



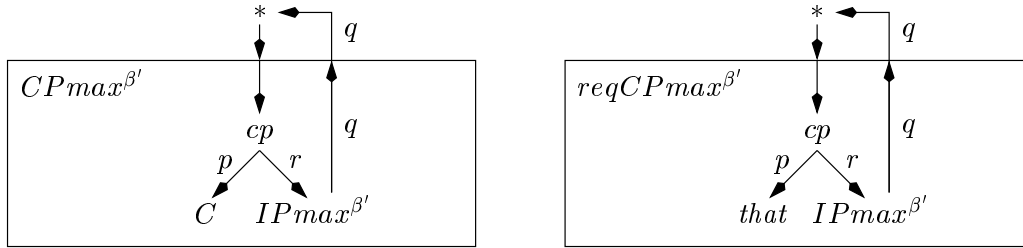
Bei der Eröffnung einer neuen IP kann eine Referenz als PRONOMEN realisiert oder die Referenz für eine spätere Auswertung weitergegeben werden.



5.5.1.2 Neue Produktionen für die funktionale Kategorie C

zusätzliche Nichtterminalknoten : $CPmax^{\beta'}$, $reqCPmax^{\beta'}$

Die beiden folgenden Produktionen leiten einen Nebensatz ein und transportieren die Indexinformation im Graphen weiter „nach unten“.



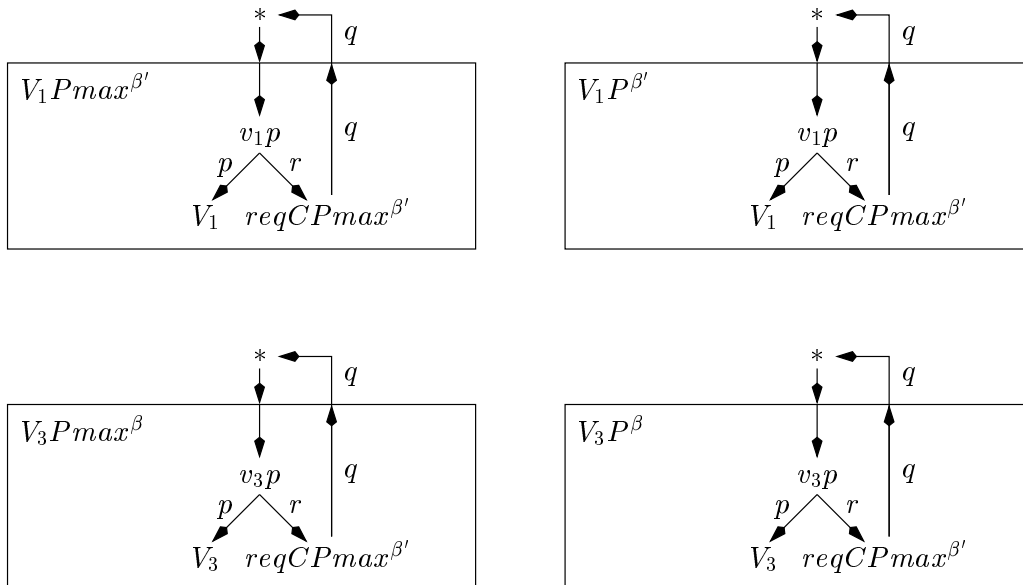
Da die maximale Projektion einer CP einen Wechsel von β nach β' erzwingt, braucht hier keine Version der beiden Produktionen für das Indexsymbol β vorgesehen werden.

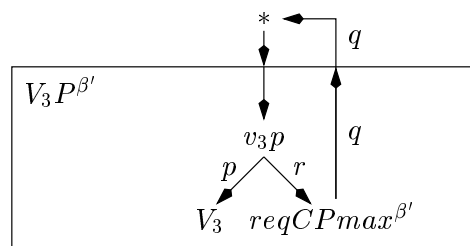
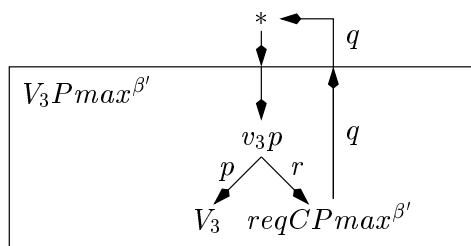
5.5.1.3 Neue Produktionen für Verbalphrasen

zusätzliche Nichtterminalknoten : $V_iP_{AP}^{\beta}$, $V_iP_{PP}^{\beta}$, $V_iP_{CP}^{\beta}$

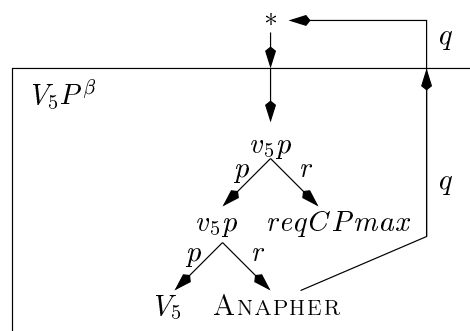
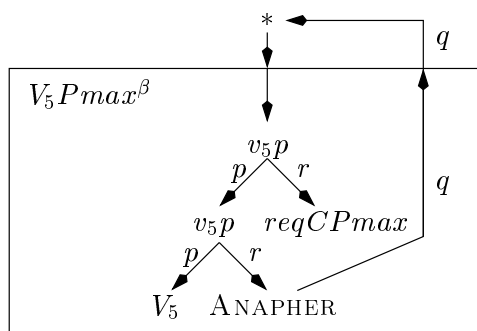
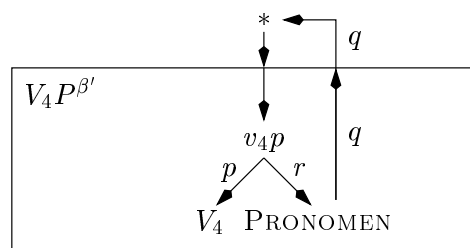
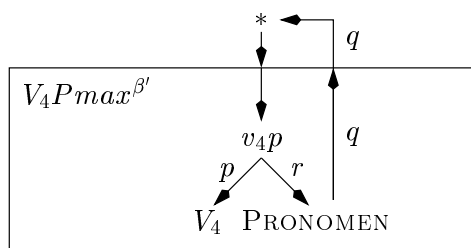
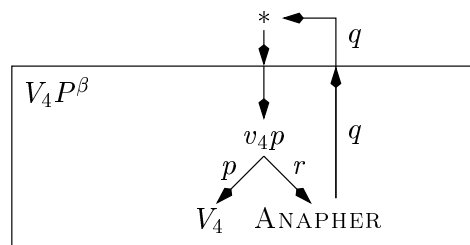
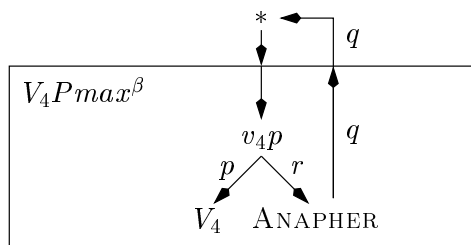
zusätzliche Terminalknoten : ANAPHER

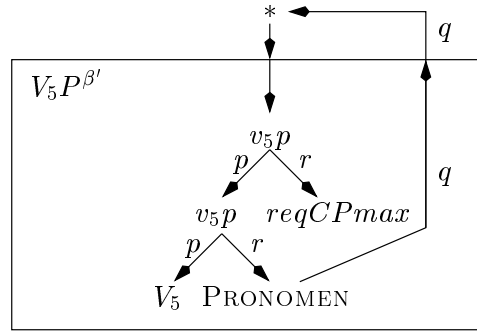
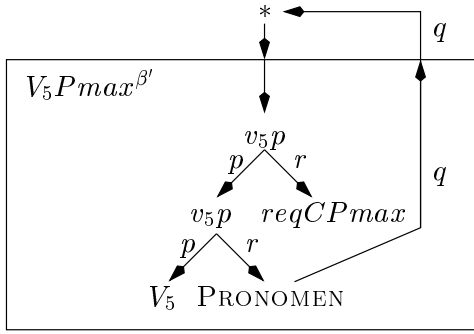
Abhängig vom Typ des Verbs wird unterschiedlich expandiert. Bei der Subkategorisierung von einem Nebensatz bei den Typen V_3 und V_5 kommt es zu einem Moduswechsel von β zu β' . Der Typ V_1 darf nicht in der Version V_1Pmax^{β} vorkommen, weil in der Specifier Position der dominierenden IP kein Subjekt steht (Prinzip 5):



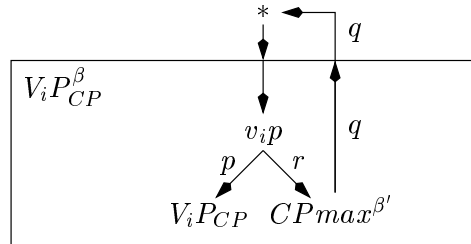
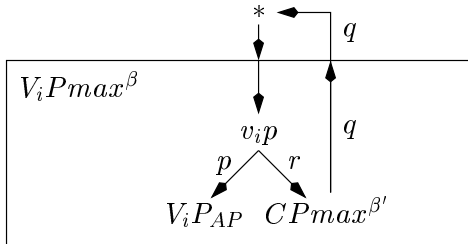
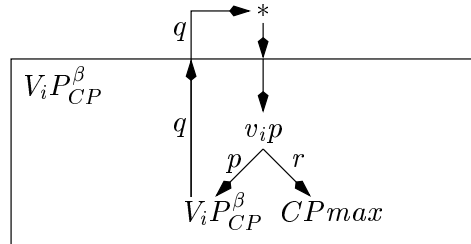
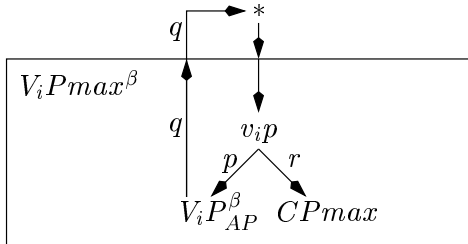
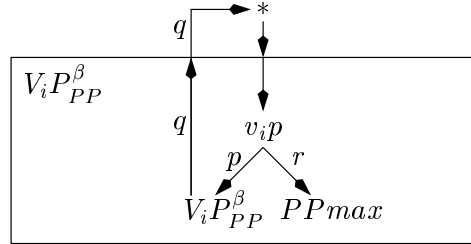
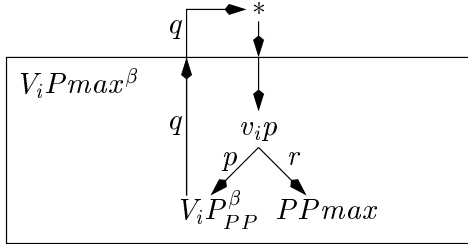
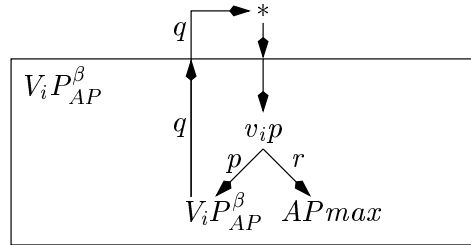
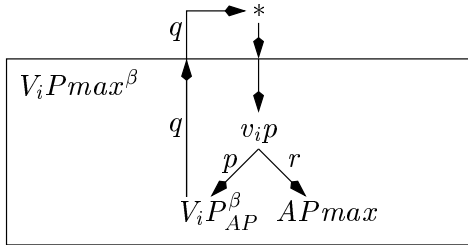


Die Kategorien V_4 und V_5 subkategorisieren eine NP , die im Rahmen einer Kette als Pronomen oder Anapher realisiert werden kann. Welche von beiden Alternativen zur Anwendung kommt, hängt von der Kontextsituation ab, die durch die Indizes β und β' zum Ausdruck kommt. An dieser Stelle wird also die Information der Indizes ausgewertet:

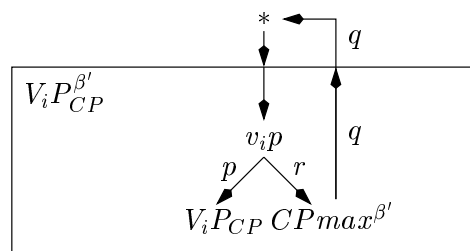
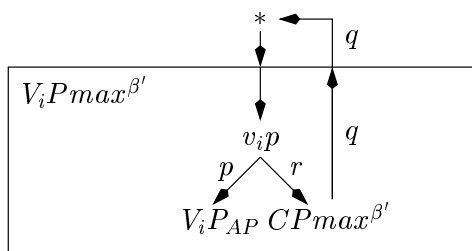
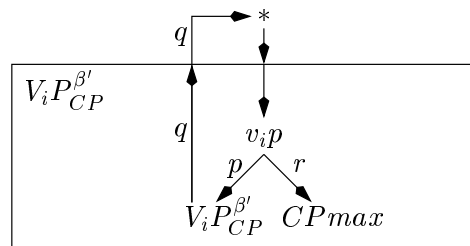
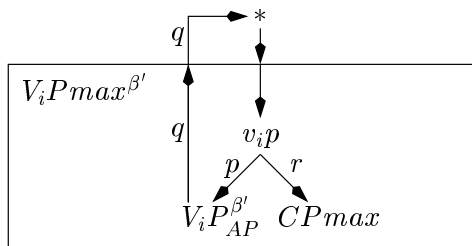
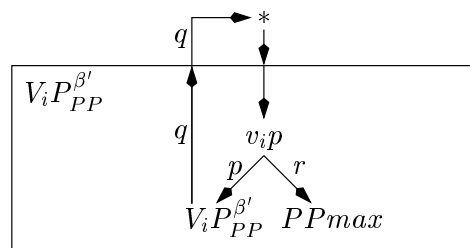
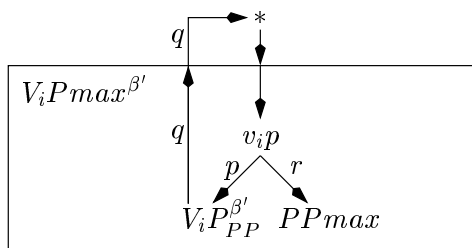
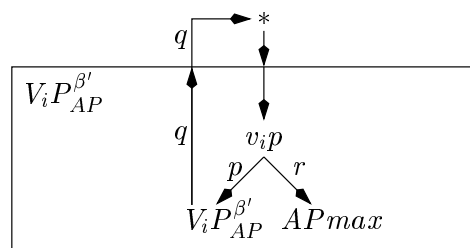
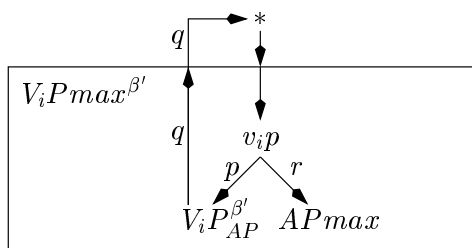




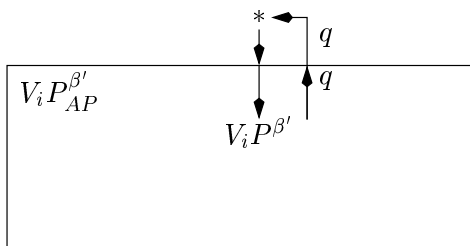
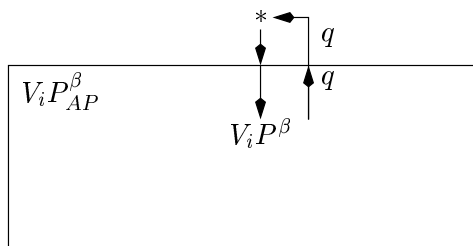
Fallweise kommen in der Satzstruktur Adjunkte zur Anwendung. Diese sind wieder für den Fall β und β' zu modellieren. Der Index i bei den folgenden Produktionen repräsentiert jeweils Werte $0 \leq i \leq 5$.

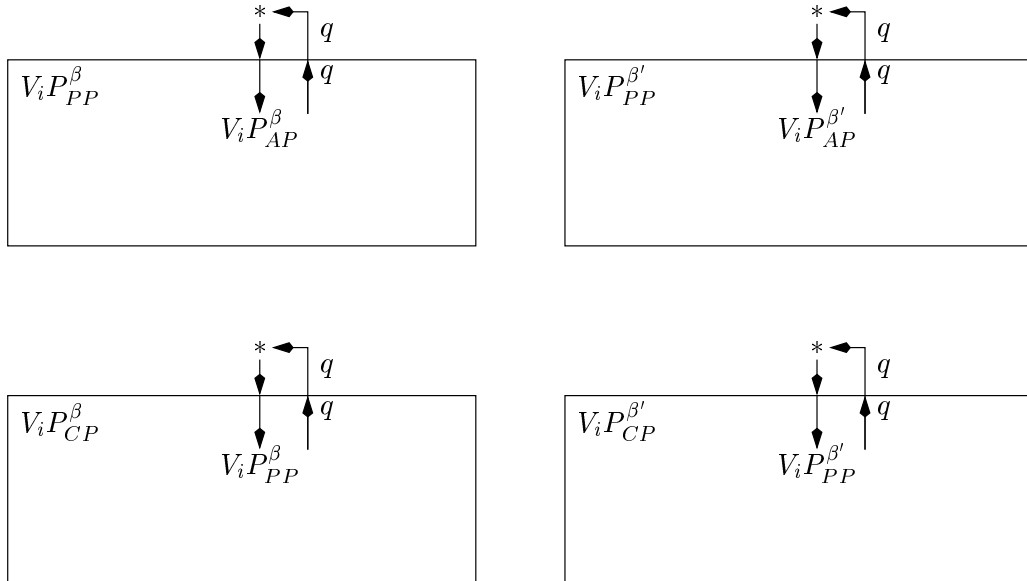


Entsprechend mit β' :



Auch bei Umbenennungen ist die Weitergabe der Informationen β und β' sicherzustellen:





Mit diesen zusätzlichen Produktionen für (einige) β -Ketten sollte deutlich geworden sein, wie sich die Restriktionen für Bindungsprinzipien (viertes Prinzip) in den Formalismus einer Graphgrammatik übersetzen lassen.

5.5.1.4 Eigenschaften der zusätzlichen Produktionen

Sei nun $GG_{deriv-\beta}$ eine Graphgrammatik, die auf der Basis von GG_{deriv} und den in Abschnitt 5.5.1 vorgestellten zusätzlichen Produktionen P_β besteht. $L(GG_{deriv-\beta})$ enthält ausschließlich GB-kompatible Graphen. Für das zugrundeliegende Gerüst ist dies klar. Da wir dort keine Änderungen vorgenommen haben, gilt das bereits für GG_{deriv} gezeigte. Letztlich fügen die neuen Produktionen nur weitere Kanten hinzu und zwar solche Querkanten, die den Restriktionen für β -Ketten genügen. Auch $GG_{deriv-\beta}$ ist konfluent, wie sich leicht überprüfen läßt (boundary-Eigenschaft).

In mehrerlei Hinsicht reizt die vorgestellte Ergänzung aber noch nicht das aus, was β -Ketten erlauben. Grundsätzlich wurde bereits gezeigt, daß eine konfluente Graphgrammatik nicht die Menge aller Syntaxgraphen generieren kann, weil bei beschränktem Grad auch nur ein beschränkter Separator generiert werden kann. Darüber hinaus wurden hier noch weitere Möglichkeiten von β -Ketten ignoriert, deren Umsetzung zusätzliche Produktionen erfordern würde:

Beispielsweise wird nur die Indexinformation β/β' für eine β -Kette modelliert. Deshalb können Sätze mit verschränkten Referenzen nicht modelliert werden wie bei „*John $_\alpha$ tells Mary $_\beta$ that he $_\alpha$ loves her $_\beta$.*“. Unverschränkt sind aber schon mehrere R-Ausdrücke pro Satz möglich: Ein Satz wie „*John $_\alpha$ tells himself $_\alpha$ that Mary $_\beta$ loves herself $_\beta$.*“ ist syntaktisch korrekt und kann von den Produktionen in P_β modelliert werden. Für eine Erweiterung auf verschränkte Ketten ist das Indexalphabet und die Menge der Produktionen entsprechend zu erweitern.

5.5.2 Erweiterung 2: Querkanten für Bewegungen

Bewegungsphänomene werden durch α -Ketten modelliert. Unsere Darstellung modelliert mehrere Bewegungsphänomene. Um das Prinzip der Modellierung von α -Ketten durch Graphgrammatiken anschaulich zu demonstrieren, nehmen wir eine weitere Beschränkung vor und zeigen exemplarisch, wie die *Topikalisierung von obligatorischen Nebensätzen*²⁵ beschrieben werden kann. Wie auch im Fall der β -Ketten können nötige Erweiterungen mit Hilfe zusätzlicher Alphabetsymbole modelliert werden.

Im Falle der Topikalisierung muß die Menge der zusätzlichen Produktionen P_α insbesondere sicherstellen, daß die Subjazenbedingung für alle ableitbaren α -Ketten eingehalten wird. Die Umsetzung dieser Restriktion erfordert den meisten Aufwand bei der Erstellung der Graphgrammatik.

Graphgrammatiken kennen keine Transformationsregeln für Graphen, wie sie in der GB-Literatur beschrieben werden. Der Vorgang muß also auf andere Weise modelliert werden. Eine Analyse der Bewegungsrestriktionen führt zu dem Ergebnis, daß Bewegungen in einer syntaktischen Struktur nur zur Wurzel hin stattfinden. Dies machen wir uns nachfolgend zunutze:

Ein Syntaxgraph mit einem topikalisiertem Nebensatz beginnt *immer* mit einer CP-Phrase. Dies eröffnet die Möglichkeit, im Rahmen einer entsprechenden Produktion eine (gemessen am Ableitungsprozeß später stattfindende) Bewegung vorzubereiten, indem „Platzhalter“ für die späteren Querkanten generiert werden. Abhängig von der Entfernung, die von einem Bewegungsprozeß im Syntaxgraphen überwunden wird, sind im Ableitungsprozeß drei Phasen zu unterscheiden:

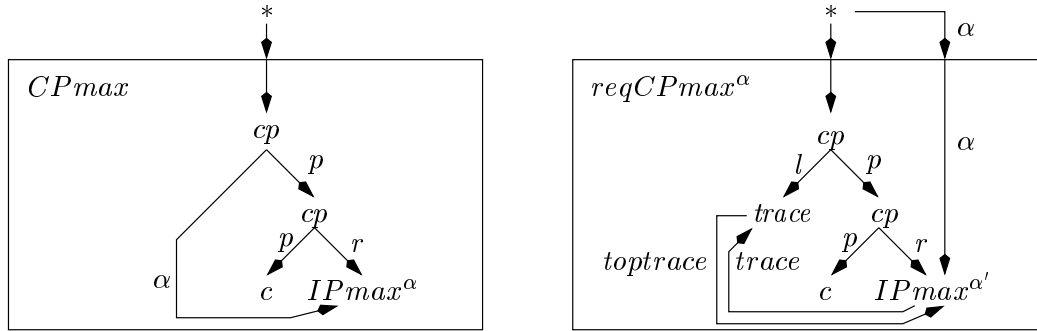
1. Die Bewegung erfolgt nur lokal. Es braucht keine Spur in den Syntaxgraphen integriert zu werden. Eine Querkante verbindet Start- und Landeplatz der Bewegungsoperation.
2. Es wird kurz bewegt, ein Grenzknoten wird überschritten. Dann ist eine Spur anzulegen, die nach oben mit der bewegten Phrase und nach unten mit dem Startplatz der Bewegungsoperation über jeweils eine Querkante verbunden ist. Produktionen aus dieser Phase sind mit dem Indexsymbol α versehen.
3. Es wird über eine weite Entfernung bewegt: Beliebige viele zusätzliche Grenzknoten erfordern zusätzliche Spuren, die miteinander durch Querkanten verbunden sind und einen Kettengraphen bilden. Produktionen aus dieser Phase sind mit dem Indexsymbol α' versehen.

5.5.2.1 Neue Produktionen für die funktionale Kategorie C

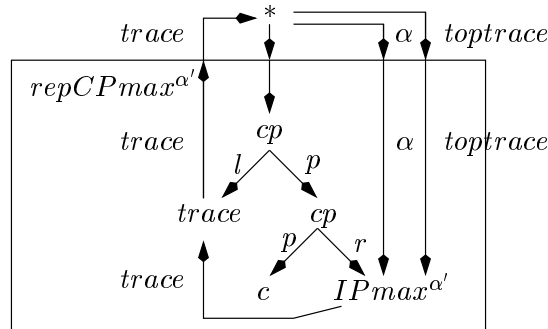
zusätzliche Nichtterminalknoten : $IPmax^\alpha, IPmax^{\alpha'}, reqCPmax^\alpha, reqCPmax^{\alpha'}$

zusätzliche Nichtterminalkanten : $\alpha, trace, toptrace$

²⁵ Vereinfachend sind obligatorische Nebensätze daran zu erkennen, daß sie mit „that“ eingeleitet werden, wie in : „Peter believes, that John loves Mary“.



