Modellbasierte und Statistische Lernverfahren zur Erweiterung von Unifikationsgrammatiken

Diplomarbeit im Fach Informatik
vorgelegt
von

Philipp Köhn
geb. am 1. August 1971 in Erlangen

Angefertigt am
Institut für Mathematische Maschinen und Datenverarbeitung
Lehrstuhl für Künstliche Intelligenz
Friedrich-Alexander-Universität Erlangen-Nürnberg

Betreuer: Prof. Dr. Günter Görz
Dr. Hans Weber

Beginn der Arbeit: 1. Mai 1996
Abgabe der Arbeit: 4. November 1996

Ich versichere, daß ich diese Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe, und daß die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, den 4. November 1996

(Philipp Köhn)

Widmung

Diese Arbeit ist all denen gewidmet, die mich mit ihrer Liebe, ihrer Freundschaft und ihrem Geld davon abgehalten haben, sie zu schreiben. Ohne deren Hilfe wäre sie nicht zu dem geworden, was sie ist.

Zusammenfassung

In Systemen zur Verarbeitung natürlicher Sprache wird die Syntax oft mittels Grammatiken modelliert. Das Schreiben dieser Grammatiken ist jedoch sehr zeitaufwendig. Es scheint sogar unmöglich zu sein, jemals eine Grammatik zu erstellen, die vollständig eine natürliche Sprache abdeckt. Es gibt immer wieder unbeachtete Sonderfälle und Ausnahmen, so daß nach jedem Testen neue Regelen der Grammatik hinzugefügt werden müssen.

Miles Osborne [Osb94] führte ein Verfahren für das Englische ein, in dem zu einer gegebenen Grundgrammatik zusätzliche Regeln gelernt werden. Die Regelkonstruktion benutzt dabei Charts von nicht erfolgreich geparsten Trainingssätzen. Das Verfahren besteht aus einer linguistisch motivierten modellbasierten Komponente und einer statistisch angelegten datengesteuerten Komponente, um neue Regeln zu konstruieren und zu bewerten.

Zielsetzung dieser Arbeit war es, dieses Verfahren für das Deutsche anwendbar zu machen. Da sich die deutsche Sprache erheblich vom Englischen unterscheidet, konnte insbesondere die modellbasierte Komponente so nicht übernommen werden. Es wurde ein neuer Grammatikformalismus und ein neues Grammatikmodell entwickelt. Auch wurde die Regelgenerierung genauer beschränkt, so daß die linguistische Plausibilität der Grammatik in höherem Maße erhalten bleibt. Das Verfahren wurde in Prolog, C und Perl implementiert.

In den durchgeführten Experimenten konnte gezeigt werden, daß mit dem Verfahren erfolgreich Regeln gelernt werden konnten, die erlaubten, einen Großteil der Testsätze zu parsen.

Danksagung

Vielen Dank an Hans Weber für das interessante Thema und die Freiheit, die er mir beim Schreiben der Arbeit gelassen hat. Dank an die Geduld an das Verständnis aller am Lehrstuhl 8 für exzessive Benutzung ihrer Rechnerresourcen. Vielen lieben Dank auch an Michael Sträubig für das Korrekturlesen.

Philipp Köhn

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einführung

Ein philosophische Problem hat die Form: ,,Ich kenne mich nicht aus."

Ludwig Wittgenstein
Philosophische Untersuchungen, 123

Diese Einführung umreißt den philosophischen und wissenschaftlichen Rahmen dieser Arbeit. Es wird auf die Entwicklung der Forschung auf dem Gebiet der Sprache und der KI hingewiesen und die Bedeutung statistischer Verfahren für maschinelle Sprachverarbeitung herausgestellt. Im letzten Abschnitt wird ein Überblick auf die folgenden Kapitel gegeben.

1.1 Sprache

Sprache ist ein weites Feld. So ist sie ein Mittel zur Kommunikation, sie ermöglicht es uns Menschen, unsere Wünsche, Ziele und Absichten anderen Menschen mitteilen zu können. Diese Fähigkeit, komplexe Sachverhalte kommunizieren zu können, ist eine der größten Geistesleistungen des Menschen, sie macht einen wesentlichen Teil unserer Intelligenz aus.

Auch entwickeln wir einen Großteil unseres Wissens über die Welt in Sprache. Über Jahrhunderte hinweg wurden philosophische und wissenschaftliche Ansichten in Form von Schriften und Büchern verbreitet und erhalten. Es ist daher nicht verwunderlich, daß in der Philosophie die Behauptung aufkam, daß Denken im Kern sprachlich ist. So schreibt Ludwig Wittgenstein im Tractatus Logico-Philosphicus: ,,Die Grenzen meiner Sprache bedeuten die Grenzen meiner Welt" [Wit60]. Descartes argumentierte, daß unser Denken auf Sprache beruht und Tiere, die nicht sprechen können, kein Bewußtsein haben.

Neben dem aktiven Sprachgebrauch (sprechen, hören, lesen), reden wir ständig mit uns selbst. Dennet schreibt in seinem Buch ,,Consciousness Explained" [Den91], daß dieser innere Dialog uns erst über uns selbst klarwerden läßt. Erst wenn wir einen Gedanken in Worte fassen, können wir ihn wirklich verstehen. Unser Selbst ist nichts anderes als das Zentrum aller unserer Aussagen über uns (*Center of Narrative Gravity*).

Zwar erkennen wir heute zunehmend, daß die Rolle der Sprache wohl überschätzt und ein wesentlicher Teil unseres Denkens eben nicht sprachlich ist. Dennoch müssen wir zweifelsohne zugeben, daß Sprache wesentliche Aspekte unseres Denkens widerspiegelt. Sprache ist immer noch der wichtigste Schlüssel zu unserem Geist.

Kompositionalitätsprinzip

Ludwig Wittgenstein[1] schrieb Anfangs des Jahrhunderts eine der bedeutendsten philosophischen Arbeiten über Sprache: den „Tractatus logico-philosophicus" [Wit60]. Nach seiner Auffassung bildet Sprache die Welt ab. Die Struktur der Sprache stimmt mit der Struktur der Welt überein. Ziel jeder philosophischen Bemühung ist es somit, die Sprache von Verwirrungen zu befreien, damit der Blick auf die wahre Welt möglich ist.

Wörter bezeichnen Dinge in der Welt oder beschreiben, wie diese verknüpft werden können. Damit läßt sich die Bedeutung von Sätzen als Funktion seiner Wörter darstellen. Dies wird das *Frege'sche Kompositionalitätsprinzip* genannt: die Annahme, daß wir längere Texte dadurch verarbeiten können, indem wir sie immer mehr in ihre Bestandteile zerlegen.

Obwohl dies ist eine hilfreiche Annahme für begrenzte Sprachsituationen ist, versagt sie bei einer vollständigen Betrachtung der Sprache. So lassen sich Wörter nicht klar atomar definieren, sondern haben eine eher vage, vielschichtige Bedeutung. Wittgenstein distanzierte sich somit in seinen späteren „Philosophischen Untersuchungen" [Wit60] von seinen ursprünglichen Ansichten. Er erkennt, daß die Bedeutung eines Wortes entscheidend von dem jeweiligen *Sprachspiel*, von der jeweiligen Sprachsituation abhängt.

Ähnliche Widersprüche zum Kompositionalitätsprinzip rufen Kontexteffekte, Metaphern und Sarkasmus hervor, wo sich die wörtliche Bedeutung von der tatsächlichen unterscheidet. Lakoff und Johnson argumentieren in dem Buch „Methaphors we live by" [LJ80], daß natürliche Sprache von Grund auf metaphorisch organisiert ist.

Je mehr wir über menschliche Sprache lernen, desto mehr Schwierigkeiten erkennen wir. Sprache ist voller Ungenauigkeiten, Unregelmäßigkeiten und Ausnahmen. Es scheint keine Regel zu geben, die nicht auch gebrochen werden kann. Nur eines ist klar: Unser Gehirn verarbeitet Sprache anders als wir es uns bislang vorstellen können.

1.2 Künstliche Intelligenz

Die Geburtsstunde der Forschungsrichtung *Künstliche Intelligenz* (KI) als Teildisziplin der Informatik folgte kurz auf das Aufkommen der ersten elektronischen Digitalrechner. Ihr letztendliches Ziel ist nichts Geringeres als das Nachbilden und Übertreffen menschlicher Geistesleistungen.