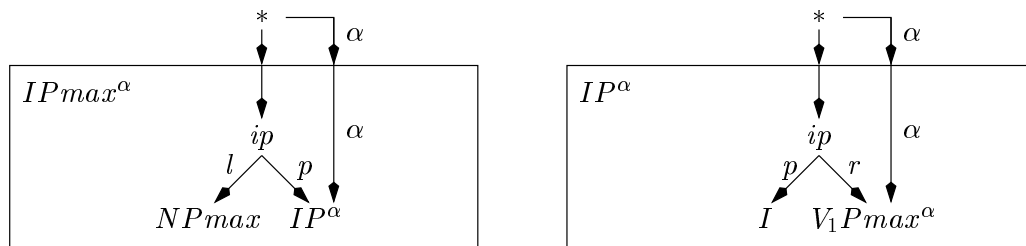
Die rechte Produktion initiiert den Übergang von Phase 1 zu Phase 2 und stellt die Einhaltung der Subjazenbedingung sicher. Dasselbe leistet die folgende Regel für den Übergang von Phase 2 zu Phase 3. Wenn bereits eine Spur vorhanden ist und der nächste Grenzknoten überschritten wird, dann ist eine neue Spur zu generieren.

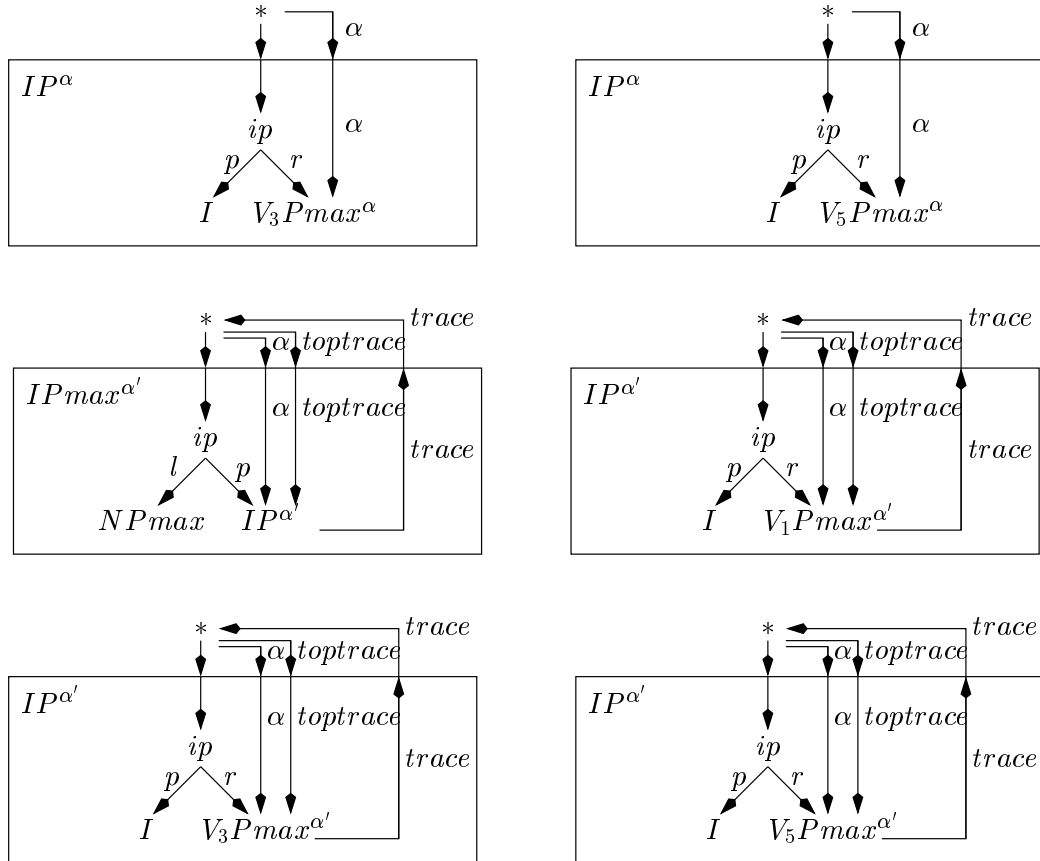


5.5.2.2 Neue Produktionen für die funktionale Kategorie *I*

Die nachfolgenden Produktionen erfüllen die Aufgabe, die Information α (α') über den Ableitungsprozeß hinweg „nach unten“ zu transportieren. Die Beschränkung auf die Typen V_1 , V_3 und V_5 folgt aus der Vorgabe, daß nur obligatorische Nebensätze modelliert werden: Nur solche CP-Phrasen lassen sich topikalisieren, die auch durch eine Subkategorisierung lizenziert sind.

zusätzliche Nichtterminalknoten : V_1Pmax^α , V_3Pmax^α , V_5Pmax^α , $V_1Pmax^{\alpha'}$, $V_3Pmax^{\alpha'}$, $V_5Pmax^{\alpha'}$, IP^α , $IP^{\alpha'}$

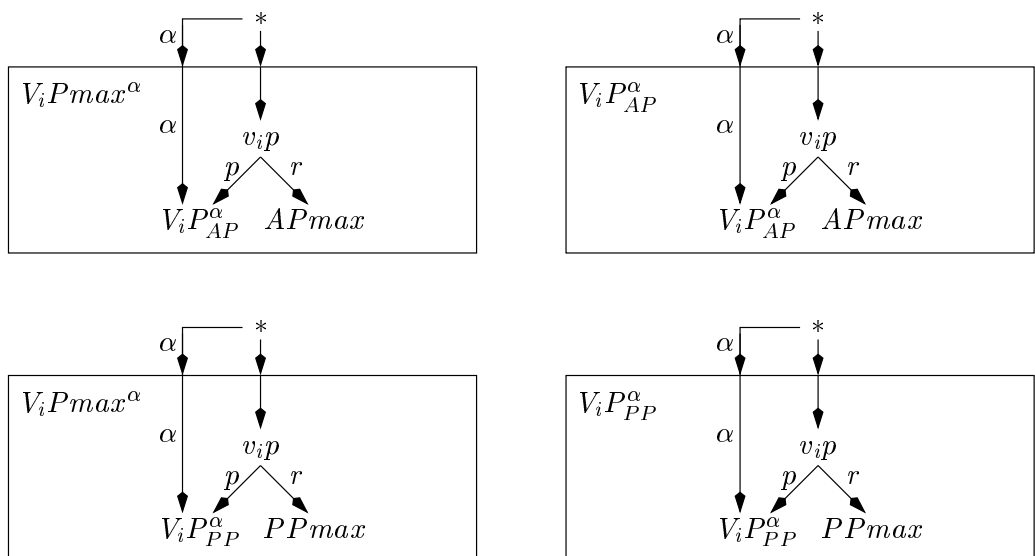


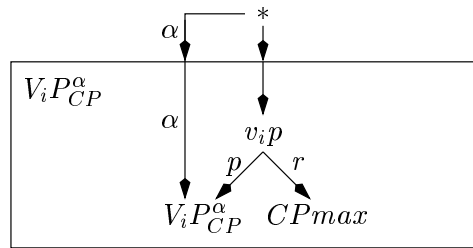
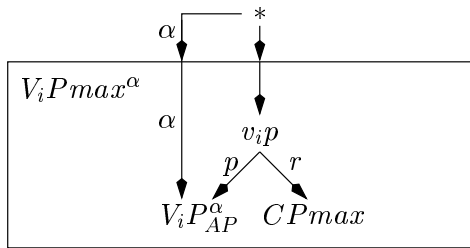


5.5.2.3 Neue Produktionen für Verbalphrasen

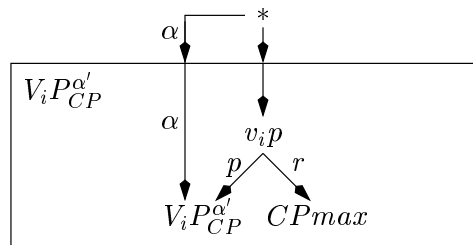
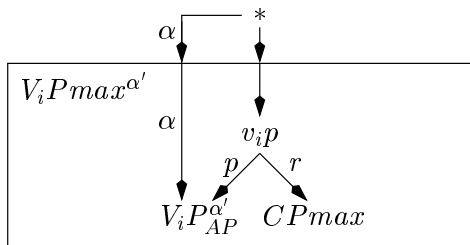
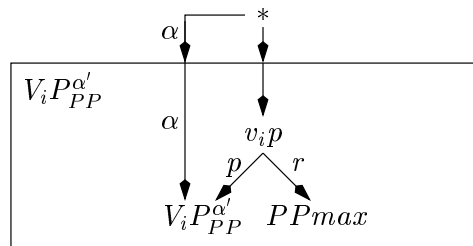
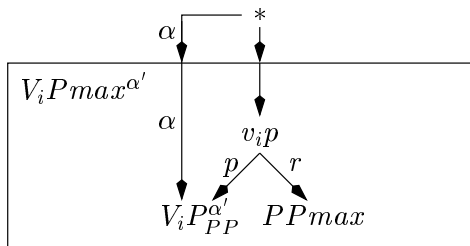
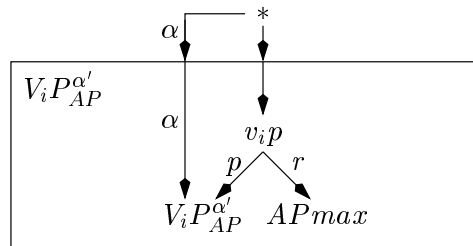
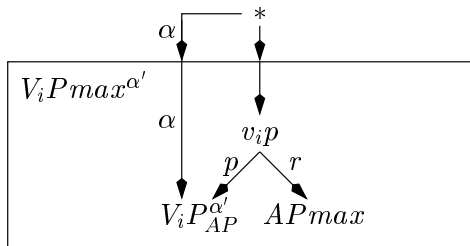
zusätzliche Nichtterminalknoten : $V_i P_{AP}^\alpha$, $V_i P_{PP}^\alpha$, $V_i P_{CP}^\alpha$, $V_i P_{AP}^{\alpha'}$, $V_i P_{PP}^{\alpha'}$, $V_i P_{CP}^{\alpha'}$

zusätzliche Terminalknoten $trace$

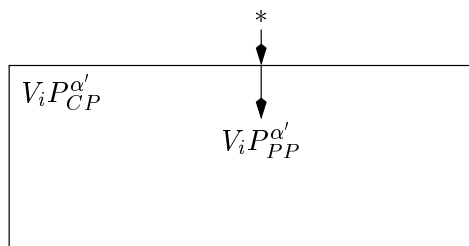
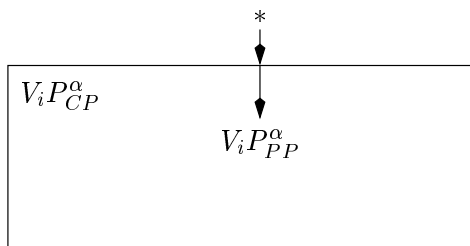


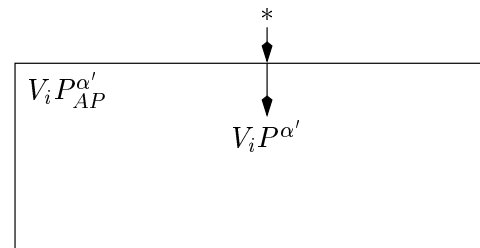
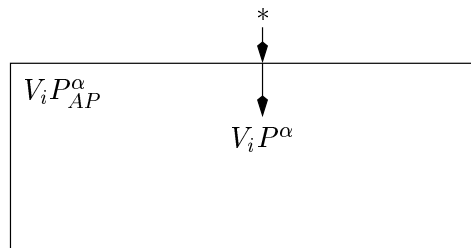
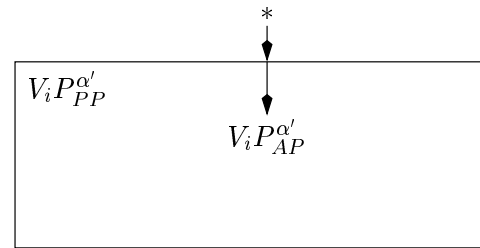
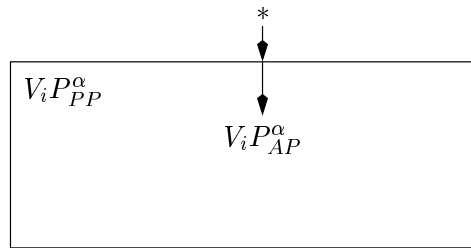


Es folgen dieselben sechs Produktionen noch einmal mit dem Index α' .

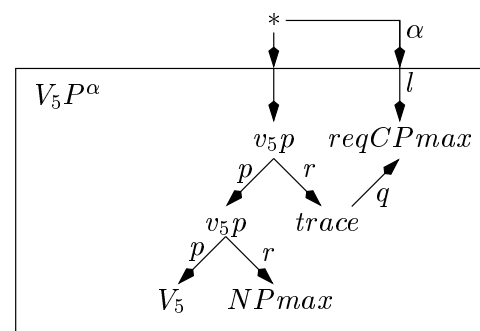
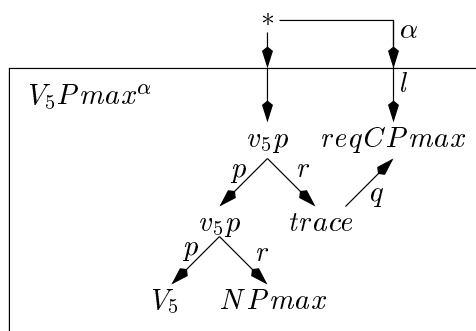
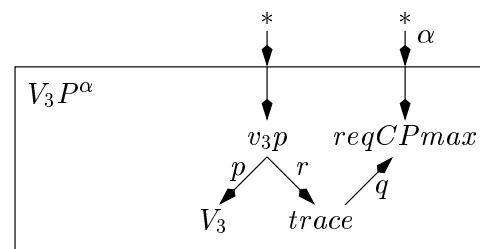
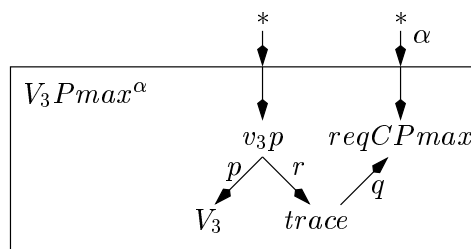
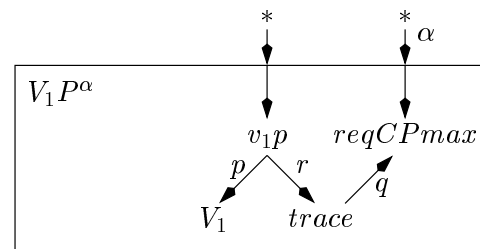
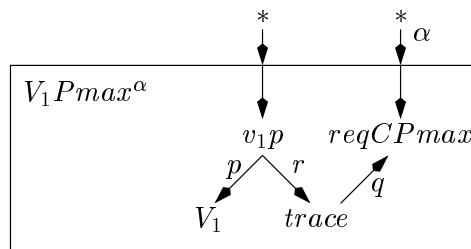


Wie auch bei den vorangehenden Grammatiken folgen wieder Umbenennungen, die bezüglich der Indizes α und α' informationserhaltend sein müssen:

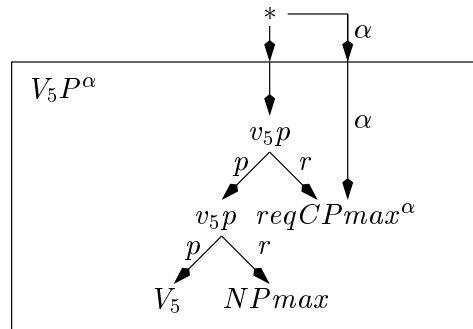
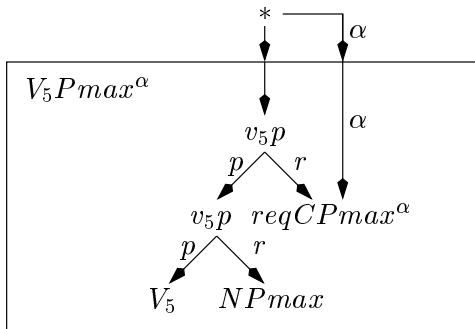
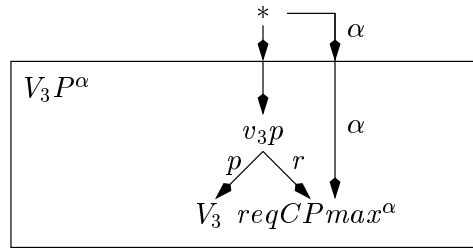
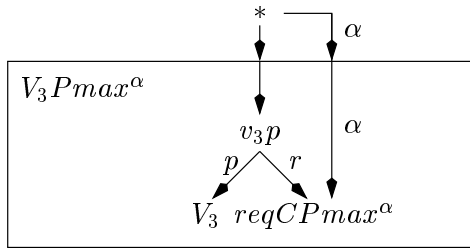
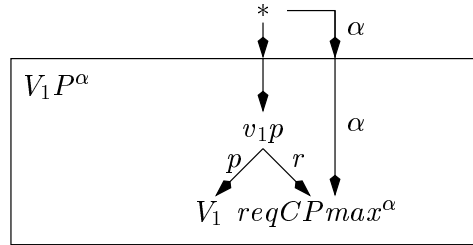
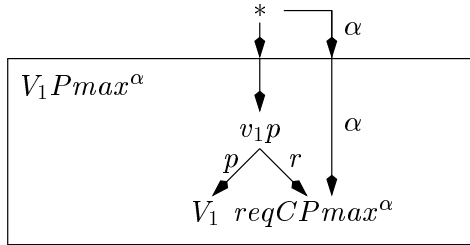




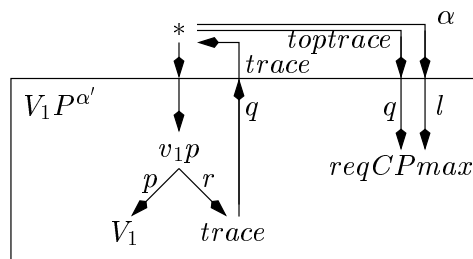
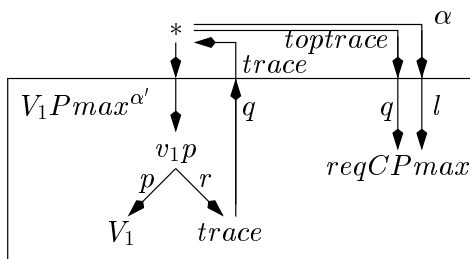
Abhängig vom Typ des Verbs wird unterschiedlich expandiert:

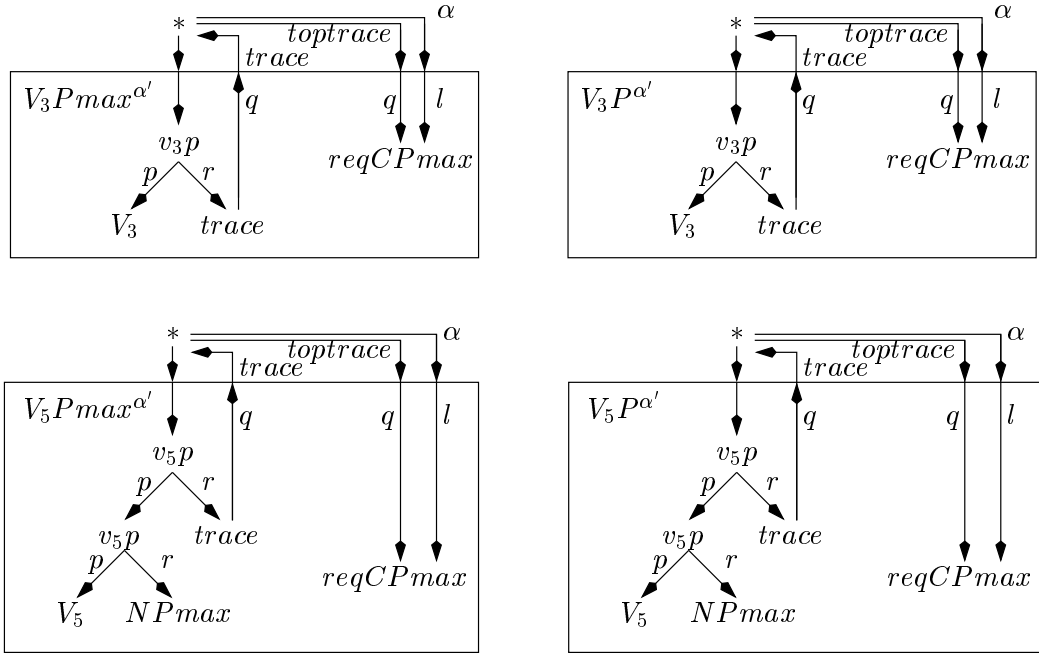


Falls nicht der erste Nebensatz topikalisiert wird, sondern erst ein späterer, dann muß mit jedem Grenzknoten eine weitere Spur initiiert werden, um der Subjazenbedingung zu genügen. Dies wird durch folgende Produktionen vorbereitet:



Schließlich folgt die terminale Einsetzung, welche die mitgeführten Kanten auf eine Weise reorganisiert, daß sie im resultierenden Graphen eine GB-kompatible Kette darstellen. Alle nichtterminalen Kanten erhalten das Terminalsymbol q und werden damit zu einer Querkante:





Die vorangehenden sechs Produktionen integrieren neue Knoten in den Graphen, die *nicht* unmittelbar miteinander zusammenhängen. Der mittelbare Zusammenhang im Syntaxgraphen ist allerdings dennoch gesichert, denn die Indexinformation α' garantiert, daß die Kanten mit den Bezeichnungen *trace*, *toptrace* und α existieren. Damit wird der Knoten an der richtigen Stelle im Syntaxgraphen – nämlich dem Landeplatz der Bewegungsoption – in den Syntaxgraphen integriert.

5.5.2.4 Eigenschaften der zusätzlichen Produktionen

Sei $GG_{\text{deriv-}\alpha}$ eine Graphgrammatik, die auf der Basis von GG_{deriv} und den in Abschnitt 5.5.2 vorgestellten zusätzlichen Produktionen P_{α} besteht. Auch $L(GG_{\text{deriv-}\alpha})$ besteht wie $L(GG_{\text{deriv-}\beta})$ nur aus GB-kompatiblen Graphen. Die Argumente zum Nachweis dieser Eigenschaften sind dieselben. Auf Grund der Boundary-Eigenschaft von $GG_{\text{deriv-}\alpha}$ liegt auch hier Konfluenz vor.

5.5.3 Erweiterung 3: α -Ketten und β -Ketten gleichzeitig

Die vorangehende Darstellung hat *entweder* α -Ketten *oder* β -Ketten umgesetzt. Für die Konstruktion einer Graphgrammatik, die Ketten beider Art auf einmal generiert, reicht es nicht aus, die Mengen P_{α} und P_{β} zu GG_{deriv} hinzuzufügen. Wie in Abbildung 5.13 illustriert, müssen dann zusätzliche Produktionen P_{γ} formuliert werden, die den gemischten Ableitungsprozeß mit Indizes α und β modellieren.

Die als Ergebnis aus dieser Erweiterung entstehende Graphgrammatik bezeichnen wir mit $GG_{\text{deriv-}\alpha-\beta-\gamma}$. Die Menge der Produktionen wächst dann stark an, weil mit zunehmender Anzahl an Indizes immer mehr Zustandskombinationen unterschieden werden müssen.

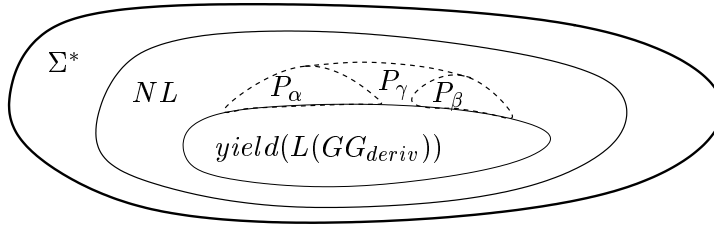
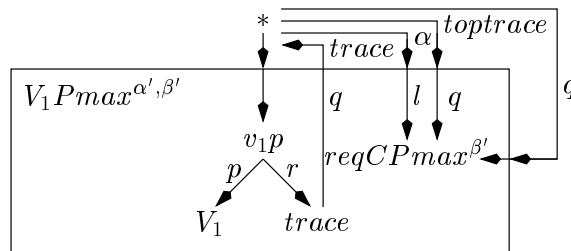
Abbildung 5.13: Erweiterungsmöglichkeiten von GG_{deriv}

Abbildung 5.14 zeigt exemplarisch ein Beispiel für eine solche erweiterte Produktion. Es wird nicht nur eine subkategorisierte CP an den Satzanfang bewegt (unter Berücksichtigung der Subjazenrestriktionen), sondern darüber hinaus eine Kante mit Bindungsinformation weitergegeben.