Die Anfangszeit der KI war gekennzeichnet von heute kaum mehr nachvollziehbarer Euphorie. 1958 machte beispielsweise Herbert Simon folgende Vorhersagen für das Jahr 1968:

1. *A digital computer would be world chess champion, unless the rules barred it from competition.*
2. *A digital computer would discover and prove an important new mathematical theorem.*
3. *Most theories in psychology would take the form of computer programs, or of qualitative statements about the characteristics of computer programs.* [2]

[1] Ein sehr empfehlenswertes Buch über die Philosophie Ludwig Wittgensteins ist [Har62].

[2] Übersetzung: „1. Ein Digitalcomputer würde Schachweltmeister werden, wenn er nicht durch Regeln vom Wettbewerb ausgeschlossen würde.
2. Ein Digitalcomputer würde ein bedeutendes neues mathematisches Theorem entdecken und beweisen.
3. Die meisten psychologischen Theorien würden die Form von Computerprogrammen annehmen, oder qualitative Aussagen über die Eigenschaften von Computerprogrammen sein." [SN58], zitiert nach [Dre92, Seiten 81–82]

Heute, fast vierzig Jahre später, ist keines dieser Ziele erreicht. KI-Forschung hat vielerorts das Image einer Wissenschaft, die ihre Versprechen nicht einhalten kann. Teile der KI haben sich daher vom ursprünglichen Ziel verabschiedet und widmen sich der Entwicklung von sogenannten KI-Methoden und deren Anwendung in mehr kommerziellen Produkten.

Andererseits ist der Erfolg von Computern insgesamt umbestritten. Es gibt fast keinen Bereich der Gesellschaft, den die Informationsrevolution unberührt gelassen hat. Software erledigt Arbeitsvorgänge, die traditionell von Menschen ausgeführt wurden, und eröffnet neue Möglichkeiten. Der Computer zählt zweifelslos zu den wichtigsten Erfindungen der Moderne.

Diese herausragenden Leistungen der maschinellen Datenverarbeitung stoßen sehr schnell an Grenzen, wenn intelligenterer Umgang mit Informationen verlangt ist. Die Schwierigkeiten der KI haben klar aufgezeigt, daß sich menschliches Denken nicht einfach in die Regelsprache von Computerprogrammen übersetzen läßt. Die Popularität von neuronalen Netzen oder Fuzzy Logik ist ein Zeichen für die Suche nach alternativen Berechnungsparadigmen, die Unschärfe und Ungenaugigkeiten menschlichen Denkens besser fassen läßt.

In der kognitiven Psychologie spiegelt sich dies: Während sie sich vor 100 Jahren vor allem mit schlußfolgernden Denken und Problemlösen beschäftigte, ist dies heute nur noch ein kleiner Teil der Forschung [And89]. Viel wichtiger sind bespielsweise Fragen der Repräsentation von Wissen und der Organisation des Gedächtnisses.

Menschliches Wissen und Denken beruht wesentlich auf gemachten Erfahrungen und deren Verarbeitung. Ebenso können KI-Systeme nicht auf fertig in Regeln gepackte Erkenntnisse beruhen, sondern muß Methoden entwicklen, die aus Beispielen lernen.

1.3 Sprache und KI

Natürliche Sprachverarbeitung ist das herausfordernste Problem auf dem Gebiet der KI und steht gleichzeitig für ihr größtes Versagen. So ist der selbstgestellte Test, ob ein Computer denken kann, der *Turing Test*, eigentlich ein Sprachverarbeitungsproblem. Wenn ein Computer sich im Gespräch nicht von einem Menschen unterscheiden läßt, so Turing, muß man ihm Intelligenz zubilligen [Tur50].

Das erste sprachverarbeitende Programm, das über die KI-Gemeinde hinaus Aufsehen erregte, war *Eliza* von Joseph Weizenbaum [Wei66], das einen Psychologen simuliert. Seine Fähigkeiten sind für den Laien beeindruckend, obwohl es auf einfachem Pattern Matching beruht und sich auf Fragen und ausweichende Reaktionen beschränkt.