Abbildung 5.14: Erweiterte Produktion können α - und β -Ketten gemeinsam generieren

Produktionen dieser Art sind nötig, um beispielsweise Sätze folgender Art modellieren zu können: „ $(That\ Mary\ loves\ him_\alpha)_i$, $John_\alpha\ believes\ trace_i$ “. An den grundsätzlichen Eigenschaften der Graphgrammatik ändert sich durch diese Erweiterungen nichts. Sie bleibt boundary und gradbeschränkt. Alle generierten Graphen sind zusammenhängend.

Die Integration zusätzlicher Prinzipien und Phänomene in das Modell würde mehr Ketentypen erfordern, als wir sie in dieser Arbeit vorgestellt haben. Mit zusätzlichen Produktionen lassen sich immer bessere Approximation für NL entwerfen.

Jedoch sollte die Auswahl der für die jeweilige Anwendung relevanten syntaktischen Phänomene wohlüberlegt sein. Schließlich ist bei einer weiteren Verfeinerung von GG_{deriv} ein Wachstum der Anzahl der Produktionen in der Größenordnung der Potenzmenge über die Menge der zur Anwendung kommenden Indizes zu erwarten. Das hat für praktische Anwendungen dahingehend Folgen, daß auch bei theoretisch vorhandenen Polynomialzeitparsern die Laufzeit durch exponentiell viele Produktionen drastisch ausgebremst wird.

5.5.4 Erweiterung 4: Kontextsensitive Phänomene

Der nachfolgend diskutierte Aspekt bricht etwas mit dem vorangehenden Gedanken-gang. Wie bereits in Abschnitt 3.3.1 dargestellt, reichen kontextfreie Mechanismen nicht aus, um alle relevanten Wortstellungsphänomene in natürlichen Sprachen zu modellieren. Nun ist die englische Sprache nur in umstrittenen Fällen von diesem Phänomen

betroffen²⁶ und die Graphgrammatik $GG_{deriv-\alpha-\beta-\gamma}$ produziert mit und ohne Ketten im Zusammenhang mit der *yield*-Funktion ausschließlich kontextfreie Strings:

SATZ 28: Sei $L = \{w | w = yield(sg); sg \in L(GG_{deriv-\alpha-\beta-\gamma})\}$ die Menge der von Syntaxgraphen beschriebenen Terminalstrings. Dann ist L eine kontextfreie Sprache.

BEWEIS: Durch Konstruktion: Die Produktionen von $L(GG_{deriv-\alpha-\beta-\gamma})$ lassen sich in zwei Gruppen einteilen: (i) Die rechte Seite der Produktion ist zusammenhängend und (ii) die rechte Seite der Produktion ist nicht zusammenhängend.

Für die Produktionen von (i) gilt, daß sie bezüglich der Kanten l , p und r eine Baumstruktur mit eventuell überlagerten Querkanten q darstellen. Nichtterminale Knoten kommen nur an den Blättern vor. Bezüglich der Ordnung $l < p < r$ sind die Kanten und damit auch die Blattknoten geordnet. Unter Vernachlässigung der Querkanten (die für die Wortstellung keine Relevanz haben) lassen sich Produktionen vom Typ (i) auf kanonische Weise in Produktionen einer kontextfreien Grammatik übersetzen, die denselben *yield* besitzen.

Produktionen vom Typ (ii) kommen nur im Zusammenhang mit α -Ketten vor. Diese bestehen aus einer Struktur vom Typ (i) (die sich leicht in eine kontextfreie Produktion übersetzen lassen) und zusätzlich einem isolierten Knoten v . Dieser isolierte Knoten wird nun *immer* links unterhalb des Wurzelknotens eingefügt. Dieser Vorgang kann dadurch ersetzt werden, daß v durch die Produktion einer kontextfreien Regel $S \rightarrow v^{\beta'} Rest^{\beta'}$ an den Anfang plaziert wird. Der Index β' stellt zusammen mit weiteren Produktionen sicher, daß in beiden Zweigen der Ableitung die für eine GB-kompatible Struktur nötigen Expansionen vorgenommen werden.

Die vorangehende Konstruktion ignoriert den Verlauf von Querkanten, was zulässig ist, denn diese haben auf die links-rechts-Ordnung der Knoten keinen Einfluß. □

Was die Terminalstrings betrifft, erweist sich der Zugang über Graphgrammatiken also bis an diese Stelle als eine aufwendige Methode, um kontextfreie Sprachen zu beschreiben.²⁷ Wie bereits dargestellt, lassen sich in manchen Sprachen aber auch kontextsensitive Phänomene bei der Wortstellung beobachten. Nachfolgend wird demonstriert, wie sich auch diese mit (boundary) Graphgrammatiken beschreiben lassen. Um dabei die Zahl der Produktionen übersichtlich zu halten, lösen wir uns von der mit der Graphgrammatik GG_{deriv} gewählten Darstellung und beschränken uns darauf, die grundlegenden Mechanismen zu beschreiben.

Gesucht ist ein Mechanismus, der in der Lage ist, die folgenden Daten zu beschreiben:

Englisch: $N_1 V_1 N_2 V_2 N_3 V_3 \dots$

„*Tim said, that we₁ let₁ the children₁ help₁ John₁ paint₁ the house.*“

²⁶ Siehe dazu Higginbotham in [Hig87] („*such that*“-Konstruktionen) sowie Bar-Hillel in [BH64] („*respectively*“) und Gegenpositionen von Pullum und Gazdar in [PG87]

²⁷ Die Charakterisierung „aufwendig“ bezieht sich auf die Beschreibung von Terminalstrings. Darüber hinaus besitzen die Graphgrammatiken für die Beschreibung syntaktischer Strukturen eine eigene Daseinsberechtigung.

Deutsch: $N_1 N_2 N_3 \dots V_3 V_2 V_1$

„Tim sagt, daß wir₁ die Kinder₂ Hans₃ das Haus anstreichen₃ helfen₂ lassen₁.“

Schweizerdeutsch: $N_1 N_2 N_3 \dots V_1 V_2 V_3 \dots$

„Tim säit, das mer₁ d'chind₂ em Hans₃ es Huus lönd₁ hälfe₂ aastriche₃.“

Die englischen Daten lassen sich als sequentielle Abfolge problemlos beschreiben mit einer Produktion wie in Abbildung 5.15.

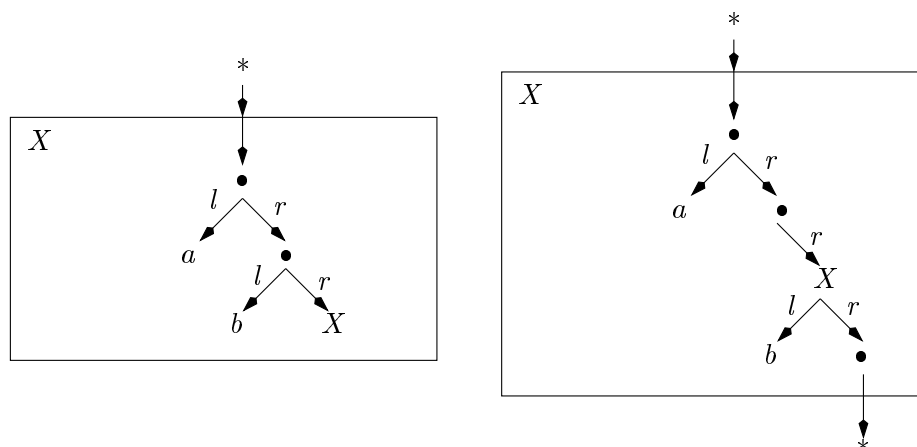


Abbildung 5.15: Produktionen für englische (links) und deutsche (rechts) Wortstellung

Die deutschen Daten lassen sich bei Wortgrammatiken mit einer kontextfreien Einbettungsregel modellieren, die sich auch mit einer Graphgrammatik-Produktion nachbauen läßt. Die Produktion in Abbildung 5.15 (rechts) beschreibt einen *yield*, dessen Ableitungsfolge wie ein korrekt geklammerter Term verschachtelt ist.

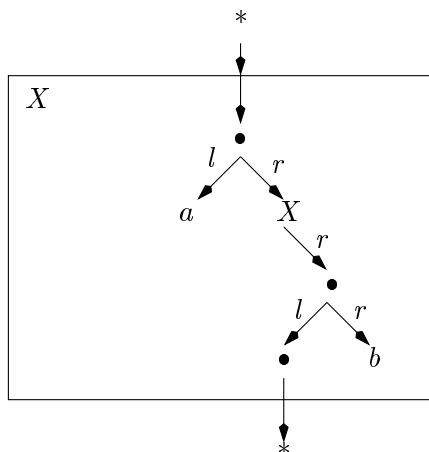


Abbildung 5.16: Produktion, welche die schweizerdeutsche Wortstellung beschreibt

Von der Schweizerdeutschen Struktur ist bekannt, daß sie nicht kontextfrei ist. Abbildung 5.16 zeigt, daß mit einer kleinen Modifikation der deutschen Produktionsregel ein Syntaxgraph beschrieben werden kann, welcher als *yield* die gewünschte Wortstel-

lung repräsentiert. Die Produktion beschreibt einen *yield*, dessen Blätter im Sinne der COPY-Language *ww* verschränkt sind. Die Anwendung dreier aufeinanderfolgender Ableitungsschritte demonstriert Abbildung 5.17.

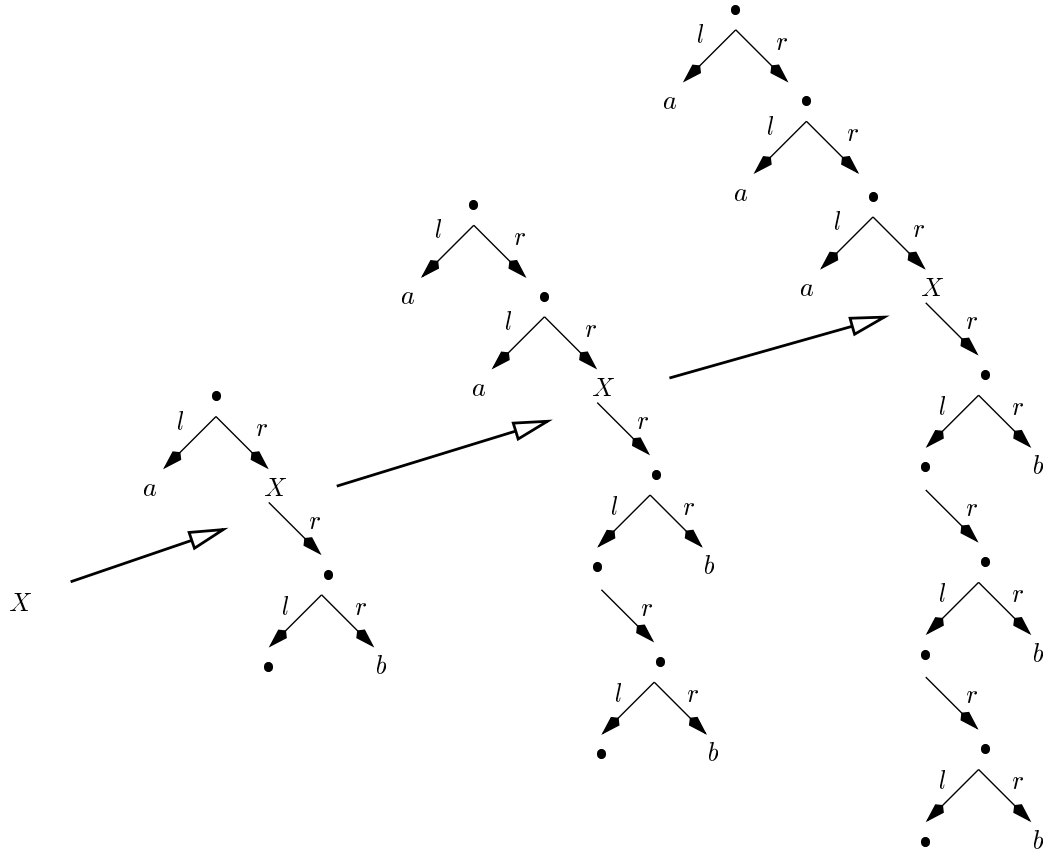


Abbildung 5.17: In drei Ableitungsschritte zum *yield* $a_1a_2a_3 \dots b_1b_2b_3$

Zwar werden bei jedem Ableitungsschritt neue Symbole lokal in der Nähe vom Nichtterminalsymbol X generiert. Auf den *yield* bezogen wird dieser allerdings an zwei Stellen modifiziert, die weit auseinander liegen können. Diesen Mechanismus werden wir in Kapitel 6 ausnutzen, um eine entsprechende Wortgrammatik zu entwerfen.

5.6 Analyse von GG_{deriv*} bezüglich des Wortproblems

DEFINITION 63 (GG_{deriv*}): Eine Graphgrammatik auf der Basis der in den Abschnitten 5.5.3 und 5.5.4 vorgestellten Mechanismen heißt nachfolgend GG_{deriv*} .

Eine Graphgrammatik GG_{deriv*} erzeugt GB-kompatible Graphen und kann Wortstellungsphänomene wie im Schweizerdeutschen beschreiben. Wir haben gezeigt, daß dafür kein besonders aufwendiger Grammatiktyp verwendet werden muß. Auch boundary Graphgrammatiken erweisen sich als mächtig genug.

Letzteres erweist sich in Zusammenhang mit weiteren Eigenschaften von GG_{deriv*} als nützlich bei der Betrachtung des Wortproblems. Wie Rozenberg und Welzl zeigen, lassen sich boundary Graphgrammatiken dann in Polynomialzeit parsen, wenn alle Graphen in der Sprache zusammenhängend sind und der maximale Grad all dieser Graphen durch eine Konstante beschränkt werden kann.²⁸ Alle diese Voraussetzungen sind für GG_{deriv*} erfüllt. Die Autoren zeigen aber auch, daß die Exponenten des Polynoms groß werden können: Sie schätzen die Komplexität mit $O((|V| + |E|)^{2*|\Delta|*|\Sigma|*d+2}) \text{ ab}^{29}$.

5.7 Zusammenfassung

Auf dem Hintergrund der vorangehenden Analyse erweisen sich konfluente Graphgrammatiken als geeignetes Mittel zur Beschreibung syntaktischer Strukturen einer Menge NL . Sie ermöglichen eine *derivationelle* und zugleich *effiziente* Modellierung. Die Verwendung von C-edNCE Graphgrammatiken in dieser Arbeit ist dabei exemplarisch zu verstehen. Wie Brandenburg³⁰ zeigt, werden Graphsprachen im Falle beschränkten Grades von vielen Graphgrammatikformalismen erkannt, wenn dies auch für C-edNCE der Fall ist. Entsprechend lassen sich die Ergebnisse zum Beispiel auch auf Hyperkantenersetzungssysteme übertragen.

Die formalen Grenzen für konfluente Graphgrammatiken folgen aus dem Separatorthorem. Interpretieren läßt sich dieses Ergebnis dahingehend, daß es mit unterschiedlichsten Herangehensweisen nicht gelingen kann, die Struktur Syntaxgraph rekursiv in kleine handhabbare Einheiten zu zerlegen, die aus einer großen Problem Instanz mehrere kleine Instanzen machen. Zwar ist die linguistische Forschung bestrebt, als ein universelles Prinzip das der *Lokalität* auszuzeichnen, doch scheint es hier Grenzen des Machbaren zu geben. Schwierigkeiten zeichnen sich genau da ab, wo dieses Lokalisprinzip nicht herrscht, wie zum Beispiel bei koreferenten Beziehungen, die in dieser Arbeit als Kette eingeführt wurden. Phänomene dieser Art machen den ansonsten „dünnen“ Graphen „breit“.

Was bedeutet dies für praktische Anwendungen? Die Beispiele in dieser Arbeit machen deutlich, daß die problematischen Fälle sich fast immer auf „exotische“ Satzkonstruktionen beziehen. Wir deuten dies als Hinweis darauf, daß GB hinsichtlich der zu beschreibenden Menge NL zu mächtig ist. Die in den vorgestellten Satzbeispielen verwendeten Konstruktionen unterstützen diese Vermutung.

Die Separatoreigenschaft weist einen Weg, bessere von schlechteren Satzkonstruktionen zu unterscheiden: So wie Fanselow und Felix anführen, daß Sätze um so schlechter werden, je mehr Barrieren beim Bewegungsprozeß überschritten werden³¹, so stellen wir die ergänzende These auf, daß Sätze mit wachsendem Separator schlechter werden. Das Argument von Fanselow/Felix beschreibt *die Größe einer Menge* zusammenhängender Querbeziehungen, also einer Kette. Der Separatorbegriff beschreibt *die Anzahl mehrerer*

²⁸ [RW86]

²⁹ Dabei repräsentiert Δ das Kantenalphabet, Σ das Knotenalphabet und d den maximalen Grad eines Graphen $G = (V, E)$.

³⁰ [Bra91]

³¹ [FF87b, S.248]

solcher Kette, die gleichsam parallel im Syntaxgraphen verlaufen. Die beiden Zugänge schließen sich also nicht aus, sondern sie ergänzen sich.

In diesem Sinne ist es möglich, für eine Sprache NL bei einem gegebenen festen Separator eine beliebig gute konfluente Graphgrammatik zu entwerfen. Ein solcher fester Separator beschränkt dann *nicht* die Anzahl der Ketten im Satz. Im Gegenteil zeigen die Beispiele, daß auch konfluente Graphgrammatiken in der Lage sind, unendlich viele Ketten zu modellieren. Die Beschränkung hat vielmehr Einfluß darauf, wieviele Ketten parallel verlaufen. Aus der Sicht von Anwendungen stellt dies keine starke Einschränkung dar.

Wir haben gezeigt, daß eine boundary Graphgrammatik ausreicht, um syntaktische Strukturen von natürlichen Sprachen zu beschreiben. Dies hat dahingehend positive Konsequenzen, als daß für Graphgrammatiken dieser Klasse das Wortproblem in Polynomialzeit lösbar ist. Zwar kann die Konstante des Polynoms dabei sehr groß werden, jedoch fällt dies nur beschränkt ins Gewicht, da sich die relevanten Instanzen typischerweise durch ein kleines n charakterisieren lassen. Aufmerksamkeit erfordert die Ausformulierung der Grammatik: Wie das Beispiel GG_{deriv} deutlich gemacht hat, kann eine Grammatik abhängig von der Zahl modellierter Phänomene sehr umfangreich werden, was meßbare Auswirkungen auf die Laufzeit von Parsingalgorithmen hat, die in der O -Notation nicht zum Ausdruck kommen.

Davon unberührt bleibt, daß mit dem Wortproblem für Syntaxgraphen nicht auch automatisch das Wortproblem für Zeichenketten gelöst ist. Das leitet über zur Fragestellung des letzten Kapitels. Dort wird gezeigt, wie es mit einer erweiterten kontextfreien Wortgrammatik möglich ist, die (String)-Sprache zu beschreiben, die in diesem Kapitel mit Hilfe von Produktionen einer Graphgrammatik charakterisiert worden ist.

in everyday mathematics formalisation in the strict sense is absent: only precision and clarity are required. Much work in linguistics lacks both, and it is a noble task to look for remedies.

MARCUS KRACHT IN [KRA97]

6 Von Graphgrammatiken zu Wortgrammatiken

6.1 Ziel: Entwurf einer effizienten Wortgrammatik

In den vorangehenden Kapiteln wurde gezeigt, wie sich syntaktische Strukturen natürlicher Sprachen mit Hilfe von Graphgrammatiken beschreiben lassen. Diese Kapitel macht sequentielle Zeichenketten zum Gegenstand der Betrachtung und entwickelt für diese eine effiziente Beschreibungsform.

Die bis jetzt vorgestellten Techniken beschreiben lineare Wortfolgen natürlicher Sprachen mit einem Zweitupel $(GG_{deriv*}, yield)$. Abbildung 6.1 illustriert, wie dieser Vorgang in zwei Phasen abläuft. Gesucht ist nun ein Verfahren, welches in der Tradition herkömmlicher Wortgrammatiken Zeichenketten in einer einzigen Phase generiert. Daß sich mit Hilfe von Graphgrammatiken Zeichenketten auch direkt beschreiben lassen, ist keine neue Idee.¹ Wir nutzen diese Methode aus, um die kontextsensitiven Aspekte von \mathcal{NL} besser zu verstehen. Dies hilft dabei, die relevanten Mechanismen besser zu durchschauen und eine geeignete Wortgrammatik auszuarbeiten. Dabei gehen wir in diesem Kapitel in drei Schritten vor:

Zunächst definieren wir eine Graphgrammatik GG_{chain} , welche den Mechanismus der *yield*-Funktion gleich mit in den Ableitungsprozeß integriert. GG_{chain} generiert Ket tengraphen. Die terminalen Knoten stehen am Ende der Ableitung in der richtigen Reihenfolge. Außerdem drücken Querkanten die relevanten Querbeziehungen zwischen den Knoten aus.

In einem zweiten Schritt wird dann auf der Grundlage von GG_{chain} ein Wortgrammatik-formalismus (Right Adjunct Grammar) entwickelt, welche bezüglich der Reihenfolge der Terminalknoten dieselbe Funktionalität besitzt, die Querkanteninformation aber nicht ausdrückt. Zu diesem Zweck wird das Prinzip kontextfreier Grammatiken um einen geeigneten Mechanismus erweitert.

Schließlich werden die so entstandenen Right Adjunct Grammars auf ihre formalen Eigenschaften hin untersucht und ein Algorithmus zur Lösung des Wortproblems entwickelt.

¹ siehe zum Beispiel [Eng97, Kapitel 6]

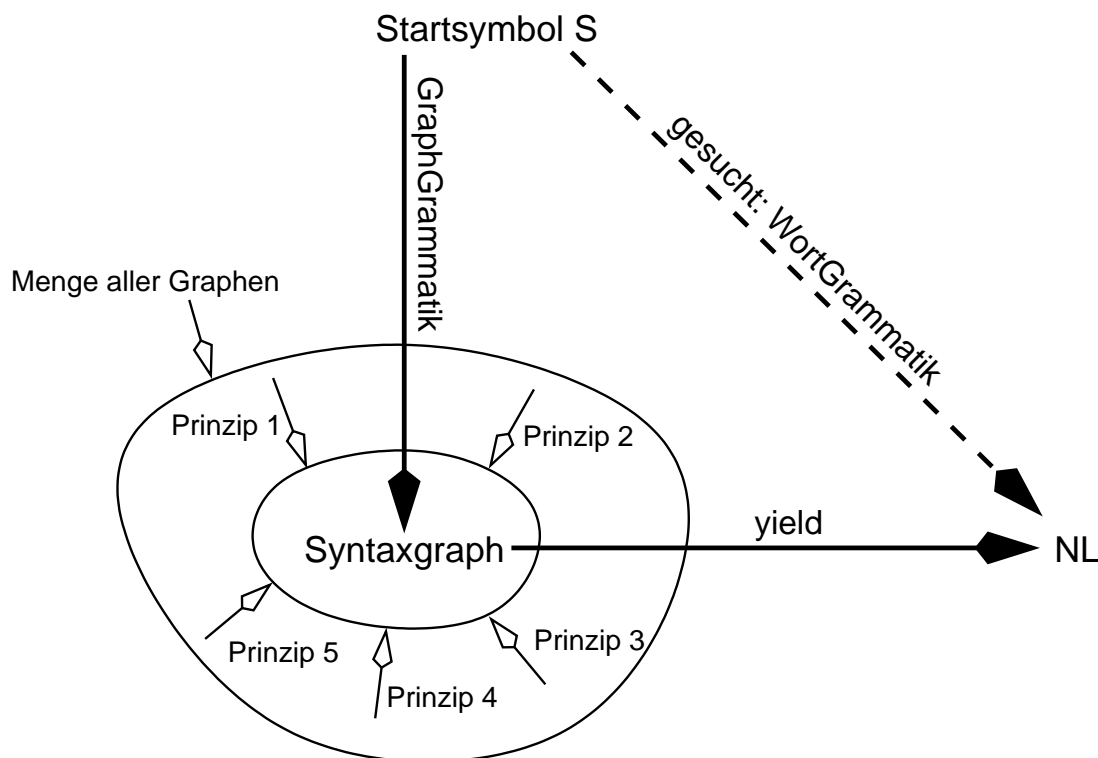


Abbildung 6.1: Eine Wortgrammatik fasst zwei Ableitungsphasen zusammen.