Die Methoden der Sprachverarbeitungen sind immer noch auf dem Stand von Wittgensteins „Tractatus". Die Bedeutung von Aussagen wird über das Frege'sche Kompositionalitätsprinzip erschlossen. Dies ergibt allerdings insofern Sinn, da heutige Forschungsziele sprachverarbeitende Systeme mit beschränkter Domäne sind. Wenn nur *ein* Sprachspiel gespielt wird, reduzieren sich viele Probleme der Semantik. Wenn beispielsweise in einem Zugauskunftssystem von einem „Zug" die Rede ist, spielen Aspekte wie das Alter des Zuges, die Höchstgeschwindigkeit oder dessen Gestalt keine Rolle. Und es ist auch klar, daß es nicht um eine Modelleisenbahn, einen Windzug oder um die metaphorische Benutzung des Wortes geht. Es interessiert nur, wohin er fährt und wann.

Die meisten Forscher auf dem Gebiet der maschinellen Sprachverarbeitung sind sich darüber im klaren, daß für die absehbare Zukunft nur solche beschränkten Systeme möglich sind. So wird zuwenig auf komplexere Fragen der Psycholinguistik eingegangen. Der Erkenntisstand, wie der Mensch Sprache verarbeitet, ist ja auch noch sehr gering. Eine beliebte Argumentation ist zwar der Vergleich mit dem künstlichen Fliegen: Ein Flugzeug schlägt nicht mit seinen Flügeln, es wird von Düsentriebwerken angetrieben. Inwieweit künstliches Sprechen auch auf völlig anderen Prinzipien beruhen kann, ist jedoch eher fraglich: Ziel ist schließlich ein Sprachverhalten, das vom menschlichen ununterscheidbar sein soll.

Der Loebner Preis ist eine vereinfachende Form des Turing Tests, bei dem das Gespräch thematisch und zeitlich beschränkt ist. Daß ein Computerprogramm einen echten Turing Test besteht, ist aus heutiger Sicht utopisch. Es ist bezeichnend für den Stand der Forschung, daß das bisher erfolgreichste Programm, das am Loebner Preis teilnahm, vor allem wirr daherredete und damit seine Gesprächsteilnehmer täuschte [Eps92].

1.4 Statistische Methoden

Der Begriff „Statistische Methoden" umfaßt vielerlei. Klassische stochastische Ansätze wie Hidden Markov Models (HMM) kann man ebenso dazuzählen wie Neuronale Netze. Ihr Reiz liegt darin, daß umfangreiches Wissen nicht mehr deklarativ beschrieben werden muß, sondern mittels Training aus Beispieldaten gewonnen wird.

Eine verlockende Idee, jedoch werden die Möglichkeiten statistischer Methoden gerne überschätzt. So schreibt zum Beispiel David Magerman in seiner Doktorarbeit:

> *I would have liked nothing more than to declare in my dissertation that linguists can be completely replaced by statistical analysis of corpora.* [3]

Es könnten hier noch weitere Zitate dieser Art genannt werden. Doch muß klar sein, daß statistische Methoden nur so gut sein können, wie ihr Rahmen ihnen vorgibt. Das Vorwissen, das man in ein solches Verfahren steckt, entscheidet über den Erfolg. Wird nichts vorgegeben, ist der Lösungsraum meist zu groß, um befriedigende Ergebnisse zu erhalten. So gesteht auch Magermann nach dem Abschluß seiner Arbeit ein:

> *This error analysis leads me to conclude that linguistic input is crucial to natural language parsing ...* [4]

Es ist ebenso unsinnig wie unpraktikabel, völlig auf linguistisches Vorwissen zu verzichten. Warum sollten auch grundlegende Erkenntnisse aufwendig durch statistische Analysen gewonnen werden, wenn es diese schon gibt? Es ist meine Überzeugung, daß Systeme zur Verarbeitung natürlicher Sprache einen wichtige Schritt vorangebracht werden können, wenn statistische Methoden und linguistisches Wissen verbunden werden.

In der Erkennung gesprochener Sprache haben mittlerweile statistische Methoden klassische linguistische Ansätze abgelöst [ST94]. Diese Arbeit ist Teil des Versuches, diesen Erfolg auf den Bereich der Syntax auszudehnen.

[3] Übersetzung: „Mir wäre nichts lieber gewesen als in meiner Dissertation zu erklären, daß Linguisten vollständig durch die statistische Analyse von Korpora ersetzt werden können." [Mag94]

[4] Übersetzung: „Diese Fehleranalyse führt mich zu der Schlußfolgerung, daß linguistische Eingaben entscheidend für das Parsing natürlicher Sprache ist." [Mag94]

1.5 Überblick

Kapitel zwei führt zunächst einige grundlegende Begriffe ein und im dritten Kapitel wird verwandte Forschung auf diesem Gebiet referiert. Kapitel vier gibt einen Überblick über das zentrale Anliegen der Arbeit: das Lernen von Unifikationsgrammatiken. Das fünfte Kapitel beschäftigt sich mit dem modellbasierten Lernen neuer Grammatikregeln, das sechste mit datengesteuerter Optimierung gelernter Regeln. Im siebten Kapitel werden experimentelle Ergebnisse des vorgeschlagenen Verfahrens berichtet. Kapitel acht bewertet dies und gibt Ausblick auf zukünftige Forschung auf diesem Gebiet.

Im Anhang findet sich die Beschreibung der im Verfahren verwendeten Grundgrammatik, das verwendete Korpus und Anmerkungen zur seiner Implementierung.

Kapitel 2

Grundlegende Begriffe

Dieses Kapitel führt einige Begriffe ein, die grundlegend für diese Arbeit sind. Nach der Erläuterung der grundsätzlichen Architektur sprachverarbeitender Systeme, wird der Formalismus der Unifikationsgrammatiken erklärt. Dieser wird für die Zwecke dieser Arbeit verfeinert. Schließlich wird das Chartparsing beschrieben.

2.1 Aufbau klassischer sprachverarbeitender Systeme

Sprachverarbeitung wird klassischerweise in drei Aspekte aufgeteilt:

- *Syntax* betrachtet Regeln und Beschränkungen, wie Wörter zu Sätzen zusammengefaßt werden dürfen.
- *Semantik* betrachtet die Bedeutung von Sätzen.
- *Pragmatik* untersucht schließlich die Verwendung von Aussagen in einem Kontext.

Zudem gibt es *Phonetik* und *Phonologie*, die sich mit Phänomenen in gesprochener Sprache beschäftigen. *Morphologie* untersucht die Bildung von Wortformen (z.B. `gegangen`, `ging`, `gehst`) eines Wortes (z.B. `gehen`).

Diese Arbeit beschränkt sich ausschließlich auf Syntax. Methoden zur Syntaxanalyse werden sowohl zur Spracherkennung als auch zum Sprachverstehen benötigt. Unter *Spracherkennung* versteht man die Erfassung sprachlicher Äußerungen. Mit *Sprachverstehen* bezeichnet man Versuche, Texte nach ihrer Bedeutung zu verarbeiten. Spracherkennung und Sprachverstehen werden oft als aufeinanderfolgend bei natürlichsprachlichen Systemen betrachtet. Syntaxanalyse ist dabei das gemeinsame Bindeglied.

Die Ansprüche an die Syntaxanalyse sind dabei unterschiedlich. Vom Blick der Spracherkennung werden Entscheidungshilfen erwartet, welche Wortfolgen möglich sind. Im Hinblick auf Sprachverstehen muß eine Äußerung so strukturiert werden, daß sie für die Semantikverarbeitung in sinnvolle Bestandteile aufgliedert ist.

Wichtige Bestandteile der Syntaxverarbeitung sind:

- Die *Grammatik* umfaßt Regeln, wie wohlgeformte Sätze strukturiert werden können. Das Entwickeln dieser Regeln steht im Mittelpunkt dieser Arbeit.

- Das *Lexikon* beinhaltet Wortformen und ihre grammatikalischen Eigenschaften.

- Der *Parser* ist das Verfahren, das unter Verwendung von Daten aus dem Lexikon und der Grammatik versucht, die Wohlgeformtheit eines Satzes zu überprüfen und ihn zu strukturieren.

2.2 Kompetenz versus Performanz

In der Linguistik wird zwischen *Kompetenz* und *Performanz* im Sprachgebrauch unterschieden. Unter Kompetenz versteht man die grammatisch richtige Verwendung der Sprache, unter Performanz die tatsächliche Verwendung bei spontan gesprochener Sprache.