Um diese drei Ziele zu erreichen, fassen wir einige relevante Ergebnisse der letzten Kapitel zusammen: Satz 28 zeigt, daß die durch die Graphgrammatik $GG_{deriv-\alpha-\beta-\gamma}$ und die Auswertungsfunktion *yield* beschriebene Menge von Zeichenketten eine kontextfreie Sprache darstellen. Wie in Abschnitt 3.3.1 gezeigt, kommen in manchen natürlichen Sprachen auch kontextsensitive Mechanismen zur Anwendung. Die Produktion in Abbildung 5.16 demonstriert, wie auch diese sich mit Hilfe von Graphgrammatiken modellieren lassen.

Für die Modellierung der in dieser Arbeit betrachteten syntaktischen Phänomene reicht es aus, die Anwendbarkeit dieser Produktion nur an einer Stelle im Ableitungsbaum zuzulassen. An einer solchen Stelle ist die Anzahl der Produktionsanwendungen aber unbeschränkt. Das Ergebnis einer zwei- und dreifachen Mehrfachanwendung zeigt Abbildung 5.17. Offensichtlich sind die durch solche Mehrfachanwendungen beschriebenen Zeichenketten mit der COPY-Language *ww* verwandt, von der bekannt ist, daß sie nicht kontextfrei ist. In diesem Kapitel werden wir einen Formalismus für eine Wortgrammatik entwerfen, der diesen Mechanismus verallgemeinert und konkret umsetzt.

Dazu nutzen wir die Beobachtung aus, daß sich alle bis jetzt vorgestellten Produktionen nach dem Gesichtspunkt klassifizieren lassen, ob sie einen zusammenhängenden *yield* generieren, oder nicht. Entsprechend unterscheiden wir *adjungierende* und *nichtadjungierende* Produktionen. Diese Klassifizierung ermöglicht die systematische Generierung von Produktionen für Kettengraphen auf der Basis der bis jetzt vorgestellten Regeln. Eine entsprechende Bauanleitung entwickeln wir im nächsten Abschnitt.

6.2 Entwurf einer Graphgrammatik GG_{chain}

Ein Zweitupel $(GG_{deriv*}, yield)$ beschreibt zulässige *Sequenzen von Knoten* (also lineare Strukturen) in einem zweistufigen Prozeß. Um das Ergebnis von $(GG_{deriv*}, yield)$ durch eine einzige Graphgrammatik GG_{chain} zu beschreiben, muß die Funktionalität der Funktion $yield$ in die Graphgrammatik GG_{deriv*} hineincodiert werden. Wir zeigen nachfolgend systematisch, welche Modifikationen dazu nötig sind.

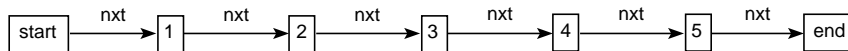


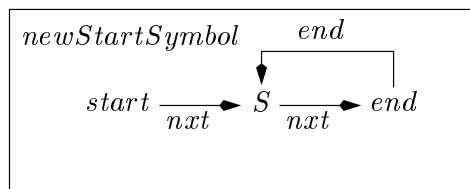
Abbildung 6.2: Gesucht: Graphgrammatiken für Kettengraphen

Angestrebt wird die Ableitung eines terminalen Kettengraphen, der dem $yield$ entspricht (Abbildung 6.2). In diesem Graphen sind die Knoten ab einem Knoten $start$ sequentiell durch Kanten mit der Markierung nxt bis hin zu einem Knoten end verbunden. Die inneren Knoten und Kanten des Syntaxgraphen werden in diesem Fall nicht erzeugt, die Information über die syntaktische Struktur geht – gewollt – verloren.

Eine *neue Startproduktion* stellt sicher, daß eine zugrundeliegende Kettenstruktur generiert wird, in welche die übrigen Expansionen eingebettet werden. Dafür führen wir ein neues Startsymbol ein, welches dem alten Startsymbol S vorgeschaltet ist:

Neues Startsymbol : $newStartSymbol$

Neue terminale Kantenmarkierungen : nxt, end

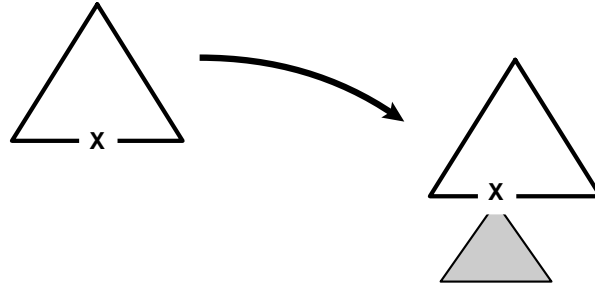


Wie bereits dargestellt, lassen sich die Produktionen von GG_{deriv*} in zwei Klassen einteilen: (i) solche, deren Einfluß auf den $yield$ durch eine kontextfreie Produktion beschrieben werden kann und (ii) solche, für die das nicht möglich ist.

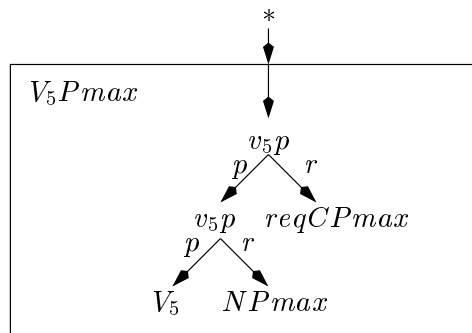
Für beide Klassen geben wir eine Anleitung zur Modifikation an:

1. nichtadjungierende Produktionen

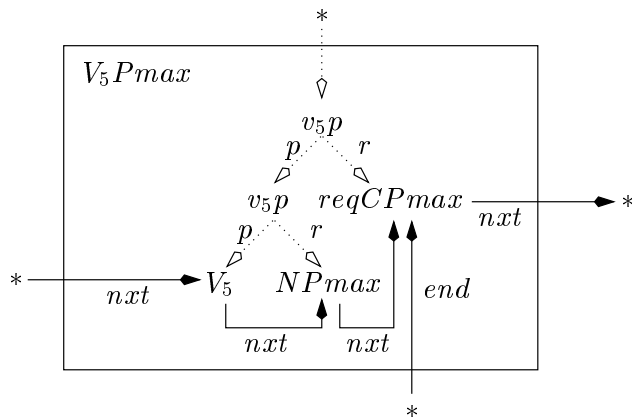
Wir nutzen aus, daß GG_{deriv*} die boundary-Eigenschaft besitzt. Die inneren terminalen Knoten auf den rechten Seiten der Produktionen spielen für den $yield$ keine Rolle, ein nichtterminaler Knoten wird ersetzt durch einen terminalen Baum mit nichtterminalen Blättern.



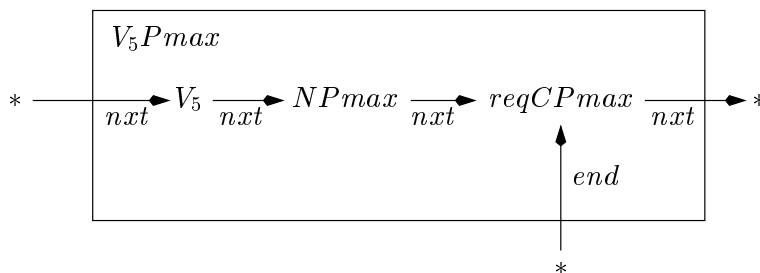
Die so charakterisierbaren Produktionen lassen sich systematisch in eine Produktion für GG_{chain} umwandeln. Dies sei am Beispiel einer umfangreichen Produktion von GG_{deriv*} demonstriert, die wir hier noch einmal wiederholen:



Durch das Einfügen zusätzlicher Kanten kann die durch die Kantenbeschriftung implizit gegebene Ordnungsrelation auf die Blätter explizit zum Ausdruck gebracht werden. Die alten Kanten sind in der Abbildung zur Verdeutlichung gepunktet angegeben:



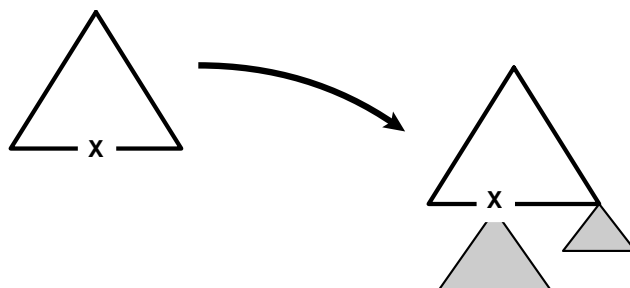
Wenn – wie in unserer Situation – in erster Linie die sequentielle Wortstellung interessiert, dann können die Kanten mit der Bezeichnung l , p , r ignoriert werden wie auch die inneren terminalen Knoten. Querkanten bleiben erhalten.



Der Verweis von dem Endknoten in den beiden vorherigen Produktionen darf an *höchstens einen* Nichtterminalknoten weitergegeben werden. Diese explizit ausgesprochene Restriktion ist für die Umsetzung adjungierender Produktionen relevant und beschränkt die Anzahl der möglichen Anwendungen kontextsensitiver Erscheinungen.

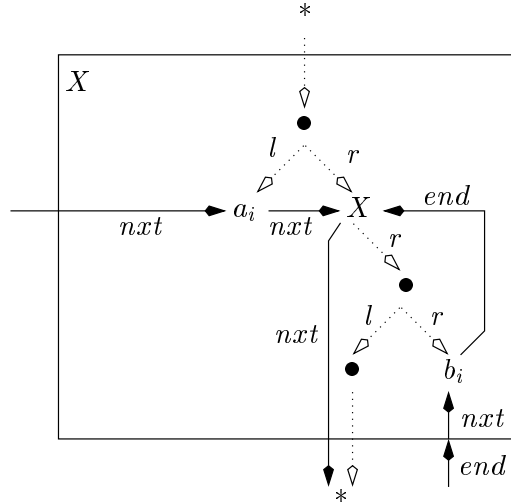
2. adjungierende Produktionen

In diesem Fall erweist sich die Modifikation als schwieriger, weil die entsprechenden Produktionen den *yield* nicht nur an einer Stelle expandieren. Vielmehr wird bei jeder Produktionsanwendung zusätzlich am Ende des *yields* ein Nichtterminalknoten generiert, aus dem gemäß den üblichen Regeln eine weitere Ableitungsstruktur hervorgehen kann.

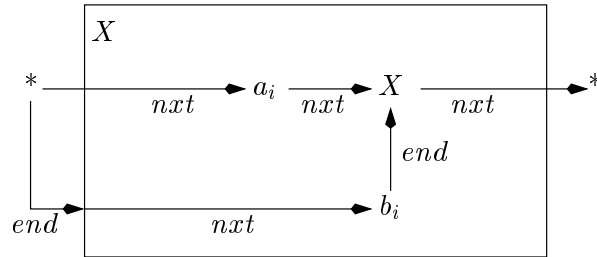


Um diesen Mechanismus als Produktion einer Graphgrammatik GG_{chain} für Kettengraphen zu modellieren, muß die Einbettungsrelation entsprechend formuliert werden. Wir demonstrieren diese Modifikation anhand der bereits in Abbildung 5.16 vorgestellten Produktion zur Generierung schweizerdeutscher Phänomene.

Da der *yield* durch diese Produktion nicht mehr zusammenhängend expandiert wird, sondern an zwei unzusammenhängenden Stellen, erweist sich die Umwandlung in eine „Kettenproduktion“ als nicht ganz so einfach. Die nächste Produktion demonstriert unter Beibehaltung der inneren terminalen Knoten die Nachbarschaftsrelation auf den Blattknoten. Aus der ursprünglichen Produktion übernommene Kanten sind wieder punktiert gekennzeichnet:



So modifiziert erweisen sich die inneren Knoten als irrelevant für die sequentielle Abfolge der Blattknoten. Damit erhalten wir eine Produktion folgender Gestalt:



Werden die beiden vorgestellten Modifikationen für alle Produktionen von GG_{deriv*} entsprechend Typ (i) oder Typ (ii) vorgenommen und ist dabei durch entsprechende Knotenmarkierungen sichergestellt, daß Produktionen von Typ (ii) zu keinem Zeitpunkt der Ableitung an zwei verschiedenen Stellen gleichzeitig angewendet werden können, so erhalten wir eine Graphgrammatik GG_{chain} , die Grundlage der weiteren Betrachtung ist.

DEFINITION 64 (GG_{chain}): Eine Graphgrammatik GG_{chain} für Kettengraphen entsteht durch die Vereinfachung einer Graphgrammatik für Syntaxgraphen GG_{deriv*} , indem nach vorab dargestellter Strategie jeder Produktion umgeformt wird. Für einen Umformungsschritt gibt es zwei Möglichkeiten abhängig davon, ob die Produktion adjungierend oder nichtadjungierend ist.

Durch geeignete Kantenmarkierungen kann wieder sichergestellt werden, daß adjungierende Produktionen nur an einer Stelle im Ableitungsbaum verwendet werden.

6.3 Analyse von GG_{chain}

LEMMA 29: Die Graphgrammatik GG_{chain} hat folgende Eigenschaften:

1. Alle Graphen aus $L(GG_{chain})$ sind zusammenhängend.
2. Alle Graphen aus $L(GG_{chain})$ sind von beschränktem Grad $\leq k$.
3. GG_{chain} ist konfluent.

BEWEIS:

1. Die Startproduktion generiert einen zusammenhängenden Kettengraphen. Der Zusammenhang kann nur verloren gehen, wenn eine Produktion vom Typ (ii) zur Anwendung kommt, ohne daß eine Kante mit der Bezeichnung *end* das Nichtterminal auf der linken Seite der Produktion zum Ziel hat. Daß dies nicht passiert, kann durch entsprechende Kantenmarkierungen sichergestellt werden.
2. Wir haben offen gelassen, ob Querkanten bei der Modifikation von GG_{deriv*} zu GG_{chain} erhalten bleiben, oder nicht. In beiden Fällen kann der Grad nicht beliebig groß werden, weil keine wiederholten Mehrfachanwendungen von Produktionen auf ein Nichtterminalsymbol möglich sind.
3. Die nichtadjungierenden Produktionen gefährden die Konfluenz nicht. Sie können in beliebiger Reihenfolge angewendet werden. Für adjungierende Produktionen gilt das nicht: Die Reihenfolge der Anwendung hat wegen der Kante mit der Markierung *end* Einfluß auf die Struktur der resultierenden Graphen. Da aber die Definition von GG_{chain} sicherstellt, daß adjungierende Produktionen nicht an verschiedenen Stellen im Ableitungsbaum angewendet werden können, kommt dieser Fall nicht vor. □

Aus den Eigenschaften ergibt sich, daß das Wortproblem für Kettengraphen aus GG_{chain} in Polynomialzeit lösbar ist.² Der Exponent wird wieder groß, was dadurch kompensiert wird, daß n typischerweise klein ist. Die Ergebnisse im folgenden Abschnitt zeigen, daß sich bei einem für den Zweck der Beschreibung von Zeichenketten optimiertem Grammatikformalismus das Wortproblem in der Zeit $O(n^4)$ lösen läßt.

Daß Erweiterungen von kontextfreien Grammatiken existieren, die sich in kubischer Zeit parsen lassen, ist nicht überraschend: So ist bekannt, daß der Schnitt zweier kontextfreier Sprachen nicht kontextfrei sein muß, dasselbe gilt für das Komplement einer kontextfreien Sprache. Dennoch lassen sich derartige Sprachen leicht durch minimale Modifikation eines CFG-Parsers in $O(n^3)$ erkennen.³ Die von uns vorgestellte Erweiterung hebt sich von anderen Vorschlägen dadurch ab, daß sie unmittelbar durch die Eigenschaften einer Graphgrammatik motiviert ist, welche die zugrundeliegenden syntaktische Strukturen beschreibt.

² Die entsprechenden Zusammenhänge werden auf S. 147 erörtert.

³ [Bou99]

Andere Wortgrammatiken für Erweiterungen kontextfreier Sprachen werden in Abschnitt 3.3.3 im Zusammenhang mit *mildly Contextsensitive Languages* ($mCSL$) vorgestellt. Die besten derzeit bekannten Parsingalgorithmen für $mCSL$ besitzen eine Komplexität von $O(n^6)$, was die Suche nach besseren Alternativen motiviert. Schließlich ist nicht klar, inwiefern auch die Klasse $mCSL$ nicht noch zu mächtig ist für natürlichsprachige Phänomene. Der nachfolgend vorgestellte Grammatiktyp besitzt bessere Parsingeigenschaften und erfüllt trotzdem die Anforderungen, die sich aus unserer Untersuchung ergeben.

6.4 Right Adjoining Grammars

Wenn eine adjungierende Produktion an einer zulässigen Stelle X in einem Wort $w_1 X w_2$ angewendet wird, dann verändert sich die Satzform an zwei Stellen A und B :

$$w_1 X w_2 \Rightarrow w_1 A w_2 B$$

Der Vorgang läßt sich als (i) *lokale kontextfreie Ersetzung* und zusätzliche (ii) *Adjunktion an den rechten Rand der Zeichenkette* in einem unteilbaren Schritt verstehen. Dieser Mechanismus wird nachfolgend formalisiert:

DEFINITION 65 (RIGHT ADJOINING GRAMMAR (RAG)): Eine *Right Adjoining Grammar* RAG ist ein Tupel $G = (N, T, P, (S, +))$ ($N \cap T = \emptyset$):

1. N : endliche Menge von Nichtterminalsymbolen, die markiert oder unmarkiert sein können. Dies wird in Form von Paaren notiert, zum Beispiel $(A, -)$ (A ist unmarkiert) oder $(A, +)$ (A ist markiert). Es gilt $(S, +) \in N$
2. T : endliche Menge von Terminalsymbolen. Diese besitzen keine Möglichkeit zur Markierung.
3. P : endliche Menge von erweiterten Produktionen (siehe unten)
4. $(S, +)$: Ein markiertes Startsymbol.⁴

Eine erweiterte Produktion p genügt der Form: $A \rightarrow [B_1 B_2 \dots B_n] C_1 C_2 \dots C_m$ mit $A \in N$ und $B_i, C_i \in N \cup T$. Auf der rechten Seite stehen zwei Wörter aus Terminal- und Nichtterminalsymbolen. Das Wort in eckigen Klammern heißt *Kern der Expansion*, das Wort rechts davon heißt *Adjunkt*. Eine Produktion mit leerem Adjunkt heißt *nichtadjungierend*. Dort können die eckigen Klammern zur kompakten Schreibweise weggelassen werden. Die Produktion sieht dann aus wie bei einer kontextfreien Grammatik.

Die linke Seite einer Produktion kann grundsätzlich markiert oder unmarkiert sein: (i) Wenn die linke Seite unmarkiert ist, darf auf der rechten Seite kein Symbol markiert sein und das Adjunkt ist leer. (ii) Wenn die linke Seite markiert ist, darf höchstens ein Symbol im Kern der rechten Seite markiert sein. Wenn es sich dabei um eine nichtadjungierende Produktion handelt, muß genau ein Symbol markiert sein.⁵

⁴ Grundsätzlich lassen wir auch die Form $(S, -)$ für das Startsymbol zu. Dann fällt dieser Grammatikformalismus aber von der Mächtigkeit her mit dem einer kontextfreien Grammatik zusammen.

⁵ Diese Bedingung verbietet terminale Produktionen der Form $(A, +) \rightarrow a$, denn terminale Symbole können nicht markiert werden.

Eine *Right Adjoining Grammar* stellt eine Erweiterung des Konzepts kontextfreier Grammatiken dar, mit einem erweiterten Produktionsbegriff:

DEFINITION 66 (PRODUKTIONSANWENDUNG): Sei $p \in P$ eine erweiterte Produktion mit linker Seite (X, α) und s eine Satzform, welche mindestens einmal das Symbol (X, α) enthält. Für ein solches (X, α) und p können zwei Fälle eintreten:

1. **p ist nichtadjungierende Produktion:** Die Produktion wird angewendet wie bei einer kontextfreien Grammatik. Ersetze das Symbol (X, α) in s durch den Kern von p und erhalte s' . Da das Adjunkt leer ist (weil nichtadjungierende Produktion), sind wir fertig.
2. **p ist adjungierende Produktion:** (i) Ersetze das (nach Definition notwendig markierte) Symbol $(X, +)$ in s durch den Kern von p und erhalte s' (ii) Konkateniere das Adjunkt von p rechts an s' und erhalte s'' .

BEISPIEL 25: Die Produktion $p = ((A, +) \rightarrow [(B_1, -)(B_2, -)](C_1, -)(C_2, -))$ einer RAG auf eine Satzform $s = (P, -)(Q, -)(A, +)(R, -)$ angewendet ergibt als Ergebnis $s'' = (P, -)(Q, -)(B_1, -)(B_2, -)(R, -)(C_1, -)(C_2, -)$. Wir schreiben dafür auch alternativ $(P, -)(Q, -)(A, +)(R, -) \Rightarrow^p (P, -)(Q, -)(B_1, -)(B_2, -)(R, -)(C_1, -)(C_2, -)$ oder kurz $s \Rightarrow^p s''$. Die Markierung geht in diesem Fall verloren, weil kein Symbol auf der rechten Seite der Produktion markiert ist.

Der *Ableitungsbaum* zu einem Wort einer RAG stellt die Reihenfolge der Produktionsanwendungen hierarchisch dar. Anders als bei dem Ableitungsbaum einer CFG können bedingt durch die Adjunktionsoperation Zweige des Baumes „über Kreuz“ laufen. Ein Beispiel zeigt Abbildung 6.8. Die Adjunktionsoperationen kommen durch Kanten mit der Beschriftung *adj* zum Ausdruck. Je später eine Adjunktionsoperation zur Anwendung kommt, um so weiter rechts steht das Resultat im terminalen String. Die Zeichnung zeigt weiter, daß die Adjunktionen entlang eines Pfades im Baum zur Anwendung kommen und nicht etwa parallel in verschiedenen Teilbäumen.

Mit Induktion zeigt man leicht, daß über den Ableitungsprozeß hinweg in jeder Satzform höchstens ein Nichtterminalsymbol markiert ist. Diese Markierung wird im Ableitungsbaum entlang eines Pfades weitergegeben. Durch die Definition von Produktionen ist implizit sichergestellt, daß eine Markierung nur im Rahmen einer adjungierenden Produktion verloren gehen kann. In diesem Fall kann eine Markierung nicht mehr nachträglich erzeugt werden.

DEFINITION 67 (SPRACHE EINER RAG, \mathcal{RAL}): Sei G eine RAG. Als Sprache $L(G)$ bezeichnen wir die Menge aller terminalen Wörter, die durch endlich viele Anwendungen von erweiterten Produktionen aus dem Startsymbol abgeleitet werden können. Als $\mathcal{RAL} = \{L \mid \exists G, G \text{ RAG} : L = L(G)\}$ bezeichnen wir die Menge aller Sprachen, die mit Hilfe einer RAG erzeugt werden können.

Wie bei einer kontextfreien Grammatik können bei einer *RAG* Kettenproduktionen⁶ und Produktionen mit nutzlosen Symbolen⁷ erkannt und eliminiert werden.

DEFINITION 68 (CHOMSKY-NORMALFORM EINER *RAG*): Eine Produktion einer *RAG* ist in Chomsky-Normalform (CNF), wenn sie einer der folgenden Formen genügt:

1. Expansion eines Nichtterminalsymbols zu zwei Nichtterminalsymbolen als nichtadjungierende Produktion gemäß der Vorschrift:
 $(X, \alpha) \rightarrow [(A, \beta)(B, \gamma)]$ (im Kern muß ein Symbol markiert sein, wenn $\alpha = +$)
2. Expansion eines markierten Nichtterminalsymbols zu zwei Nichtterminalsymbolen als adjungierende Produktion gemäß der Vorschrift:
 $(X, +) \rightarrow [(A, \alpha)](B, -)$ (mit $\alpha \in \{-, +\}$)
3. Expansion eines unmarkierten Nichtterminalsymbols zu einem Terminalsymbol:
 $(X, -) \rightarrow a.$

Eine *RAG* ist in CNF, wenn jede Produktion in CNF ist.

Man sieht leicht, daß wie bei kontextfreien Grammatiken für jede *RAG* eine äquivalente Grammatik in Chomsky Normalform existiert: Für die systematische Erzeugung werden zunächst nutzlose Symbole entfernt und durch Umformung wird weiter sichergestellt, daß weder ϵ - noch Kettenproduktionen in der Grammatik enthalten sind. Dann können lange rechte Seiten mit mehr als zwei Symbolen, durch Einführung zusätzlicher Nichtterminalzeichen aufgebrochen werden. Ein terminales Symbol wird - wenn die entsprechenden Produktion nicht bereits in CNF ist - durch ein neues nichtterminales Symbol T ersetzt und der Grammatik eine Produktion der Form $(T, -) \rightarrow t$ hinzugefügt.

6.5 Charakterisierungen der Klasse \mathcal{RAL}

LEMMA 30: $\mathcal{CFL} \subset \mathcal{RAL}$

BEWEIS: Zunächst zeigen wir die echte Teilmengenbeziehung durch Konstruktion: Jede *CFG* ist auch eine *RAG* mit Produktionen ohne Adjunkt, bei der kein Symbol markiert ist. Dazu wählen wir die rechte Seite einer Produktion einer *CFG* als Kern der Produktion einer *RAG*. Es ist klar, daß die so konstruierte Grammatik dieselbe Sprache erkennt.

Um zu zeigen, daß echte Ungleichheit vorliegt, betrachten wir die folgende *RAG* $G = (\{(A, -), (B, -), (C, -), (S, +)\}, \{a, b, c\}, P, (S, +))$ mit der dazugehörigen Produktionsmenge $P = \{((S, +) \rightarrow [(A, -)(S, +)(B, -)](C, -)[(A, -)(B, -)](C, -)), ((A, -) \rightarrow a), ((B, -) \rightarrow b), ((C, -) \rightarrow c)\}$. Diese Grammatik erzeugt die Sprache $\{a^n b^n c^n | n \geq 1\}$, von der bekannt ist, daß sie nicht kontextfrei ist. □

⁶ Darunter verstehen wir umbenennende Produktionen der Form $(A, \alpha) \rightarrow (B, \beta)$ bei denen die rechte Seite nur aus einem Nichtterminalsymbol besteht.

⁷ Ein Symbol (X, α) heißt nützlich, wenn es eine Ableitung $(S, +) \Rightarrow^* w_1(X, \alpha)w_2 \Rightarrow^* w$, $w \in T^*$ gibt. Andernfalls ist (X, α) nutzlos.

Ähnlich zeigt man, daß $\{a^n b^m c^n d^m | m, n \geq 1\} \in \mathcal{RAL}$ und $\{ww | w \in \{a, b\}^*\} \in \mathcal{RAL}$.

Die Beziehung $\mathcal{RAL} \subseteq m\mathcal{CSL}$ zeigt man durch Simulation einer RAG mit einer Linearen Index Grammatik⁸ (LIG): Jede RAG $G_1 = (N_1, T_1, P_1, (S_1, +))$ kann in eine LIG $G_2 = (N_2, T_2, I, P_2, S_2[.])$ umgeformt werden mit $L(G_1) = L(G_2)$. Dafür gehen wir davon aus, daß G_1 in CNF vorliegt. Für jeden möglichen Produktionstyp aus P_1 einer RAG geben wir in Abbildung 6.3 eine Übersetzungsregel nach P_2 an.⁹

Typ der Produktion	$RAG: G_1$	$LIG: G_2$
nichtadjungierend: unmarkiert: normal expandieren links markiert: Z später auswerten rechts markiert: normal expandieren	$(X, -) \rightarrow [(Y, -)(Z, -)]$ $(X, +) \rightarrow [(Y, +)(Z, -)]$ $(X, +) \rightarrow [(Y, -)(Z, +)]$	$X \rightarrow YZ$ $X[.] \rightarrow Y[.Z_R]$ $X[.] \rightarrow YZ[.]$
adjungierend: Markierung gelöscht: Keller auswerten* Kern markiert: Z später auswerten	$(X, +) \rightarrow [(Y, -)](Z, -)$ $(X, +) \rightarrow [(Y, +)](Z, -)$	$X[.] \rightarrow Y\Phi[.Z_A]$ $X[.] \rightarrow Y[.Z_A]$
terminal: normal auswerten	$(X, -) \rightarrow x$	$X \rightarrow x$
Keller auswerten: $\forall K \in N$ ursprüngliche Rechtsexpansionen** Adjunktion simulieren** Letztes Element: Keller auflösen**		$\Phi[.K_R] \rightarrow K\Phi[.]$ $\Phi[.K_A] \rightarrow \Phi[.]K$ $\Phi[.] \rightarrow \epsilon$

Abbildung 6.3: Produktionen einer LIG modellieren eine RAG

Die Regeln illustrieren, wie das Nichtterminal- und Indexalphabet von G_2 auf der Grundlage von G_1 konstruiert wird. Nachfolgend erläutern wir die Erweiterungen im Detail und gehen darauf ein, wie mit Hilfe des Kellers der LIG in einer ersten Phase alle Adjunktionsoperationen aufgesammelt werden bis zur letzten adjungierenden Produktion. Dann erst wird bei der Auswertung des Kellers die gültige Satzform erzeugt.

1. Für das Startsymbol $(S, +)$ wird in G_2 $S[.]$ geschrieben (leerer Keller).
2. $RAGs$ besitzen die Eigenschaft, daß die Markierung entlang eines einzigen Pfades im Ableitungsbaum weitergegeben wird. Entlang dieses Pfades vererbt die LIG G_2 den Keller nach unten. Dabei werden im Keller die Symbole aufgesammelt, die von G_1 lokal nach rechts (Index „ R “) oder ganz rechts adjungiert (Index „ A “) erzeugt werden. Abbildung 6.4 (Phase I) illustriert diesen Vorgang. Für jedes Nichtterminalsymbol X in N_1 werden für das Indexalphabet der LIG zwei Symbole X_A und X_R hinzugefügt: $I := \{X_A | X \in N_1\} \cup \{X_R | X \in N_1\}$.

⁸ Lineare Index Grammatiken wurden auf Seite 38 eingeführt.

⁹ Die eckigen Klammern in der Tabelle spielen in beiden Formalismen eine unterschiedliche Rolle. Bei einer RAG wird damit Kern vom Adjunkt unterschieden. Bei einer LIG notieren die eckigen Klammern den mit einem Nichtterminalsymbol assoziierten Keller.

3. $N_2 := N_1 \cup \{\Phi\}$: Ein neues Nichtterminalsymbol Φ wird bei der mit * markierten Produktion eingeführt in dem Moment, wenn die Markierung verloren geht. Φ erhält dann den Keller übergeben, der von eigens für diesen Zweck definierten Produktionen ausgewertet wird (Die entsprechenden Produktionen sind mit ** markiert.): Symbole mit Index R sollen unmittelbar rechts vom markierten Pfad im Ableitungsbaum stehen und werden deshalb nach links vom Symbol Φ ausgewertet. Symbole mit dem Index A sind als adjungiert zu verstehen und werden deshalb rechts vom Symbol Φ realisiert. Abbildung 6.4 (Phase II) illustriert diesen Vorgang. Die vorgestellte Methode, den Keller auf- und abzubauen, stellt sicher, daß die Symbole in der Satzform von G_2 in derselben Reihenfolge generiert werden, wie in der Satzform von G_1 . Ist der Keller leer, wird er zusammen mit dem Symbol Φ gelöscht (Abbildung 6.4 Phase III).
4. Wenn Φ gelöscht wird, kommen nur noch Produktionen ohne Markierung (genau wie bei einer kontextfreien Grammatik) zur Anwendung. Mit der letzten terminalen Ersetzung endet der Ableitungsprozeß.

Das Alphabet bleibt damit endlich.

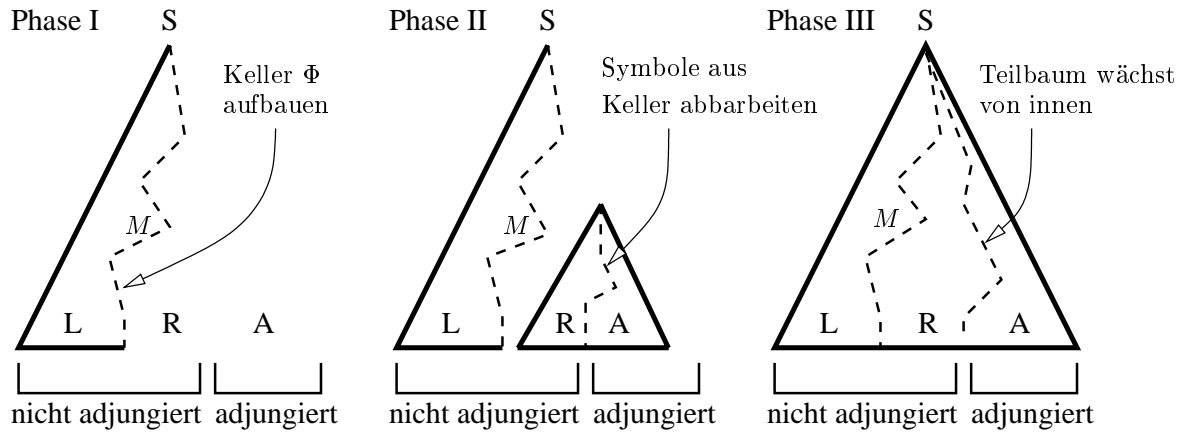


Abbildung 6.4: Mit dem Keller einer *LIG* läßt sich die Adjunktion von *RAGs* simulieren

BEISPIEL 26: Die Grammatiken G_1 (*RAG* in CNF) und G_2 (*LIG*) erzeugen dieselbe Sprache $\{a^n b^n c^n | n \geq 1\}$. Unter Anwendung der Transformationsregeln (Abbildung 6.3) werden die Regeln für G_2 erzeugt und um Regeln für die Kellerverwaltung ergänzt.

$$G_1 = (\{(A, -), (B, -), (C, -), (Q, -), (S, +), (T, +), (U, +)\}, \{a, b, c\}, P_1, (S, +))$$

$$G_2 = (\{A, B, Q, S, T, U, \Phi\}, \{a, b, c\}, \{B_R, C_A\}, P_2, S[.])$$

Die Konstruktion der Produktionen in P_2 zeigt Abbildung 6.5

Für die Menge aller Sprachen einer *RAG* gilt mit den Ergebnissen dieses Abschnitts die Beziehung:

$$\mathcal{CFL} \subset \mathcal{RAL} \subseteq_m \mathcal{CSL}$$

<i>RAG</i> : P_1	<i>LIG</i> P_2
$(S, +) \rightarrow [(T, +)](C, -)$	$S[.] \rightarrow T[..C_A]$
$(S, +) \rightarrow [(Q, -)](C, -)$	$S[.] \rightarrow Q\Phi[..C_A]$
$(T, +) \rightarrow [(A, -)](U, +)$	$T[.] \rightarrow AU[.]$
$(U, +) \rightarrow [(S, +)](B, -)$	$U[.] \rightarrow S[..B_R]$
$(Q, -) \rightarrow [(A, -)](B, -)$	$Q \rightarrow AB[.]$
$(A, -) \rightarrow a$	$A \rightarrow a$
$(B, -) \rightarrow b$	$B \rightarrow b$
$(C, -) \rightarrow c$	$C \rightarrow c$
	$\Phi[..B_R] \rightarrow B\Phi[.]$
	$\Phi[..C_A] \rightarrow \Phi[..]C$
	$\Phi[.] \rightarrow \epsilon$

Abbildung 6.5: Die Produktionen der *LIG* werden systematisch aus der *RAG* abgeleitet

Es liegt nahe, daß eine echte Teilmengenbeziehung $\mathcal{RAL} \subset mCSL$ vorliegt. Ein Kandidat für eine entsprechende Sprache $L \in mCSL, L \notin \mathcal{RAL}$ ist $L = \{a^n b^n c^n d^n | n \in \mathbb{N}\}$. Damit wäre zu erwarten, daß sich eine *RAG* leichter parsen läßt, als eine *mCSG*. Wie der folgende Abschnitt deutlich macht, ist dies tatsächlich der Fall.

6.6 Wortproblem und Parsing

Ein Parsingalgorithmus für \mathcal{RAL} kann auf der Grundlage des CYK-Algorithmus für kontextfreie Sprachen entwickelt werden. Entsprechend setzen wir voraus, daß die zugrundeliegende Grammatik in CNF vorliegt.

SATZ 31 (WORTPROBLEM FÜR \mathcal{RAL}): Sei G eine *RAG* und w ein Wort über einem terminalen Alphabet. Die Frage, ob $w \in L(G)$, ist in $O(n^4)$ entscheidbar.

BEWEIS: Der Parsing-Algorithmus¹⁰ CYK^+ in Abbildung 6.6 leistet das gewünschte. Er hat eine Laufzeit von $O(n^4)$. □

Erläuterung zur Invariante: Beim CYK-Algorithmus läßt sich aus jedem rückwärts abgeleiteten Symbol $A_{i,j}$ in der Parsingtable stets ein zusammenhängender (Teil-)String von w ableiten: $A_{i,j} \Rightarrow^* a_i \dots a_{i+j-1}$

Diese Invariante ist in der modifizierten Version CYK^+ nicht mehr garantiert. Zwar wird hier genauso aus einem Symbol A_{ij} ein Wort w_1 mit exakt j Symbolen abgeleitet, dieses Wort kann aber - bedingt durch die Adjunktionsoperation - in zwei in sich zusammenhängende Teilworte w_{11} und w_{12} zerfallen.

¹⁰ Der Parsing-Algorithmus wurde im Rahmen dieser Arbeit in der Programmiersprache JAVA implementiert.

```

Eingabe: Wort  $w$  der Länge  $n$ 
Ausgabe: true, wenn  $w \in L$ , sonst false.
00 begin
01   for  $i = 1$  to  $n$  do
02      $V_{i,1} := \{A^0 \mid \text{das } i\text{-te Symbol von } x \text{ ist } a \text{ und } ((A, -) \rightarrow a) \in P\}$ 
03   for  $j := 2$  to  $n$  do
04     begin
05       for  $i := 1$  to  $n - j + 1$  do
06         begin
07            $V_{i,j} := \emptyset$ 
08           for  $k := 1$  to  $j - 1$  do
09             begin
10                $V_{i,j} := V_{i,j} \cup \{A^0 \mid \text{wenn 1. } ((A, -) \rightarrow [(B, -)(C, -)]) \in P$ 
11                 und 2.  $B^0$  ist in  $V_{i,k}$ 
12                 und 3.  $C^0$  ist in  $V_{i+k,j-k}\}$ 
13                $V_{i,j} := V_{i,j} \cup \{A^x \mid \text{wenn 1. } ((A, +) \rightarrow [(B, +)(C, -)]) \in P$ 
14                 und 2.  $B^x$  ist in  $V_{i,k}$ 
15                 und 3.  $C^0$  ist in  $V_{i+k-x,j-k}\}$ 
16                $V_{i,j} := V_{i,j} \cup \{A^y \mid \text{wenn 1. } ((A, +) \rightarrow [(B, -)(C, +)]) \in P$ 
17                 und 2.  $B^0$  ist in  $V_{i,k}$ 
18                 und 3.  $C^y$  ist in  $V_{i+k,j-k}\}$ 
19                $V_{i,j} := V_{i,j} \cup \{A^{j-k+x} \mid \text{wenn 1. } ((A, +) \rightarrow [(B, +)](C, -)) \in P$ 
20                 und 2.  $B^x$  ist in  $V_{i,k}$ 
21                 und 3.  $C^0$  ist in  $V_{n+k-j+1-x,j-k}\}$ 
22                $V_{i,j} := V_{i,j} \cup \{A^{j-k+x} \mid \text{wenn 1. } ((A, +) \rightarrow [(B, -)](C, -)) \in P$ 
23                 und 2.  $B^x$  ist in  $V_{i,k}$ 
24                 und 3.  $C^0$  ist in  $V_{n+k-j+1-x,j-k}\}$ 
25             end
26         end
27     end
28   if  $S^{*,*} \in V_{1,n}$  then true else false
29 end

```

Abbildung 6.6: Algorithmus CYK⁺ zur Lösung des Wortproblems für RAGs

Um dennoch eine korrekte Parsingtabelle aufbauen und das Startsymbol rekonstruieren zu können, muß der Algorithmus stets in der Lage sein, aus einem Tabelleneintrag heraus die Positionen der beschriebenen Teilworte zu ermitteln. Tatsächlich reicht hierzu ein einziger Wert aus, den wir als hochgestellten Index x codieren, um die zwei Teilworte lokalisieren zu können. Dies geschieht unter Anwendung folgender Invariante:

$$A_{ij}^x \Rightarrow^* \underbrace{a_i \dots a_{i+j-1-x}}_{w_{11}} \dots \underbrace{a_{n-x+1} \dots a_n}_{w_{12}}$$

Aus dem Indexwert x kann also zurückgerechnet werden, welche beiden Teilworte aus einem Symbol A_{ij} abgeleitet werden können. Er beschreibt, wie viele Zeichen vom *yield*

infolge von Adjunktionsoperationen rechtsbündig generiert worden sind und nicht dort stehen, wo man es vom CYK-Verfahren her erwarten würde. Entsprechend viele Symbole fehlen beim klassischen CYK-*yield*. Für $x = 0$ liegt entsprechend der Fall vor, daß in dem abgeleiteten Teilbaum überhaupt nicht adjungiert wird.

Wenn wir die Zeilen 13-24 löschen, erhalten wir genau den CYK-Algorithmus für kontextfreie Grammatiken. Der Indexwert ist in diesem Falle immer 0, was gleichbedeutend damit ist, daß nirgendwo adjungiert wird.

Wegen der drei verschachtelten **for**-Schleifen kann der Algorithmus keine bessere Laufzeit besitzen als $O(n^3)$. Die gegenüber dem CYK-Algorithmus schlechtere Laufzeit von $O(n^4)$ kommt dadurch zustande, daß ein Feld $V_{i,j}$ in der Tabelle bedingt durch den Index bis zu $O(n)$ Elemente enthalten kann. Die Sicherstellung der Bedingungen 1. bis 3. kann deshalb bis zu $O(n)$ Zeit beanspruchen.

BEISPIEL 27 (PARSING EINER RAG): Die RAG mit folgenden Produktionen erzeugt die Sprache $\{ww|w \in \{a|b\}^+\}$.

$$\begin{aligned}
 (S, +) &\rightarrow [(C, +)](A, -) \quad | \quad [(A, -)](A, -) \quad | \quad [(D, +)](B, -) \quad | \quad [(B, +)](B, -) \\
 (C, +) &\rightarrow [(A, -)](S, +) \\
 (D, +) &\rightarrow [(B, -)](S, +) \\
 (A, -) &\rightarrow a \\
 (B, -) &\rightarrow b
 \end{aligned}$$

Abbildung 6.7 zeigt anhand einer Tabelle, wie das Wort *abaaba* vom Algorithmus erkannt wird. Die mit dem Symbol / gefüllten Felder werden vom Algorithmus nicht durchlaufen.

\i	1	2	3	4	5	6
j\	a	b	a	a	b	a
1	A^0	B^0	A^0	A^0	B^0	A^0
2	S^1	—	S^1	S^1	—	/
3	—	D^1	C^1	—	/	/
4	—	S^2	—	/	/	/
5	C^2	—	/	/	/	/
6	S^3	/	/	/	/	/

Abbildung 6.7: Tabelle zum Parsingvorgang

Abbildung 6.8 zeigt den zur Tabelle gehörenden Ableitungsbaum. Man sieht, daß die Kanten kreuzweise verlaufen, wenn eine adjungierende Produktion zur Anwendung kommt.

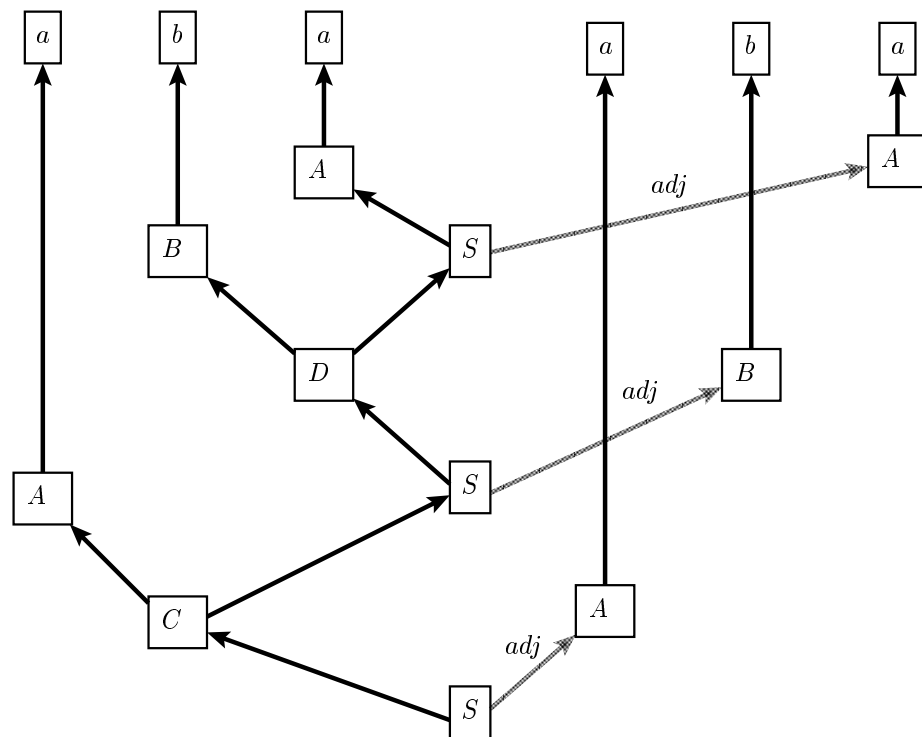


Abbildung 6.8: Ableitungsbaum eine *RAG* zu *abaaba*

6.7 Abgeschlossenheitseigenschaften von \mathcal{RAL}

Die Klasse \mathcal{RAL} ist abgeschlossen unter (inversen) Homomorphismen und Schnitt mit regulären Mengen. Die Sprachfamilie \mathcal{RAL} bildet also ein *Trio*.

LEMMA 32: Die Klasse \mathcal{RAL} ist abgeschlossen unter Homomorphismen.

BEWEIS: Sei $G = (N, T, P, (S, +))$ eine RAG , $L = L(G) \in \mathcal{RAL}$ und h ein Homomorphismus. Wenn wir G' auf der Basis von G so konstruieren, daß jede terminale Produktion $(A, \alpha) \rightarrow t$ in P durch $(A, \alpha) \rightarrow h(t)$ ersetzt wird, dann ist auch G' eine RAG . G' erkennt $h(L)$ und damit gilt $h(L) \in \mathcal{RAL}$.

☐

LEMMA 33: Die Klasse \mathcal{RAL} ist abgeschlossen unter Schnitt mit regulären Mengen.

BEWEIS: Der Beweis verwendet grundsätzlich dieselbe Idee, die bei kontextfreien Grammatiken zur Anwendung kommt und erweitert das Konzept der Tripelkonstruktion. Ein zusätzliches Attribut dient dazu, Informationen über den Verlauf vom Ableitungsprozeß zu transportieren, damit adjungierte Phrasen richtig in den Kontext eingebettet werden.

Sei dazu eine RAL $L \subseteq \Sigma^*$ gegeben. Dann wird $L \setminus \{\lambda\} = L'$ von einer RAG $G = (N, T, P, (S, +))$ in Chomsky Normalform erkannt. Betrachte weiter eine beliebige reguläre Menge $R \subseteq \Delta^*$. Sei M ein endlicher Automat $M = (Q, \Delta, \delta, s, F)$ mit $R = L(M)$.

Wir konstruieren nun eine RAG H auf der Basis von G und M , so daß $L(H) = L(G) \cap L(M)$. Wenn $T \cap \Delta = \emptyset$, dann ist $L' \cap R = \emptyset$ und $L \cap R$ ist entweder \emptyset oder $\{\lambda\}$. In beiden Fällen gilt $L \cap R \in \mathcal{RAL}$. Wir nehmen nachfolgend an, daß $T \cap \Delta \neq \emptyset$.

Sei $H = (U, T \cap \Delta, P', (S', +))$ mit einem neuen Startsymbol $(S', +)$ und den Nichtterminalsymbolen $U = \{(\langle p, q, r, A \rangle, \alpha) \mid p, q, r \in Q, (A, \alpha) \in N\} \cup \{S', +\}$.

Invariante: Abhängig von der Belegung von α beschreiben die Komponenten der Nichtterminalsymbole folgende Invariante:

- $(\langle p, q, r, A \rangle, -)$ - Aus diesem unmarkierten Nichtterminalsymbol kann *ein* zusammenhängendes terminales Wort w_1 abgeleitet werden, welches in G von $(A, -)$ ausgehend erkannt wird. Dasselbe Wort w_1 bringt den Automaten M dazu, vom Zustand p in den Zustand q überzugehen. Die Komponente r wird in diesem Ableitungsprozeß nicht verändert und spielt erst im markierten Fall eine Rolle:
- $(\langle p, q, r, A \rangle, +)$ - Aus diesem markierten Nichtterminalsymbol können *zwei* terminale Worte w_1 und w_2 abgeleitet werden, welche sich in G von $(A, -)$ aus ableiten lassen. Das Wort w_1 bringt den Automaten M dazu, vom Zustand p in den Zustand q überzugehen. Das Wort w_2 erkennt M beim Übergang vom Zustand r in den Endzustand. Der Endzustand wird in unserem Verfahren implizit erzeugt, wenn die letzte Adjunktion zur Anwendung kommt.