Als Performanzphänomene gelten Versprecher, Neuansetzen inmitten eines Satzes und andere Fehler, die aus Zerstreutheit, Streß usw. resultieren. Äußerungen, die man der Sprachperformanz, nicht jedoch der Sprachkompetenz zuordnen würde, sind bespielsweise:

1 `Das ist ein schönes Bleispiel.`
2 `Ich gehe, äh, fahre jetzt.`
3 `Kannst Du mir das Dings, na, das du weißt schon geben?`

Als Performanz kann man die Menge der akzeptablen Äußerungen definieren [Osb94, Seite 4], also die Menge der Äußerungen, die von einem Hörer der Sprache verstanden werden können. Für Performanz kann man keine scharfe Grenze ziehen, der Übergang, ab wann Äußerungen unverständlich werden, ist fließend [ST94, Seite 202].

Interessanterweise gibt es auch Äußerungen, die der Kompetenz, nicht aber der Performanz zuzurechnen sind. Ein Beispiel übersetzt aus [Osb94]:

4 `Der Krieg, den der General, den der Präsident ernennt, startet, zerstört die Welt.`

Warum dieser grammatisch wohlgeformte Satz unverständlich (zumindest beim ersten Lesen) ist, läßt sich aus kognitionspsychologischer Sicht mit der Begrenzung des Kurzzeitgedächnisses [And89] begründen. Offenbar dürfen bestimmte grammatikalische Konstrukte, wie hier Relativsätze, nicht beliebig verschachtelt werden, ohne daß das Verständis leidet.

Vor allem da sich Performanzphänomene schwer in Regeln fassen lassen [ST94], konzentrieren sich die meisten linguistischen Ansätze auf Sprachkompetenz. Auch Osborne will in seinem Verfahren Regeln für Sprachkompetenz lernen und überläßt die Performanzeffekte einem nicht näher ausgeführten *sentence correction approach* [Osb94]. Dies reicht vor allem dann nicht aus, wenn es um die Verarbeitung gesprochener Sprache geht. Wohl nur die Hälfte unserer Äußerungen sind grammatisch wohlgeformte Sätze. Trotzdem beschränken auch wir uns in dem in dieser Arbeit vorgestellten Verfahren auf Kompetenz.

2.3 Kontextfreie Grammatiken

In der Syntaxanalyse natürlicher Sprachen haben sich Formalismen etabliert, die auf der Arbeit des Linguisten Noam Chomsky basieren. Chomsky schuf eine Hierarchie von Regelsystemen, sogenannte *Grammatiken*: regulär, kontextfrei, kontextsensitiv und rekursiv aufzählbar. In dieser Reihenfolge sind reguläre Grammatiken die am wenigsten mächtigen, dafür aber am einfachsten zu verarbeiten.

Für natürliche Sprachen werden in der Regel kontextfreie Grammatiken verwendet [All95, Seite 42ff]. Dieser Formalismus wird als mächtig genug erachtet, obwohl es Hinweise gibt, daß er nicht für alle Sprachen ausreicht, wie Konstrukte im Schwyzerdeutschen zeigen [Shi85].

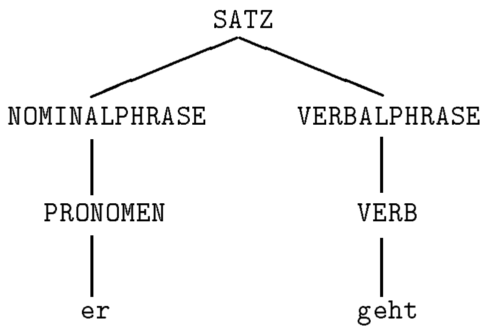
Formal gesehen sind Grammatiken eine Menge von Symbolen und Regeln. Kontextfreie Regeln haben beispielsweise folgende Form:

```
SATZ → NOMINALPHRASE VERBALPHRASE
NOMINALPHRASE → PRONOMEN
PRONOMEN → er
VERBALPHRASE → VERB
VERB → geht
```

Man unterscheidet zwischen *nichtterminalen* und *terminalen* Symbolen. Nichtterminale sind hier in Großbuchstaben dargestellt und repräsentieren meist grammatische Begriffe. Terminale in Kleinbuchstaben stehen für Wortformen. *Kontextfreie Grammatiken* zeichnen sich dadurch aus, daß sie nur ein nichtterminales Symbol auf der linken Regelseite erlauben.

Ausgehend von einem Startsymbol kann durch Anwendung der Regeln — auch *Ableitung* genannt — eine Menge von Ketten nichtterminaler Symbole erzeugt werden. In Grammatiken, die natürliche Sprachen modellieren sollen, ist dies die Menge der möglichen Sätze der Sprache. Abbildung 2.1 zeigt den *Ableitungsbaum* des Satzes `er geht`, wie er sich aus der obigen Grammatik ergibt.