Bei der Konstruktion von P' ist sicherzustellen, daß aus einem markierten und einem unmarkierten Startsymbol jeweils eine zusammenhängende Satzform erzeugt wird, die einerseits von G erkannt wird, und bei der gleichzeitig der Automat von dem Startzustand s in den Endzustand f übergeht.

Bei einem unmarkierten Startsymbol ist dies einfach: Hier wird nicht adjungiert und die die Lösung stimmt mit der Situation überein, die von kontextfreien Grammatiken her bekannt ist.

Bei einem markierten Startsymbol ist sicherzustellen, daß die beiden ableitbaren Teilworte genau aufeinanderfolgen. Dies kann unter Ausnutzung der Invariante mit einer geeigneten Belegung des Starttupels erreicht werden, indem $r = q$ gesetzt wird. Bei der mit $*$ markierten Produktion wird die Markierung gelöscht. Es wird also das letzte Mal im Ableitungsprozeß adjungiert. In diesem Schritt wird gleichzeitig sichergestellt, daß der endliche Automat mit dem letzten Symbol stets in einem akzeptierenden Endzustand $f \in F$ mündet.

Die Produktionen von H müssen weiter sicherstellen, daß bei jeder Adjunktion die Satzform mit dem letzten Zustand der vorangehenden Adjunktion fortgesetzt wird. Um diesen Ansatz zu realisieren, definieren wir entsprechend allen zulässigen Varianten der Chomsky Normalform für Produktionen einer RAG die neuen Produktionen P' von H wie folgt:

$$\begin{aligned}
P' = & \{ (S', +) \rightarrow (\langle s, e, e, S \rangle, +) \mid \text{mit } e \in Q, \text{ falls } (S, +) \text{ Startsymbol} \} \\
& \cup \{ (S', -) \rightarrow (\langle s, f, f, S \rangle, -) \mid \text{mit } f \in F, \text{ falls } (S, -) \text{ Startsymbol} \} \\
& \cup \{ (\langle p, q, y, A \rangle, -) \rightarrow a \mid \text{falls } \delta(p, a) = q \text{ und } ((A, -) \rightarrow a) \in P \} \\
& \cup \{ (\langle p, q, y, A \rangle, -) \rightarrow (\langle p, r, y, B \rangle, -)(\langle r, q, y, C \rangle, -) \mid \\
& \quad \text{mit } p, q, r, y \in Q, \text{ falls } ((A, -) \rightarrow (B, -)(C, -)) \in P \} \\
& \cup \{ (\langle p, q, y, A \rangle, +) \rightarrow (\langle p, r, y, B \rangle, +)(\langle r, q, y, C \rangle, -) \mid \\
& \quad \text{mit } p, q, r, y \in Q, \text{ falls } ((A, +) \rightarrow (B, +)(C, -)) \in P \} \\
& \cup \{ (\langle p, q, y, A \rangle, +) \rightarrow (\langle p, r, y, B \rangle, -)(\langle r, q, y, C \rangle, +) \mid \\
& \quad \text{mit } p, q, r, y \in Q, \text{ falls } ((A, +) \rightarrow (B, -)(C, +)) \in P \} \\
& \cup \{ (\langle p, q, y, A \rangle, +) \rightarrow [(\langle p, q, z, B \rangle, +)](\langle y, z, z, C \rangle, -) \mid \\
& \quad \text{mit } p, q, y, z \in Q, \text{ falls } ((A, +) \rightarrow [(B, +)](C, -)) \in P \} \\
& \cup \{ (\langle p, q, y, A \rangle, +) \rightarrow [(\langle p, q, f, B \rangle, -)](\langle y, f, f, C \rangle, -) \mid \\
& \quad \text{mit } p, q, y \in Q, f \in F, \text{ falls } ((A, +) \rightarrow [(B, -)](C, -)) \in P \}^*
\end{aligned}$$

Die Grammatik H ist wieder eine RAG und erkennt genau die Wörter aus $L(G) \cap R$. \square

Bei inversen Homomorphismen wird von einer Sprache L das Urbild $h^{-1}(L)$ betrachtet:

LEMMA 34: *Die Klasse \mathcal{RAL} ist abgeschlossen unter inversen Homomorphismen.*

BEWEIS: Da sich ein inverser Homomorphismus über eine Sprache auch charakterisieren läßt als Komposition von Standardoperationen und einem Schnitt mit einer regulären Menge¹¹, folgt aus Lemma 33 unmittelbar auch hier die Abgeschlossenheit. \square

6.8 Abgrenzung zu anderen Arbeiten

Eine vergleichende Bewertung von RAG s gemessen an $mCSG$ s haben wir bereits vorgenommen. In der Literatur werden noch weitere Konzepte vorgestellt, die sich mit unserem Ansatz teilweise überschneiden. Diese beziehen sich auf erweiterte Automatenmodelle und die damit verbundenen Grammatikformalismen:

Kontextfreie Sprachen werden von nichtdeterministischen Kellerautomaten erkannt. Für das Erkennen einer mächtigeren Sprache - wie in unserem Fall die \mathcal{RAL} s - benötigt der erkennende Automat eine Erweiterung. Wie die Modellierung rechtsadjungierender Produktionen mit Hilfe von Produktionen einer linearen Indexgrammatik (Abbildung 6.3) deutlich gemacht hat, reicht ein zusätzlicher Keller oder eine zusätzliche Schlange aus, um das Verhalten von adjungierenden Produktionen mit einem Automaten zu simulieren.

¹¹ [Woo87, S. 429]

Eigenschaften von Sprachen und Maschinen mit einem Keller und einer Schlange vergleicht Brandenburg.¹² Von den dort diskutierten Sprachen können *RAGs* DYCK, PAL und COPY erkennen.

Automaten mit beliebig viele Zuständen und beliebig vielen Schlangen sind so mächtig wie eine Turing Maschine mit entsprechend vielen Arbeitsbändern. Für unsere Zwecke ist der Spezialfall erkennender Automaten mit einem einzigen Zustand interessant. Ergebnisse aus diesem Umfeld faßt die Arbeitsgruppe um Cherubini, Citrini, Reghizzi und Mandrioli in [CCRM90] und [CCRM91] zusammen:

Besitzen diese Maschinen nur eine Schlange, so kann deren Verhalten mit sogenannten BCF-Grammatiken¹³ simuliert werden. Analog zur Chomsky Hierarchie lassen sich Typ 0 bis Typ 3 Grammatiken unterscheiden. Im Falle einer Typ 2 Grammatik sind Produktionen von der Form $A \rightarrow xB$ mit einem Wort x über Terminalsymbolen und einem Wort B , welches aus Nichtterminalsymbolen besteht. Die Ersetzungsregel besteht aus zwei Schritten:

1. Ersetze die linke Seite A durch x
2. Füge das Wort B rechts an den bis dahin abgeleiteten String an.

Weil der zugrundeliegende Automat eine Schlange verwendet, verlangen die Autoren, daß die Produktionen *leftmost* angewandt werden. Dies ist - anders als bei kontextfreien Grammatiken - wesentlich für die Mächtigkeit des Formalismus. Während kontextfreie Grammatiken von Haus aus die Konfluenzeigenschaft erfüllen, ist dies für *BCF*-Grammatiken nicht der Fall. Die *leftmost* Eigenschaft stellt sicher, daß die Reihenfolge der zu ersetzenden Symbole eindeutig wird. Es läßt sich zeigen, daß unterschiedliche Ausführungsreihenfolge unterschiedliche Sprachen generieren.¹⁴

Die Menge der von *BCF*-Grammatiken erkannten Sprachen ist mit der Klasse \mathcal{CFL} unvergleichbar: *BCF*-Grammatiken erkennen auch nichtkontextfreie Sprachen, aber Palindrome können sie nicht beschreiben. Für Palindrome ist eine *BCF*-Grammatik vom Typ 1 erforderlich.¹⁵

Es liegt nahe, den Ansatz zu erweitern und Automaten mit einer kombinierten Speichereinheit zu untersuchen.¹⁶ Ein Dequeue¹⁷-Automat besitzt eine Kombination von *LIFO* und *FIFO* Speicher. Die Autoren untersuchen sowohl Automaten mit unbeschränkt vielen Zuständen, welche die rekursiv aufzählbaren Sprachen erkennen, als auch solche, die mit einem Zustand auskommen. Nur letztere sind für uns von Interesse. Die zu diesem Automaten gehörende Grammatik nennen die Autoren *BD*-Grammatik¹⁸. *BD*-Grammatiken vereinen die Möglichkeiten von kontextfreien und *BF*-Grammatiken.

¹² [Bra80], [Bra81], [Bra86], [Bra88b]

¹³ für: *breadth-first context-free*

¹⁴ [BCR91]

¹⁵ Weitere Eigenschaften von BCF-Sprachen untersucht Mäkinen in [Mäk90]

¹⁶ [BCR95]

¹⁷ für: *double-ended queue*

¹⁸ für: *Breadth-Depth*-Grammatik

In einem weiteren Schritt schließlich erweitern die Autoren Automaten mit mehreren Kellern und Schlangen. Sinnvollerweise ist die Anzahl der Zustände auch hier auf 1 beschränkt. Die zu diesen Automaten gehörenden Sprachen werden *ACF*-Grammatiken¹⁹ genannt. Die Keller und Schlangen besitzen eine feste Reihenfolge, eine Produktion kann beim Ersetzungsschritt an jedem Speicher eine Modifikation durchführen. Für unsere Anwendungen ist dieser Ansatz zu mächtig und deshalb uninteressant. Für den Sonderfall einer *LL*(1)-Grammatik stellen die Autoren Linearzeitalgorithmen vor, eine allgemeine Lösung für beliebige Grammatiken geben sie aber nicht an.

Als Beispiel für praktische Anwendungen nennen die Autoren Scheduling-Prozesse.²⁰ Es liegt nahe, daß diese sich mit einer Schlange besser als mit einem Keller modellieren lassen.

Von all den vorgestellten Verfahren kommen *BD*-Grammatiken den *RAG*s am nächsten. Tatsächlich lassen sich Produktion einer *RAG* (gegeben in CNF) in Produktionen einer *BD*-Grammatik übersetzen. Wir geben nachfolgend eine Beweisskizze an, die Notation orientiert sich an [BRC99].

Sei $X \rightarrow [Y]Z$ eine Produktion einer *RAG*. Mit einem zusätzlichen nichtterminalen Hilfssymbol X_1 läßt sich die Wirkungsweise der Produktion durch zwei Produktionen einer *BD*-Grammatik simulieren:

1. $X \rightarrow (X_1Y)_S$ - Das Symbol Y wird wie bei einem kontextfreien Ersetzungsschritt (der Index S steht für *Stack*) lokal an der Stelle von X eingefügt, das Hilfssymbol X_1 erzwingt zusammen mit der den *BD*-Grammatiken inhärenten *leftmost*-Bedingung, daß als nächstes die folgende Produktion ausgeführt wird.
2. $X_1 \rightarrow (Z)_Q$ - Das Symbol X_1 wird gelöscht und ganz rechts das Symbol Z eingefügt. Der Index Q (für *Queue*) markiert, daß das Symbol Z im Sinne einer Warteschlange einzufügen ist, was in unserer Terminologie einer Rechtsadjunktion gleichkommt.

Die Restriktionen, die in *RAG*s durch die Markierungen ausgesprochen werden, lassen sich in das Alphabet von *BD*-Grammatiken hineincodieren. Die rechtsadjungierenden Sprachen sind also in den *BD*-Sprachen enthalten.

Das Wortproblem für *BD*-Sprachen lösen die Autoren nur für eine Teilklasse: Lediglich für solche Sprachen, die mit Produktionen einer *BD*-Grammatik in Greibach Normalform beschrieben werden können, welche die *LL*(1)-Eigenschaft erfüllen, geben sie genau wie bei den *ACF*-Grammatiken Linearzeitalgorithmen an.²¹ Das Erzeugen einer Greibach-Normalform kann (zu) teuer werden, weil unter Umständen exponentiell viele Produktionen nötig sind. Dies ist bei einer Chomsky-Normalform nicht der Fall. Sie wächst linear in der Größe der Grammatik. Für den Fall einer *RAG* können wir dann – wie in Abbildung 6.6 gezeigt – das Wortproblem in $O(n^4)$ lösen.

¹⁹ für: *augmented context-free grammar*

²⁰ [BRC99]

²¹ [BCR91]

6.9 Zusammenfassung

Das vorangehende Kapitel erweitert die Aufgabenstellung der derivationalen Beschreibung von Syntaxgraphen um einen weiteren Aspekt. Die Ergebnisse der Arbeit werden zusammengeführt, um die Frage zu beantworten, wie ein für \mathcal{NL} angemessener Wortgrammatikformalismus aussehen könnte. Bei der Spezifikation nutzen wir die Ergebnisse, die wir bei den Untersuchungen von Syntaxgraphen mit Graphgrammatiken gefunden haben.

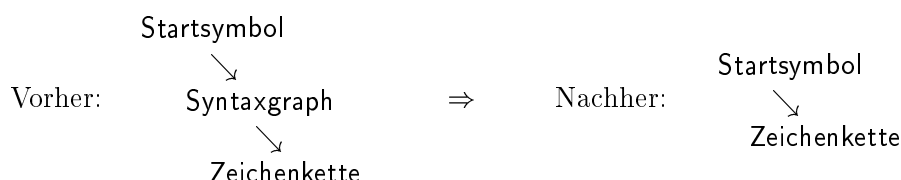


Abbildung 6.9: Wortgrammatik: zwei Phasen werden zusammengefaßt

Technisch gesehen wird das Ziel verfolgt, für die zwei Ableitungsphasen, die in den Kapiteln 4 und 5 zur Anwendung kommen, einen einzigen Mechanismus zu finden. Dies gelingt mit der Entwicklung eines Wortgrammatikformalismus *RAG*, welcher echt zwischen den kontextfreien und schwach kontextsensitiven Sprachen (*mCSG*) liegt. Das sich aus den Graphgrammatik-Betrachtungen ergebene Anforderungsprofil wird von der Klasse \mathcal{RAL} erfüllt. Außerdem zeigen wir, daß unser Vorschlag auch aus algorithmischer Sicht interessant ist: Auf der Basis der CYK-Methode wird ein effizienter $O(n^4)$ -Parsingalgorithmus vorgestellt.

Wenn für ausgewählte Anwendungen nicht die Syntaxgraphen, sondern lediglich lineare Wortsequenzen von Interesse sind, dann lassen sich Sätze auf diese Weise effizienter beschreiben und erkennen, als der zugrundeliegende Ansatz über Graphgrammatiken und Kettengraphen es ermöglicht. Dies ist dann der Fall, wenn zum Beispiel sprachspezifische Lösungen gesucht werden. Der generische Ansatz, der sich mit dem Zugang über Graphgrammatiken und Syntaxgraphen eröffnet, wird dann nicht ausgenutzt.

The theory of formal languages constitutes the stem or backbone of the field of science now generally known as theoretical computer science. In a very true sense its role has been the same as that of philosophy with respect to science in general: it has nourished and often initiated a number of more specialized fields. In this sense formal language theory has been the origin of many other fields. However, the historical development can be viewed also from a different angle. The origins of formal language theory, as we know it today, come from different parts of human knowledge. This also explains the wide and diverse applicability of the theory.

ROZENBERG UND SALOMAA IN [Roz97]

7 Zusammenfassung der Arbeit

7.1 Überblick

Die vorliegende Arbeit zeigt, daß die Graphentheorie und insbesondere Graphgrammatiken neue und anschauliche Zugänge eröffnen, um syntaktische Strukturen von Sätzen aus der Menge der natürlichen Sprachen \mathcal{NL} zu beschreiben. Die Untersuchung dieses Themas wird dadurch motiviert, daß in der Linguistik intensiv von Baumstrukturen mit koindizierten Knoten Gebrauch gemacht wird, um syntaktische Beziehungen zwischen den Komponenten eines grammatischen Satzes zu modellieren. Der hier erbrachte Nachweis der Einsatzmöglichkeiten von Graphgrammatiken erleichtert die Umsetzung von Ergebnissen der linguistischen Forschung für Algorithmen zur Sprachverarbeitung. Dieser Ansatz ist in der Literatur noch nicht systematisch untersucht worden.

Dies mag daran liegen, daß bei der Betrachtung von Sätzen aus \mathcal{NL} der lineare Aspekt zunächst aus naheliegenden Gründen im Vordergrund steht. Die hierarchische Sichtweise wird eher als Seiteneffekt, gegeben durch die Ableitungsstruktur einer formalen Sprache, zur Kenntnis genommen. Mit dem universalgrammatischen Projekt, wie es Chomsky in *Government and Binding* (GB) verfolgt, kommt der syntaktischen Struktur jedoch eine ganz eigenständige Bedeutung zu. Mit ihrer Hilfe können Zusammenhänge und Ähnlichkeiten zwischen verschiedenen Sprachen erkannt und deutlich gemacht werden, die bei der üblichen sequentiellen Herangehensweise verborgen bleiben. Linguisten, die sich auf die Untersuchung unterschiedlicher Sprachfamilien spezialisiert haben, können sich so auf der Basis einer gemeinsamen Terminologie austauschen. Dies alles rechtfertigt es, syntaktische Strukturen selbst zum Gegenstand der Untersuchung zu machen.

Unser Interesse gilt deshalb primär der Beschreibung solcher Strukturen und nur sekundär der Charakterisierung der linearen Abfolge von Terminalsymbolen. Zu diesem Zweck führen wir den Begriff des *Syntaxgraphen* ein, der das in der Linguistik verbreitete Kon-

zept des *Syntaxbaums* erweitert. Für eine Charakterisierung von Syntaxgraphen sind wir an ihren invarianten graphentheoretischen Eigenschaften interessiert, wozu wir alle entsprechenden Begriffe in Erwägung ziehen. Die Suche nach solchen Invarianten kommt dem Ziel vieler linguistischer Arbeiten entgegen, eine Menge von syntaktischen Prinzipien und Regeln zu weniger komplexen Beschreibungen hin zu vereinfachen. Zahlreiche Arbeiten in der Tradition von GB beschäftigen sich mit diesem Thema, welches von Chomsky als *minimalist program* ausgerufen wurde. Aber weder der Schritt vom Syntaxbaum zum Syntaxgraphen, noch der Ansatz einer Beschreibung über Graphgrammatiken wird in den entsprechenden Veröffentlichungen vollzogen.

Indem wir die Menge der Syntaxgraphen als Graphsprache betrachten, können wir zeigen, daß sich syntaktische Strukturen mit Graphgrammatiken beschreiben lassen. Das eröffnet die Möglichkeit, vielfältige syntaktische Phänomene aus den natürlichen Sprachen über einen einzigen Formalismus zu beschreiben.

Über den Aspekt der Charakterisierung von Syntaxgraphen hinaus ist man an effizienten erkennenden Algorithmen für Sprachen interessiert. Aus den Eigenschaften der in dieser Arbeit entwickelten Graphgrammatiken folgt, daß sich über diesen Zugang auch das Wortproblem für syntaktische Strukturen in Polynomialzeit lösen läßt.

7.2 Zum Umgang mit empirischen Problemen

Der Nachweis dieser Eigenschaften benötigt viel Vorarbeit. Wir erläutern, daß dies in ganz unterschiedlichen methodischen Herangehensweisen der Disziplinen Linguistik und Informatik an den Forschungsgegenstand \mathcal{NL} begründet liegt. Diese Methoden werden in der Literatur als *repräsentationell* und *derivationell* unterschieden. Letztere werden üblicherweise im Zusammenhang mit formalen Sprachen verwendet, in GB wird dagegen der komplementäre Ansatz verfolgt. Um die Einsatzmöglichkeiten von Graphgrammatiken zu untersuchen, muß also nicht etwa nur ein derivationeller Formalismus in einen anderen übersetzt werden.¹ Zusätzlich ist hier auch die zugrundeliegende Methode anzupassen, und dies ist mit sehr viel mehr Aufwand verbunden. Es stellt einen wesentlichen Inhalt dieser Arbeit dar, den Transfer zwischen den beiden Formalismen zu dokumentieren.

Darüber hinaus erschweren eine Reihe von Eigenschaften des zu analysierenden Datenmaterials die formale Herangehensweise: Die Menge \mathcal{NL} ist nur empirisch zugänglich. Anders als bei formalen Sprachen ist sie grundsätzlich *nicht* durch eine Grammatik gegeben, sondern vielmehr durch eine Sprachgemeinschaft, die sich dieser Sprache zum Zwecke der Kommunikation bedient. Zudem wird die Formalisierung dadurch behindert, daß die zugänglichen Daten unscharf sind: Es gibt sprachliche Äußerungen, von denen nicht klar ist, ob sie als natürlichsprachig angesehen werden sollen, oder nicht. Ein allgemeingültiges Entscheidungskriterium für die Wohlgeformtheit von sprachlichen Äußerungen liegt faktisch nicht vor. Diese Umstände machen es notwendig, in einem grundlegenden Arbeitsgang aus den gegebenen linguistischen Daten heraus eine Grammatik zu rekonstruieren. Das erschwert nicht nur die Konzeption eines beschreibenden Formalismus,

¹ Dies wäre dann der Fall gewesen, wenn GB zum Beispiel als kontextsensitive Grammatik gegeben wäre, und wir versucht hätten, einen schwach kontextsensitiven Kern davon herauszuarbeiten.

es macht auch eine abschließende Verifikation eines wie auch immer beschaffenen Modells von vornherein logisch unmöglich. Formale grammatische Beschreibungen von \mathcal{NL} können deshalb nie mehr sein, als eine Approximation.

Wir umgehen dieses Problem dadurch, daß wir uns mit GB von vornherein an einem Modell für syntaktische Strukturen orientieren. Deshalb brauchen wir uns mit empirischen Fragestellungen nicht auseinanderzusetzen.

7.3 GB besitzt ein generisches Potential

Aus algorithmischer Sicht ist GB vor allem deshalb interessant, weil hier mit dem Konzept einer Universalgrammatik ein *generischer Ansatz* verfolgt wird. Mit dem Anspruch explanatorischer Adäquatheit zielt GB darauf ab, eine umfangreiche Menge linguistischer Daten zu beschreiben. Es wird angestrebt, die syntaktischen Strukturen aller Sprachen auf der Welt auf der Grundlage einer einzigen, allen Sprachen gemeinsamen, abstrakten Beschreibung zu charakterisieren. Die einschlägige Literatur belegt, daß die universellen Gemeinsamkeiten zwischen verschiedenen Sprachen sehr viel umfangreicher sind, als es auf den ersten Blick den Anschein hat. Unterschiede zwischen den Sprachen werden in GB mit Hilfe von sprachspezifischen Parametern modelliert.

Das Ziel der Herausarbeitung effizienter Datenstrukturen und Algorithmen steht dabei in GB *nicht* im Mittelpunkt des Interesses, sondern vielmehr die Entwicklung einer konsistenten Menge von Regeln und Prinzipien, die es ermöglichen, gute Sätze von schlechten Sätzen zu unterscheiden. Deshalb muß es dem nicht mit der linguistischen Terminologie vertrauten Leser in der Regel schwer fallen, aus den Ergebnissen von GB eine Spezifikation für linguistische Algorithmen abzuleiten.

Die Ergebnisse dieser Arbeit zeigen durch eine Formalisierung der beteiligten Mechanismen Wege auf, wie das generische Potential von GB für praktische Anwendungen nutzbar gemacht werden kann. Dies stellt effiziente Algorithmen zur Sprachverarbeitung wie *automatische Sprachgenerierung*, *Übersetzung* oder *Grammatikprüfung* dahingehend in Aussicht, als daß für die syntaktische Komponente einer jeden natürlichen Sprache *NL* *nicht* mehr von Grund auf ein eigenes Verfahren entwickelt werden muß.

7.4 Formalisierung von GB und Analyse von Syntaxgraphen

Die Entwicklung dieses Ziels erfolgt in zwei Schritten, denen jeweils ein eigenes Kapitel dieser Arbeit gewidmet ist: Zunächst ist es nötig, die Regeln und Restriktionen von GB in eine graphentheoretische Lesart zu übersetzen. In einem zweiten Schritt wird diese formalisierte Fassung von GB in die Terminologie einer Graphgrammatik übersetzt. Zu diesem Zweck entwickeln wir den in dieser Arbeit zentralen Begriff des *Syntaxgraphen*. Bei dem Entwurf von Algorithmen zur Sprachverarbeitung liegt es nahe, auf Syntaxgraphen als zugrundeliegende Datenstruktur zurückzugreifen. Durch sprachspezifische Verfeinerungen können sie individuell an die jeweiligen Anforderungen angepaßt werden.

Um den Umfang der Arbeit in Grenzen zu halten, beschränken wir uns auf die Modellierung eines Fragments der englischen Sprache. Dabei reicht es nicht aus, zur Formalisierung von GB die entsprechenden Regeln und Prinzipien aus der linguistischen Literatur abzuschreiben. Da sie nicht als Regeln einer formalen Sprache konzipiert sind, ist es nötig, sie einzeln zu analysieren und dabei ihre Rolle für die Charakterisierung syntaktischer Strukturen herauszuarbeiten. Weil eine solche Umsetzung nicht automatisiert werden kann, muß sie konstruktiv erbracht werden. Natürlich gibt es mehr als einen Weg, dieses Ziel zu verfolgen. Wir machen in der Arbeit deutlich, wo alternative Herangehensweisen möglich sind und motivieren die von uns gesetzten Schwerpunkte.

Beispielsweise ist schon die Wahl von edNCE-Graphgrammatiken willkürlich, weil die Literatur unterschiedliche Typen von Graphgrammatiken unterscheidet. Wir machen jedoch deutlich, daß unsere Resultate *robust* sind, sich also auf andere Graphgrammatik-Formalismen übertragen lassen. Dies ist deshalb der Fall, weil wir es mit einer eingeschränkten Klasse von Graphen zu tun haben: Syntaxgraphen sind *einfach zusammenhängend* und *gradbeschränkt*.

Als einschränkendes Kriterium für die Möglichkeiten effizienter Approximation erweisen sich die Separatoreigenschaften von Syntaxgraphen – zumindest, solange diese strikt auf der Grundlage von GB definiert werden. Wir zeigen, daß manche Regularitäten nicht mit konfluenten Graphgrammatiken simuliert werden können. Als Ursache für diese Beschränkungen lassen sich universelle Phänomene benennen, die in allen bekannten Sprachen vorkommen. Glücklicherweise können wir jedoch diese Schwierigkeiten auf einige esoterische Fälle von praktisch irrelevanten Satzkonstruktionen reduzieren, die im sprachlichen Alltagsgebrauch keine Rolle spielen. Die entsprechenden Beweise interpretieren wir dergestalt, daß GB tendenziell zuviel beschreibt, also zu mächtig ist. Umgekehrt zeigen wir exemplarisch, mit welchen technischen Mitteln Produktionen einer Graphgrammatik die praktisch relevanten Regularitäten beschreiben können. In dieser Einschätzung beider Herangehensweisen als Approximation für *NL* ‘von oben’ und ‘von unten’ spiegelt sich der grundsätzliche Unterschied zwischen der repräsentationellen und der derivationellen Methode wieder.

Damit empfiehlt sich für praktische Anwendungsgebiete eine pragmatische Herangehensweise: Alle wichtigen syntaktischen Fragestellungen lassen sich algorithmisch mit boundary Graphgrammatiken beschreiben. Dazu zeigen wir, daß die in dieser Arbeit entwickelte Graphgrammatik alle Bedingungen erfüllt, welche die GB-Theorie formuliert. In diesem Sinne löst unser Zugang über Graphgrammatiken das in der linguistischen Literatur diskutierte Format-Problem.

7.5 Elementprobleme sind in Polynomialzeit lösbar

Bis an diesen Punkt beschäftigen wir uns mit generierenden Aspekten, also solchen Techniken und Werkzeugen, die die Menge syntaktischer Strukturen mit Hilfe von Graphgrammatiken *konstruktiv* beschreiben. Darüber hinaus ist auch das *Elementproblem* interessant, welches die Aufgabenstellung umkehrt: Hier ist zu überprüfen, ob ein gegebener Graph eine syntaktische Struktur darstellt und damit auch, ob die dazugehörige

Zeichenkette einen grammatischen Satz repräsentiert.

Die Modellierbarkeit syntaktische Strukturen mit Graphgrammatiken zeigt einen Weg auf, diese Fragen unter Anwendung von bereits bekannten Ergebnissen effizient zu lösen. Dabei erweist es sich als hilfreich, vorab gezeigt zu haben, daß boundary Graphgrammatiken syntaktische Strukturen beschreiben. Zusammen mit anderen Graphinvarianten stellt dies sicher, daß das Wortproblem in polynomialer Zeit lösbar ist. Zwar führen bekannte Verfahren nur zu Algorithmen mit großen Exponenten. Die praktische Anwendbarkeit der Verfahren bleibt aber unangetastet, solange n klein ist. Dies ist bei Sätzen natürlicher Sprachen typischerweise der Fall.

Die gezeigten Parsingeigenschaften sind nützlich für die Implementation von Systemen, welche den Prozeß des ‘automatisierten Verstehens’ natürlicher Sprache modellieren. Hier stellt die syntaktische Analyse der Eingabe eine wesentliche Komponente in der Abarbeitung dar.

7.6 Wortgrammatiken

Für sprachspezifische Anwendungen – wenn also nur Verfahren für eine ausgewählte Sprache gesucht sind – reicht es unter Umständen aus, mit einer Wortgrammatik zu arbeiten, die die terminalen Zeichenketten direkt beschreibt. Die explizite Sequentialisierung syntaktischer Strukturen zu linearen Wortfolgen eines grammatischen Satzes braucht dann nicht notwendig durch eine eigene Funktion *yield* beschrieben zu werden. Wir entwickeln ein Verfahren, mit dem sich Produktionen einer Graphgrammatik für Syntaxgraphen systematisch so umformulieren lassen, daß die Funktionalität der *yield*-Funktion in die neue Graphgrammatik hineincodiert wird. Für die Modellierung solcher Kettengraphen müssen – aus der Sicht einer Wortgrammatik – auch kontextsensitive Mechanismen herangezogen werden, denn Ergebnisse der Literatur zeigen, daß kontextfreie Grammatiken für diesen Zweck nicht mächtig genug sind. Es verbietet sich aber schon unter dem Gesichtspunkt der Effizienz, als Alternative auf kontextsensitive Grammatiken zurückzugreifen. Wir stellen die entsprechenden Argumente aus der Literatur zusammen und entwickeln aus den Mechanismen der boundary Graphgrammatik für Syntaxgraphen heraus einen neuen Formalismus für eine Wortgrammatik, welche mächtiger ist als die kontextfreien Grammatiken.

Diese im letzten Kapitel dieser Arbeit entwickelten *Right Adjunct Grammars* erkennen alle die Konstruktionen, die in der Literatur immer wieder als Standardbeispiele für nicht kontextfreie Phänomene in \mathcal{NL} aufgezählt werden. Wir zeigen für diesen Grammatiktyp einige Eigenschaften, welche eine Kategorisierung gegenüber anderen Grammatikformalismen ermöglichen. Die beschriebene Klasse liegt echt zwischen den kontextfreien und den schwach kontextsensitiven Sprachen. Bei letzteren kann das Wortproblem in $O(n^6)$ gelöst werden. Dagegen können wir zeigen, daß das Wortproblem für *Right Adjunct Grammars* in $O(n^4)$ lösbar ist. Das vorgestellte Verfahren stellt eine Erweiterung des CYK-Algorithmus für kontextfreie Sprachen dar.

7.7 Die Formalisierung liefert auch linguistische Einsichten

Interessanterweise lassen sich aus der graphentheoretischen Beschreibung Verarbeitungsaspekte von Sprache vorhersagen, die sich in empirischen Beobachtungen widerspiegeln: Wir weisen auf einen Zusammenhang zwischen der Separatoreigenschaft eines Syntaxgraphen und dem vom *ideal speaker* empfundenen Schwierigkeitsgrad des dazugehörigen Satzes hin. Es zeigt sich: Sätze werden nicht dadurch kompliziert, daß sie lang sind. Sie sind dann schwieriger zu verarbeiten, wenn der Separator des dazugehörigen Syntaxgraphen groß wird. Dieser wächst infolge von Querbeziehungen, welche sich aus linguistischer Sicht auf Bewegungsprozesse oder bindungstheoretische Beziehungen zurückführen lassen. Diese Beobachtung deckt sich mit der Intuition: Der Aufwand für die Verarbeitung steigt dann, wenn eine syntaktische Struktur viele Querkanten enthält, die den zugrundeliegenden Baum im Syntaxgraphen auf eine Weise überlagern, daß viele Kanten gleichsam ‘parallel’ angeordnet sind. Verlaufen viele Querkanten dagegen ‘hinter’- oder ‘nebeneinander’, so daß der Separator nur in kleinem Umfang wächst, dann ist auch ein langer Satz mit vertretbarem Aufwand zu verarbeiten.

Bei der Suche nach relevanten Parametern zur formalen Bewertung der Güte eines Satzes sollten demnach die Separatoreigenschaften des zugrundeliegenden Syntaxgraphen berücksichtigt werden.

Als weitere relevante Größe bei der Gütebewertung kristallisiert sich die Art und Weise heraus, wie Phrasen miteinander verknüpft sind. Wir zeigen am Beispiel (fast) isomorpher Phrasenstrukturen, daß diese abhängig von der Rolle der Kanten unter Verarbeitungspunkten sehr unterschiedlich bewertet werden.

Diese beiden Beobachtungen machen deutlich, daß die Formalisierung linguistischer Ergebnisse auch rückwirkend Hinweise auf syntaktische Zusammenhänge liefern kann, die nicht offensichtlich sind. Derartig motivierte Thesen bieten sich als Grundlage neuer empirischer Untersuchungen an.

7.8 Abgrenzung zu anderen Arbeiten

Die Formalisierung von GB machen sich auch andere Arbeiten in der Literatur zur Aufgabe. Diese verwenden in der Regel Varianten von Logiksprachen, was sehr präzise Ausdrucksformen ermöglicht, andererseits aber zu unübersichtlichen Termausdrücken führt. In dieser Hinsicht besitzt die graphentheoretische Herangehensweise offensichtlich Vorteile, insbesondere was die Möglichkeit betrifft, Ergebnisse zu veranschaulichen. Dies kommt auch dem interdisziplinären Erfahrungsaustausch entgegen.

Parallelen bezüglich des Ansatzes, Sätze über Graphsprachen zu beschreiben, gibt es mit den Arbeiten zu *Tree Adjoining Grammars* (TAGs). Dieser Grammatikformalismus beschreibt keine Zeichenketten, sondern deren zugrundeliegende Baumstrukturen. Die Erweiterung von Bäumen zu Graphenstrukturen findet aber auch hier nicht statt. Primär werden formalsprachliche Eigenschaften untersucht und entsprechend auf universalgrammatische Aspekte von GB nur sekundär eingegangen.

Ein Automatenmodell, welches die Sprachen einer *Right Adjunct Grammar* erkennt, benötigt zusätzlich zu einem Keller noch ein weiteres Speichermedium. Wir gehen auf einer Reihe entsprechender Erweiterungen ein, die in der Literatur diskutiert werden. Das gibt uns die Möglichkeit, den eigenen Ansatz zu klassifizieren.

Als Alternativen zu der Wahl von GB als zugrundeliegendes Grammatikmodell bieten sich Arbeiten zum *minimalist program* oder zur *optimality theory* an. Hier ist aber die Vereinheitlichung von Konzepten und Notationen noch längst nicht so weit vorangeschritten, wie es bei GB der Fall ist.

7.9 Ausblick

Mit Hilfe von Graphgrammatiken lassen sich syntaktische Strukturen, die sich an den GB-Restriktionen orientieren, auch über eine formale Sprache derivationell beschreiben. Dies erleichtert die Klassifizierung und nachträgliche Erweiterung um zusätzliche Regularitäten. Da das Wortproblem in Polynomialzeit lösbar ist, kann die Korrektheit von syntaktischen Strukturen effizient überprüft werden. Bezüglich der Beschreibung linearer Zeichenketten wird deutlich, daß die Erweiterung um kontextsensitive Mechanismen nicht notwendig mit einer Verschlechterung der Komplexität erkaufte werden muß, wenn man diese mit kontextfreien Verfahren vergleicht.

Der Ansatz über eine universalgrammatische Beschreibung syntaktischer Regularitäten weist den Weg zum generischen Entwurf linguistischer Algorithmen: In Zukunft wird die Herangehensweise über parametrisierte Techniken viel Arbeit und Zeit ersparen, während mit klassischen Ansätzen für jede behandelte Sprache ein eigenes Verfahren entwickelt werden muß.

Der Weg zu einer konkreten Implementation ist noch weit: In den einführenden Betrachtungen machen wir deutlich, daß mit der Fokussierung auf syntaktische Fragestellungen nur eine Komponente dessen beschrieben wird, was natürliche Sprachen charakterisiert. Der Entwurf von sprachverarbeitenden Systemen erfordert es über die Zielsetzung dieser Arbeit hinaus, die von uns formalisierten generischen Grundlagen für syntaktische Regularitäten in ein umfassenderes Modell linguistischer Beschreibung einzubetten. In diesem müssen neben anderen auch semantische und morphologische Aspekte berücksichtigt werden.

Auch wenn sich dabei eine große Zahl sprachlicher Erscheinungen für immer der Formalisierbarkeit entziehen wird, sehen wir für unseren Ansatz vielfältige praktische Anwendungen. Vor der Umsetzung der Vision eines universellen Übersetzers gibt es weitaus naheliegendere und realistischere Ziele: Dazu gehört insbesondere die automatisierte Textgenerierung, wo unter Umständen kurzfristig und mit hohem Durchsatz Texte in verschiedenen Sprachen benötigt werden, die schnell an Aktualität verlieren können und dann verworfen werden. Diese Charakterisierung trifft auf Wetterberichte, Börsennachrichten, Informationsdienste und auch auf Bedienungsanleitungen zu. Voraussetzung für so eine Umsetzung ist das Vorliegen der zu verbalisierenden Information in einer Datenbank oder als Instanz einer universellen Beschreibungssprache.

Eine Parsingkomponente wird dann nicht benötigt, was eine wesentliche Vereinfachung des Anforderungsprofils gegenüber einem universellen Übersetzer darstellt. Durch den generischen Kern sind derartig konzipierte Systeme um Funktionen für zusätzliche Sprachen erweiterbar, ohne daß dabei der grundlegenden Entwurf angepaßt werden muß.

Wir sind davon überzeugt, daß der Ansatz über die Graphentheorie bei dieser Entwicklung weit mehr leistet, als daß er nur die Anschauung beim Entwurf linguistischer Systeme erheblich erleichtert. Darüber hinaus hat diese Arbeit gezeigt, daß Syntaxgraphen *die* naheliegende Herangehensweise zur Modellierung syntaktischer Strukturen von natürlichen Sprachen darstellen und deshalb glauben wir, daß dieser Zugang vielfältige Perspektiven bietet.

Anhang

Verzeichnis der Definitionen

Verzeichnis der Prinzipien von GB

Verzeichnis der Abbildungen

Literatur

Verzeichnis der Definitionen

1	<i>edNCE</i> -Graphgrammatik	20
2	Konkreter Ersetzungsschritt	21
3	Ableitungsschritt	22
4	ableitbar	22
5	Sprache einer Graphgrammatik	22
6	Äquivalenz von Graphgrammatiken	23
7	Konfluenz	24
8	boundary	24
9	<i>NL</i> , natürliche Sprache	29
10	Lineare Index Grammatik	38
11	Namen für Knoten	56
12	Projektionslinie, Projektionskante	57
13	Phrase, projizieren	58
14	Dominanz	59
15	c-Kommando	59
16	m-Kommando	59
17	Nachbarschaftskante	60
18	Kompakte Notation von Kantentypen	60
19	Pfad, <i>path</i> , Pfadstring	61
20	<i>num_{preorder}</i>	61
21	Ordnung auf die Blätter eines Syntaxbaums	62
22	<i>yield</i>	62
23	l-depth, r-depth	63
24	Verzweigungstiefe	63
25	Alphabete im Syntaxbaum, Lexikon	66
26	Relation $<_{max}$	68
27	Knotenname	68

28	<i>label</i> , Bezeichner, Index, barlevel	68
29	Lexikalische Kategorien	70
30	Funktionale Kategorien	70
31	Kategorien	70
32	Typen	70
33	Typ einer Phrase	71
34	koreferent	72
35	koindiziert	72
36	Kette, Antezedens, Referent, Basis	73
37	Barriere	75
38	Subjazenzenz	75
39	regieren, Rektion, Regens	75
40	lexikalische Rektion	75
41	Antezedens Rektion	75
42	Strenge Rektion	75
43	α -Kette	75
44	Untersyntaxbaum	76
45	Bindung, binden, gebunden, frei	77
46	β -Kette	77
47	R-Ausdruck	78
48	Complement, Specifier, Adjunkt	81
49	notwendige Nachbarschaftsbeziehungen	82
50	optionale Nachbarschaftsbeziehungen	85
51	Referenzkante, Indexkante	99
52	Kette	99
53	GB-kompatibel	101
54	Syntaxgraph, \mathcal{SG}	101
55	Gerüst	101
56	Kantenseparator	108

57	(r, c) - $f(n)$ -Kantenseparator	108
58	starker Kantenseparator	108
59	$n \times m$ -Quasigitter	109
60	Baumzerlegung	114
61	Baumbreite, Treewidth	114
62	Knotenseparator	114
63	GG_{deriv*}	146
64	GG_{chain}	154
65	Right Adjoining Grammar (RAG)	156
66	Produktionsanwendung	157
67	Sprache einer RAG , \mathcal{RAL}	157
68	Chomsky-Normalform einer RAG	158

Verzeichnis der Prinzipien von GB

1	Gerüst eines Syntaxbaums, Minimalanforderungen	53
2	Kantenbeschriftung und Ordnung	60
3	Namen im Syntaxbaum	68
4	Restriktionen auf koindizierte Knoten	78
5	Nachbarschaftsprinzip	86

Abbildungsverzeichnis

2.1	Zwei Produktionen für vollständige Graphen	21
2.2	Ableitungsschritte für den K_4	23
2.3	Produktionen für <i>extended binary trees</i>	23
2.4	Ein 5×5 Gittergraph	24
3.1	Linguistische Disziplinen betrachten verschiedene Aspekte von Sprache . .	29
3.2	Ein einfacher Syntaxbaum für „ <i>John loves Mary</i> “	32
3.3	Manchmal sind mehrere Satzanalysen möglich	32
3.4	kreuzweise und verschachtelte Abhängigkeit	35
3.5	Lineare Index Grammatiken	38
3.6	Ein Modell approximiert eine unscharfe Menge	39
3.7	Eine parametrisierte Universalgrammatik beschreibt alle Sprachen	42
3.8	Eine UG erlaubt generische Ansätze für linguistische Algorithmen	43
3.9	Approximation von „ <i>innen</i> “ und von „ <i>außen</i> “	46
3.10	Beziehungen wichtiger Schlüsselbegriffe in Informatik und Linguistik . . .	47
4.1	Ziel: GB in eine graphentheoretische Spezifikation übersetzen	52
4.2	edNCE-Graphgrammatik für Gerüste von Syntaxbäumen	54
4.3	Ein Baum aus $L(GG_{X-Bar})$	56
4.4	Syntaxbäume zerfallen in Projektions- und Nachbarschaftskanten	58
4.5	Syntaxbaumerüst mit Kantenbeschriftungen	62
4.6	Ordnung auf Knoten gemäß $num_{preorder}$	63
4.7	Knotenpositionen im Baum beschränken erlaubte Namensgebung	69
4.8	Zusammenhang der Kettentypen und mögliche Erweiterungen	79

4.9	Eine Instanz des X-Bar Schemas	82
4.10	Notwendige Nachbarschaftsbeziehungen	83
4.11	Ein minimaler Syntaxbaum	84
4.12	Minimaler Syntaxbaum in GB: Viele Abzweigungen bleiben ungenutzt. . .	84
4.13	Optionale Nachbarschaftsbeziehungen	85
4.14	Nominalphrase für „ <i>Marys house in London</i> “	86
4.15	Syntaxbaum: „ <i>John sings.</i> “	87
4.16	Syntaxbaum: „ <i>John loves Mary.</i> “	88
4.17	Syntaxbaum: „ <i>Peter believes that John loves Mary.</i> “	88
4.18	Syntaxbaum: „ <i>Peter_α believes that he_α loves Mary.</i> “	89
4.19	Syntaxbaum: „ <i>Peter_α believes that he_α loves himself_α.</i> “	89
4.20	Syntaxbaum: „ <i>(That John loves Mary)_i, Peter believes trace_i.</i> “	90
4.21	Syntaxbaum: „ <i>(That he_α loves himself_α)_i, Peter_α believes trace_i.</i> “	90
4.22	Syntaxbaum: „ <i>(That race)_i, I think trace_i that John won trace_i.</i> “	91
4.23	Eine versuchte Analyse für *„ <i>John, Peter believes, that loves Mary.</i> “ . . .	92
4.24	Versuchte Analyse: *„ <i>John_α expects, that Mary defends himself_α.</i> “	93
4.25	Prinzipien für Graphen beschreiben Zeichenketten.	94
5.1	Eine vormalig schmale und hohe Struktur wird flach und breit	100
5.2	Zu Beispiel 21: Ein Syntaxbaum und der entsprechende Syntaxgraph . . .	102
5.3	Zu Beispiel 22: Ein Syntaxbaum und der entsprechende Syntaxgraph. . .	103
5.4	Querkanten lassen keine Zyklen entstehen	104
5.5	Syntaxgraphen sind nicht planar	107
5.6	Der Graph aus Abbildung 5.5 anders dargestellt	108
5.7	Aufbau von einem $n \times m$ -Quasigitter	109
5.8	Konstruktion eines starken Separators	110
5.9	‘diagonal’ benachbarte Knoten werden verschmolzen	111
5.10	Verschmolzene Knoten werden abwechselnd verteilt	111
5.11	$n \times m$ -Quasigitter lassen sich als verdichtete Syntaxgraphen ansehen . . .	113

5.12	Die Graphgrammatik GG_{deriv} approximiert Teilmengen von NL	127
5.13	Erweiterungsmöglichkeiten von GG_{deriv}	143
5.14	Erweiterte Produktion können α - und β -Ketten gemeinsam generieren . .	143
5.15	Produktionen für englische (links) und deutsche (rechts) Wortstellung . .	145
5.16	Produktion, welche die schweizerdeutsche Wortstellung beschreibt	145
5.17	In drei Ableitungsschritte zum <i>yield</i> $a_1a_2a_3 \dots b_1b_2b_3$	146
6.1	Eine Wortgrammatik faßt zwei Ableitungsphasen zusammen.	150
6.2	Gesucht: Graphgrammatiken für Kettengraphen	151
6.3	Produktionen einer LIG modellieren eine RAG	159
6.4	Mit dem Keller einer LIG läßt sich die Adjunktion von RAG s simulieren .	160
6.5	Die Produktionen der LIG werden systematisch aus der RAG abgeleitet .	161
6.6	Algorithmus CYK^+ zur Lösung des Wortproblems für RAG s	162
6.7	Tabelle zum Parsingvorgang	163
6.8	Ableitungsbaum eine RAG zu <i>abaaba</i>	164
6.9	Wortgrammatik: zwei Phasen werden zusammengefaßt	169

Literaturverzeichnis

- [And92] E. S. Andersen, *Complexity and Language Acquisition: Influences on the Development of Morphological Systems in Children*, J. A. Hawkins, M. Gell-Mann, Hrsg., The Evolution of Human Languages. Proceedings of the workshop on the evolution of human languages, 241–271, Addison-Wesley, The Advanced Book Program, Redwood City, California, 1992.
- [Aou92] J. Aoun, *A Brief Presentation of the Generative Enterprise*, J. A. Hawkins, M. Gell-Mann, Hrsg., The Evolution of Human Languages. Proceedings of the workshop on the evolution of human languages, Addison-Wesley, The Advanced Book Program, Redwood City, California, 1992.
- [BBR87] G. E. Barton, R. C. Berwick, E. S. Ristad, *Computational Complexity and Natural Language*, A Bradford Book, The MIT Press, 1987.
- [BCR91] L. Breveglieri, C. Citrini, S. C. Reghizzi, *Deterministic Dequeue Automata and $LL(1)$ Parsing of Breadth-Depth Grammars*, Budach, Hrsg., Fundamentals of Computation Theory, Band 529, Lecture Notes in Computer Science, 146–156, 1991.
- [BCR95] L. Breveglieri, A. Cherubini, S. C. Reghizzi, *Deterministic Parsing for Augmented Context-free Grammars*, J. Wiedermann, P. Hajek, Hrsg., Mathematical Foundations of Computer Science, Band 969, Lecture Notes in Computer Science, 326–336, 1995.
- [BH64] Bar-Hillel, *Finite State Languages: Formal Representations and Adequacy Problems*, Bar-Hillel, Hrsg., Language and Information, 87–98, Addison-Wesley, 1964.
- [Blo33] L. Bloomfield, *Language*, Holt, Rinehart & Winston, New York, 1933.
- [Bod92] H. L. Bodlaender, *A Tourist Guide through Treewidth*, Tech. rep., Utrecht University, Department of Computer Science, 1992.
- [Boo80] R. V. Book, Hrsg., *Formal Language Theory. Perspectives and Open Problems*, Academic Press, 1980.
- [Bör95] F. Börncke, *Entwurf und Analyse eines Grammatikformalismus für natürliche Sprachen mit Hilfe von Graph-Grammatiken*, Diplomarbeit, Fakultät für Mathematik und Informatik – Universität Passau, 1995.

- [Bou99] P. Boullier, *A Cubic Time Extension of Context-Free Grammars*, 1999, Manuskript zum gleichnamigen Vortrag auf der MOL 6 - Sixth Meeting on Mathematics of Language.
- [BP92] E. Barber, A. Peters, *Ontogeny and Phylogeny: What Child Language and Archaeology Have to Say to Each Other*, J. A. Hawkins, M. Gell-Mann, Hrsg., The Evolution of Human Languages. Proceedings of the workshop on the evolution of human languages, 305–352, Addison-Wesley, The Advanced Book Program, Redwood City, California, 1992.
- [BR98] P. Blackburn, M. de Rijke, Hrsg., *Specifying Syntactic Structures*, Studies in Logic, Language and Information, CSLI Publ., 1998.
- [Bra80] F. J. Brandenburg, *Multiple Equality Sets and Post Machines*, Journal of Computer and System Sciences, Band 21, 292–316, 1980.
- [Bra81] F. J. Brandenburg, *Analogies of PAL and COPY*, F. Gecseg, Hrsg., Fundamentals of Computation Theory, Band 117, Lecture Notes in Computer Science, 61–70, 1981.
- [Bra86] F. J. Brandenburg, *Intersections of some Families of Languages*, Automata, Languages and programming, Band 226, Lecture Notes in Computer Science, 60–68, 1986.
- [Bra87] F. J. Brandenburg, *On Partially Ordered Graph Grammars*, STACS'87, Band 291, Lecture Notes in Computer Science, 99–111, Springer Verlag, Berlin, 1987.
- [Bra88a] F. J. Brandenburg, *On Polynomial Time Graph Grammars*, STACS'88, Band 294, Lecture Notes in Computer Science, 227–236, Springer Verlag, Berlin, 1988.
- [Bra88b] F. J. Brandenburg, *On the Intersection of Stacks and Queues*, Theoretical Computer Science, Band 58, 69–80, 1988.
- [Bra91] F. J. Brandenburg, *The Equivalence of Boundary and Confluent Graph Grammars on Graph Languages of Bounded Degree*, R. V. Book, Hrsg., Rewriting Techniques and Applications, Band 488, Lecture Notes in Computer Science, 312–322, Springer Verlag, Berlin, 1991.
- [BRC99] L. Breveglieri, S. C. Reghizzi, A. Cherubini, *Modeling Operating Systems Schedulers with Multi-Stack-Queue Grammars*, G. Ciobanu, G. Paun, Hrsg., Fundamentals of Computation Theory, Band 1684, Lecture Notes in Computer Science, 161–172, 1999.
- [Bus92] S. Busemann, *Generierung natürlicher Sprache mit generalisierten Phrasenstruktur-Grammatiken*, Springer, 1992.
- [CCRM90] A. Cherubini, C. Citrini, S. C. Reghizzi, D. Mandrioli, *Breadth and Depth Grammars and Dequeue Automata*, International Journal of Foundations of Computer Science, Band 1, Nr. 3, 219–232, 1990.

- [CCRM91] A. Cherubini, C. Citrini, S. C. Reghizzi, D. Mandrioli, *QRT FIFO automata, breadth-first grammars and their relations*, Theoretical Computer Science, Band 85, 171–203, 1991.
- [CER93] B. Courcelle, J. Engelfriet, G. Rozenberg, *Handle-rewriting Hypergraph Grammars*, Journal of Computer and System Sciences, Nr. 46, 218–270, 1993.
- [Cho55] N. Chomsky, *The logical structure of linguistic theory*, Manuscript MIT, 1955.
- [Cho56] N. Chomsky, *Three models for the Description of Language*, Transactions on Information Theory, Nr. 2:3, 113–124, 1956.
- [Cho57] N. Chomsky, *Syntactic Structures*, Mouton, Den Haag, 1957.
- [Cho59] N. Chomsky, *On certain formal properties of grammars*, Information and Control, Nr. 2:2, 137–167, 1959.
- [Cho61] N. Chomsky, *Some methodological remarks on generative grammar*, Word, Nr. 17, 219–239, 1961.
- [Cho65] N. Chomsky, *Aspects of the theory of syntax*, MIT Press, 1965, dt. Übersetzung: *Aspekte der Syntax-Theorie*, Frankfurt/Main, 1969.
- [Cho81] N. Chomsky, *Lectures on Government and Binding*, Foris, Dordrecht, 1981.
- [Cho86a] N. Chomsky, *Barriers*, MIT Press, Cambridge, Mass. u.a., 1986.
- [Cho86b] N. Chomsky, *Knowledge of Language. Its nature, origin, and use*, Praeger, 1986.
- [Cho93] N. Chomsky, *A Minimalist Program for Linguistic Theory*, K. Hale, S. J. K. (edts), Hrsg., *The view from Building 20 - Essays in Linguistics in Honor of Sylvain Bromberger*, MIT Press, Cambridge, Massachusetts, London, 1993.
- [Cho94] N. Chomsky, *Bare Phrase Structure*, MIT Occasional Papers in Linguistic 5, MIT, 1994.
- [Com92] B. Comrie, *Before Complexity*, J. A. Hawkins, M. Gell-Mann, Hrsg., *The Evolution of Human Languages. Proceedings of the workshop on the evolution of human languages*, 193–211, Addison-Wesley, The Advanced Book Program, Redwood City, California, 1992.
- [Cou87] B. Courcelle, *An axiomatic definition of context-free rewriting and its applications*, Theoretical Computer Science, Band 55, 141–181, 1987.
- [Cry92] D. Crystal, *The Cambridge Encyclopedia of Language*, Cambridge University Press, 1987-1992.
- [Cul97] P. W. Culicover, *Principles and Parameters. An Introduction to Syntactic Theory*, Oxford University Press, 1997.

- [Dea] T. W. Deacon, *Brain-Language Coevolution*, J. A. Hawkins, M. Gell-Mann, Hrsg., The Evolution of Human Languages. Proceedings of the workshop on the evolution of human languages, 49–83.
- [Dre93] F. Drewes, *Recognising k -connected hypergraphs in cubic time*, Theoretical Computer Science, Band 109, 83–122, 1993.
- [Dre96] F. Drewes, *Computation by Tree Transductions*, Dissertation, Fachbereich Mathematik und Informatik - Universität Bremen, 1996.
- [EL89] J. Engelfriet, G. Leih, *Linear Graph Grammars: Power and Complexity*, Information and Computation, Nr. 81, 88–121, 1989.
- [EL90] J. Engelfriet, G. Leih, *Complexity of boundary graph grammars*, RAIRO Theoretical Informatics, Nr. 24, 267–274, 1990.
- [ELW88] J. Engelfriet, G. Leih, E. Welzl, *Boundary Graph Grammars with Dynamic Edge Relabeling*, ACM transactions on programming languages and systems, Band 10, 312–339, 1988.
- [Eng89] J. Engelfriet, *Context-free NCE graph grammars*, Proceedings: Fundamentals of Computation Theory, Band 380, Lecture Notes in Computer Science, 148–161, 1989.
- [Eng97] J. Engelfriet, *Context-Free Graph Grammars*, G. Rozenberg, A. Salomaa, Hrsg., Handbook of Formal Languages, Band 3, 125–213, Springer, 1997.
- [ER90] J. Engelfriet, G. Rozenberg, *A comparison of boundary graph grammars and context-free hypergraph grammars*, Information and Computation, Nr. 84, 163–206, 1990.
- [Fal96] Y. N. Falk, *Case Typology and Case Theory*, <http://pluto.msc.huji.ac.il/~msyfalk/case.html>, 1996.
- [FF87a] S. Felix, G. Fanselow, *Sprachtheorie 1. Grundlagen und Zielsetzungen*, UTB Francke, 1987.
- [FF87b] S. Felix, G. Fanselow, *Sprachtheorie 2. Die Rektions- und Bindungstheorie*, UTB Francke, 1987.
- [FFO92] S. Felix, H. Farke, J. Olsen, *The Format Problem in a GB-Parser*, Passauer Arbeitspapiere zur Linguistik und Informatik Nr. 3, Universität Passau, 1992.
- [Gaz88] G. Gazdar, *Applicability of indexed grammars to natural languages*, U. Reyle, C. Rohrer, Hrsg., Natural Language Parsing and Linguistic Theories, 69–94, Reidel, 1988.
- [GHS87] G. Grewendorf, F. Hamm, W. Sternefeld, *Sprachliches Wissen – Eine Einführung in moderne Theorien der grammatischen Beschreibung*, Suhrkamp, 1987.

- [GJ79] M. R. Garey, D. S. Johnson, *Computers and Intractability. A guide to the theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, 1979.
- [GM92] M. Gell-Mann, *Complexity and Complex Adaptive Systems*, J. A. Hawkins, M. Gell-Mann, Hrsg., The Evolution of Human Languages. Proceedings of the workshop on the evolution of human languages, Addison-Wesley, The Advanced Book Program, Redwood City, California, 1992.
- [GP87] G. Gazdar, G. K. Pullum, *Computationally Relevant Properties of Natural Languages and their Grammars*, W. J. Savitch, E. Bach, W. Marsh, G. Safran-Naveh, Hrsg., The formal complexity of natural language, Studies in linguistics and philosophy, 387–437, Reidel, 1987.
- [GPWS96] L. Guthrie, J. Pustejovsky, Y. Wilks, B. M. Sator, *The Role of Lexicons in Natural Language Processing*, Band 39, Nr. 1, Communications of the ACM, 1996.
- [Gre92] J. H. Greenberg, *Preliminaries to a Systematic Comparison between Biological and Linguistic Evolution*, J. A. Hawkins, M. Gell-Mann, Hrsg., The Evolution of Human Languages. Proceedings of the workshop on the evolution of human languages, 139–158, Addison-Wesley, The Advanced Book Program, Redwood City, California, 1992.
- [Hab92] A. Habel, *Hyperedge Replacement: Grammars and Languages*, Springer Verlag, 1992.
- [Har78] M. A. Harrison, *Introduction to formal language theory*, Addison-Wesley, Addison-Wesley, Reading, Mass. u.a., 1978.
- [Hat87] J. N. Hattiangadi, *How is language possible? philos. reflections on the evolution of language and knowledge*, Open Court, La Salle, Ill., 1987.
- [Hau99] R. Hausser, *Foundations of Computational Linguistics. Man-Machine Communication in Natural Language*, Springer, 1999.
- [Haw92] J. A. Hawkins, *Innateness and Function in Language Universals*, J. A. Hawkins, M. Gell-Mann, Hrsg., The Evolution of Human Languages. Proceedings of the workshop on the evolution of human languages, Addison-Wesley, The Advanced Book Program, Redwood City, California, 1992.
- [Hig87] J. Higginbotham, *English is not a context-free language*, W. J. Savitch, E. Bach, W. Marsh, G. Safran-Naveh, Hrsg., The formal complexity of natural language, Studies in linguistics and philosophy, 335–368, Reidel, 1987.
- [HK87] A. Habel, H.-J. Kreowski, *Some Structural Aspects of Hypergraph Languages Generated by Hyperedge Replacement*, STACS'87, Band 291, Lecture Notes in Computer Science, 207–219, Springer Verlag, Berlin, 1987.
- [HSW80] H.-J. Heringer, B. Strecker, R. Wimmer, *Syntax. Fragen, Lösungen, Alternativen*, Wilhelm Fink Verlag, München, 1980.

- [HU79] J. E. Hopcroft, J. D. Ullman, *Introduction to automata theory, languages, and computation*, Addison-Wesley, 1979.
- [Hur92] J. R. Hurford, *An Approach to the Phylogeny of the Language Faculty*, J. A. Hawkins, M. Gell-Mann, Hrsg., The Evolution of Human Languages. Proceedings of the workshop on the evolution of human languages, 273–303, Addison-Wesley, The Advanced Book Program, Redwood City, California, 1992.
- [JJ82] D. F. Jonas, A. D. Jonas, *Das erste Wort - Wie die Menschen sprechen lernten*, Ullstein Sachbuch, 1982.
- [JLT75] A. K. Joshi, L. Levy, M. Takahashi, *Tree Adjunct Grammars*, Journal of Computer and System Sciences, 136–163, 1975.
- [Jos85] Joshi, *Tree Adjoining Grammars: How much Context-Sensitivity is required to provide reasonable structural descriptions?*, D. R. Dowty, L. Kartunen, A. M. Zwicky, Hrsg., Natural language parsing, 206–230, Cambridge Univ. Press, 1985.
- [Jos87] Joshi, *The relevance of Tree Adjoining Grammars to generation*, G. Kempen, Hrsg., Natural language generation. New results in artificial intelligence, psychology and linguistics, Kapitel 16, 233–252, Nijhoff, 1987.
- [JR80a] D. Janssens, G. Rozenberg, *On the structure of node label controlled graph languages*, Information Science, 191–216, 1980.
- [JR80b] D. Janssens, G. Rozenberg, *Restrictions, extensions, and variations of NLC grammars*, Information Science, Band 20, 217–244, 1980.
- [JR82] D. Janssens, G. Rozenberg, *Graph grammars with neighbourhood-controlled embedding*, Theoretical Computer Science, Band 21, 55–74, 1982.
- [JRR90] K. Jablonski, A. Rau, J. Ritzke, *Wissensbasierte Textgenerierung. Linguistische Grundlagen und softwaretechnische Realisierung*, Gunter Narr Verlag Tübingen, 1990.
- [Kay94] R. S. Kayne, *The antisymmetry of syntax*, MIT Press. u.a., 1994.
- [Kha97] S. Khalaily, *One Syntax for All Categories - Merging Nominal Atoms in Multiple Adjunction Structures*, Holland Academic Graphics, 1997.
- [Kin71] R. D. King, *Historische Linguistik und generative Grammatik*, Athenäum Verlag GmbH, Frankfurt/M., 1971.
- [Kin83] M. King, Hrsg., *Parsing Natural Language*, Acad. Press, London u.a., 1983.
- [Kin96] M. King, *Evaluating Natural Language Processing Systems*, Band 39, Nr. 1, Communications of the ACM, 1996.
- [KJ87] A. S. Kroch, Joshi, *Analyzing Extraposition in a TAG*, G. J. Huck, Hrsg., Discontinuous constituency, 20, 107–149, Acad. Press, 1987.

- [KL96] J. Kowie, W. Lehnert, *Information Extraction*, Band 39, Nr. 1, Communications of the ACM, 1996.
- [Klo93] T. Kloks, *Treewidth*, Universiteit Utrecht, 1993.
- [Knu73] Knuth, *Fundamental algorithms. The art of computer programming*, Band 1, Addison-Wesley, 1973, 2. Auflage.
- [Koe82] W. Koenig, *Probleme der Repräsentativität in der Dialektologie*, W. Besch, U. Knoop, W. Putschke, H. E. Wiegand, Hrsg., Dialektologie. Ein Handbuch zur deutschen und allgemeinen Dialektforschung, Band 1, Handbücher zur Sprach- und Kommunikationswissenschaft, 463–485, de Gruyter, 1982.
- [Kra97] M. Kracht, *On reducing principles to rules*, P. Blackburn, M. de Rijke, Hrsg., Specifying Syntactic Structures, Studies in Logic, Language and Information, Kapitel 3, 45–73, CSLI Publ., 1997.
- [Kra98] M. Kracht, *Advances in modal logic*, Band 1, CSLI lecture series, CSLI Publ., Stanford, Calif., 1998.
- [Kra99] M. Kracht, *Tools and techniques in modal logic*, Band 142, Studies in logic and the foundations of mathematics, Elsevier, 1999.
- [Kro89] A. S. Kroch, *Asymmetries in Long-Distance Extraction in a TAG*, M. R. Baltin, Hrsg., Alternative conceptions of phrase structure, Kapitel 4, 66–98, University of Chicago Press, 1989.
- [Kut74] F. von Kutschera, *Sprachphilosophie*, Wilhelm Fink Verlag, 1974, 2. Auflage.
- [Lan88] M. E. Landsberg, Hrsg., *The Genesis of Language. A different judgement of Evidence*, Mouton de Gruyter, Berlin, New York, Amsterdam, 1988.
- [Lie92] P. Lieberman, *On the evolution of human language*, J. A. Hawkins, M. Gell-Mann, Hrsg., The Evolution of Human Languages. Proceedings of the workshop on the evolution of human languages, Addison-Wesley, The Advanced Book Program, Redwood City, California, 1992.
- [LJ96] D. D. Lewis, K. S. Jones, *Natural Language Processing for Information Retrieval*, Band 39, Nr. 1, 92–98, Communications of the ACM, 1996.
- [LT79] R. J. Lipton, R. E. Tarjan, *A Separator Theorem for Planar Graphs*, SIAM Journal of Applied Mathematics, Band 36, 177–189, 1979.
- [LT80] R. J. Lipton, R. E. Tarjan, *Applications of a Planar Separator Theorem*, SIAM Journal on Computing, Band 9, 615–627, 1980.
- [Mäk90] E. Mäkinen, *On Breadth-first Context-free Grammars*, Journal Information Processing Cybernetics, Band 3, 129–135, 1990.
- [Mar60] A. Martinet, *Éléments de linguistique générale*, Paris, 1960, in deutscher Übersetzung von Anna Fuchs: Grundzüge der allgemeinen Sprachwissenschaft, Stuttgart, fünfte Auflage, 1971.

- [MB91] T. Mason, D. Brown, *LEX and YACC. UNIX programming tools - A nutshell handbook*, O' Reilly, 1991.
- [Mei88] G. Meier, *Im Anfang war das Wort. Die Spracharchäologie als neue Disziplin der Geisteswissenschaften*, Verlag Paul Haupt Bern und Stuttgart, 1988.
- [MR90] M. G. Main, G. Rozenberg, *Edge-Label Controlled Graph Grammars*, Journal of Computer and system sciences, Band 40, 188–228, 1990.
- [MS95] A. Mateescu, A. Salomaa, *Views on Linguistics*, ECS - Jahresbericht 1995, 1995.
- [MS97] A. Mateescu, A. Salomaa, *Formal Languages: an Introduction and a Synopsis*, G. Rozenberg, A. Salomaa, Hrsg., Handbook of Formal Languages, Band 1, 1–39, Springer, 1997.
- [Nag79] M. Nagl, *Graph-Grammatiken, Theorie und Implementierung*, Vieweg, 1979.
- [Pel84] H. Pelz, *Linguistik für Anfänger*, Hoffman und Campe, 1984.
- [PG87] G. K. Pullum, G. Gazdar, *Context-Sensitive Grammar And Natural Language Syntax*, W. J. Savitch, E. Bach, W. Marsh, G. Safran-Naveh, Hrsg., The formal complexity of natural language, Studies in linguistics and philosophy, 138–182, Reidel, 1987.
- [Pin94] S. Pinker, *The language instinct*, William Morrow and Company, 1994.
- [Pin97] S. Pinker, *How the mind works*, W. W. Norton & Company, 1997.
- [PMW90] B. H. Partee, A. G. ter Meulen, R. E. Wall, *Mathematical Methods in Linguistics*, Kluwer Academic Publishers, Dordrecht u.a., 1990, student Edition.
- [Pol84] C. Pollard, *Generalized Phrase Structure Grammars, Head Grammars, and Natural Language*, Dissertation, Stanford University, CA, 1984.
- [Pol89] J. Y. Pollock, *Verb Movement, Universal Grammar and the Structure of IP*, Linguistic Inquiry, Band 20, 365–424, 1989.
- [Rog97] J. Rogers, *On Descriptive Complexity, Language Complexity, and GB*, P. Blackburn, M. de Rijke, Hrsg., Specifying Syntactic Structures, Studies in Logic, Language and Information, 157–183, CSLI Publ., 1997.
- [Rog98] J. Rogers, *A Descriptive Approach to Language-Theoretic Complexity*, Studies in Logic, Language and Information, CSLI Publ., 1998.
- [Rom92] S. Romaine, *The Evolution of Linguistic Complexity in Pidgin and Creole Languages*, J. A. Hawkins, M. Gell-Mann, Hrsg., The Evolution of Human Languages. Proceedings of the workshop on the evolution of human languages, 213–238, Addison-Wesley, The Advanced Book Program, Redwood City, California, 1992.

- [Roz97] G. Rozenberg, *Handbook of Graph Grammars and Computing by Graph Transformation*, Band 1, , World Scientific, 1997.
- [RS83] N. Robertson, P. Seymour, *Graph minors. I. Excluding a forest*, Journal Comb. Theory Series B, Band 35, 39–61, 1983.
- [RS86] N. Robertson, P. Seymour, *Graph minors. II. Algorithmic aspects of tree-width*, Journal of Algorithms, Band 7, 309–322, 1986.
- [RS97] G. Rozenberg, A. Salomaa, Hrsg., *Handbook of formal languages*, Bände 1-3, Springer Verlag, Berlin, 1997.
- [Ruh92] M. Ruhlen, *An overview of genetic Classification*, J. A. Hawkins, M. Gell-Mann, Hrsg., The Evolution of Human Languages. Proceedings of the workshop on the evolution of human languages, 159–189, Addison-Wesley, The Advanced Book Program, Redwood City, California, 1992.
- [RW86] G. Rozenberg, E. Welzl, *Boundary NLC Graph Grammars – Basic Definitions, Normal Forms, and Complexity*, Information and Control, Band 69, 136–167, 1986.
- [Sal73] A. Salomaa, *Formal Languages*, ACM monograph series, Academic Press, 1973.
- [Sal81] A. Salomaa, *Jewels of formal language theory*, Pitman, London, 1981.
- [Sav87] W. J. Savitch, *Context-Sensitive Grammar And Natural Language Syntax*, W. J. Savitch, E. Bach, W. Marsh, G. Safran-Naveh, Hrsg., The formal complexity of natural language, Studies in linguistics and philosophy, 358–368, Reidel, 1987.
- [Sch87] R. Schuster, *Graphgrammatiken und Grapheinbettungen: Algorithmen und Komplexität. Dissertation*, Dissertation, Universität Passau, Passau, 1987.
- [Sch93] K.-M. Schneider, *Entwurf und Implementierung eines GB-Parsers für ein Fragment des Deutschen*, Diplomarbeit, Fakultät für Mathematik und Informatik – Universität Passau, 1993.
- [Shi88] S. M. Shieber, *Evidence against the context-freeness of Natural Language*, J. Kulas, Hrsg., Philosophy, language, and artificial intelligence. Resources for processing natural language, 79–89, Kluwer, 1988.
- [Sko92] K. Skodinis, *Pumping Lemma für BCF-Sprachen*, Manuskript - Universität Passau, 1992.
- [Sko95] K. Skodinis, *Komplexität bei Graph Grammatiken und Graph Automaten*, Dissertation, Fakultät für Mathematik und Informatik - Universität Passau, 1995.
- [Spe90] M. Speas, *Phrase structure in natural language*, Studies in natural language and linguistic theory, Kluwer Acad. Publ., Dordrecht u.a., 1990.

- [Spr92] R. Sproat, *Morphology and computation*, ACL-MIT Press series in natural-language processing, MIT Press, 1992.
- [Tom86] M. Tomita, *Efficient parsing for natural language - a fast algorithm for practical systems*, Kluwer, Boston u.a., 1986.
- [Ull84] J. D. Ullman, *Computational aspects of VLSI*, Principles of computer science series, Computer Science Press, 1984.
- [Vog90] W. Vogler, *Recognizing Edge Replacement Graph Languages in Cubic Time*, G. R. H. Ehrig, H.-J. Kreowski, Hrsg., Graph Grammars and Their Application to Computer Science, Band 532, Lecture Notes in Computer Science, 676–687, Springer Verlag, 1990.
- [VSW94] K. Vijay-Shanker, D. Weir, *The equivalence of Four Extensions of Context-Free Grammars*, Mathematical Systems Theory 27, Band 27, 511–546, 1994.
- [Win83] T. Winograd, *Language as a cognitive process*, Band 1: Syntax, , Addison Wesley, Reading, Mass. u.a., 1983.
- [Woo87] D. Wood, *Theory of Computation*, Harper & Row, New York, 1987.
- [XTA95] XTAG, *A Lexicalized Tree Adjoining Grammar for English*, Tech. rep., Institute for Research in Cognitive Science - University of Pennsylvania, 1995.
- [YJ83] Yokomori, Joshi, *Semilinearity, Parikh-Boundedness and Tree Adjunct Languages*, Information processing letters, Nr. 17, 137–143, 1983.
- [Zam96] R. Zamparelli, *Layers in the DP*, <http://www.cogsci.ed.ac.uk/~roberto/layers/intro.html>, 1996.