Abbildung 2.1: Ableitungsbaum des Satzes `er geht`



2.4 Merkmalsstrukturen und Unifikation

In den obigen Beispielen sind die Nichtterminale atomare Symbole, die grammatische Kategorien repräsentieren. Nun will man jedoch zusätzliche Eigenschaften angeben: `er` ist ein Pronomen, dritte Person Maskulin Singular im Nominativ. Das Subjekt des Satzes und das Verb müssen in Numerus und Genus übereinstimmen. Ein Nomen muß mit einem ihm vorangestellten Adjektiv in Numerus, Genus und Kasus übereinstimmen. Diese Informationen werden in Form von Merkmalsstrukturen kodiert.

Eine *Merkmalsstruktur* besteht aus einer Menge von *Merkmalen*, die bestimmte Werte annehmen können. Das Pronomen `er` könnte also durch die folgende Merkmalsstruktur beschrieben werden:

$$\begin{bmatrix} \text{CAT: } \textit{pronoun} \\ \text{PERSON: } \textit{third} \\ \text{NUMBER: } \textit{singular} \\ \text{GENDER: } \textit{male} \\ \text{CASE: } \textit{nominativ} \end{bmatrix}$$

Diese Merkmalsstrukturen können an Stelle von atomaren Symbolen nun auch in Grammatikregeln verwendet werden. Die Regel, daß einem Nomen ein Adjektiv vorangestellt werden kann, das in Numerus, Genus und Kasus übereinstimmt, kann beispielsweise so formuliert werden:

$$\begin{bmatrix} \text{CAT: } \textit{noun} \\ \text{NUMBER: } \langle 1 \rangle \\ \text{GENDER: } \langle 2 \rangle \\ \text{CASE: } \langle 3 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT: } \textit{adjective} \\ \text{NUMBER: } \langle 1 \rangle \\ \text{GENDER: } \langle 2 \rangle \\ \text{CASE: } \langle 3 \rangle \end{bmatrix} \begin{bmatrix} \text{CAT: } \textit{noun} \\ \text{NUMBER: } \langle 1 \rangle \\ \text{GENDER: } \langle 2 \rangle \\ \text{CASE: } \langle 3 \rangle \end{bmatrix}$$

Statt festen Werten sind hier die Merkmale mit Variablen besetzt. Da zum Beispiel der Numerus in Adjektiv und Nomen mit der selben Variablen $\langle 1 \rangle$ besetzt ist, kann diese Regel auch nur bei übereinstimmenden Numerus angewendet werden.

Unifikation

In einfachen kontextfreien Grammatiken ist eine Regel anwendbar, wenn die Symbole in der Regel den Symbolen in der Symbolkette gleichen. In Merkmalsstrukturen gibt es hingegen eine Reihe von Merkmalen, die mit Werten besetzt sein können, aber nicht müssen. Ob eine Regel angewendet werden kann, hängt daher davon ab, ob die Strukturen unifizierbar sind.

Unifikation ist praktisch das Ersetzen von Variablen mit Werten oder anderen Variablen, so daß sich zwei Strukturen gleichen. Eine Grammatik mit Merkmalsstrukturen nennt man daher auch *Unifikationsgrammatik*.

Dies soll nun anhand eines Beispiels verdeutlicht werden. Angenommen wir finden in einem Satzteil die beiden Worte `scharfe Augen`. Nun soll überprüft werden, ob die obige Adjektivregel angewandt werden kann.

Die Lexikoneinträge für `scharfe` und `Augen`, die in diesem Zusammenhang relevant sind, lauten:

$$\begin{bmatrix} \text{CAT: } \textit{adjective} \\ \text{NUMBER: } \textit{plural} \\ \text{CASE: } \textit{nominativ} \end{bmatrix} \rightarrow \texttt{scharfe} \qquad \begin{bmatrix} \text{CAT: } \textit{nomen} \\ \text{NUMBER: } \textit{plural} \\ \text{GENDER: } \textit{neutral} \end{bmatrix} \rightarrow \texttt{Augen}$$

Überprüfen wir zunächst, ob sich der Lexikoneintrag für `scharfe` mit dem ersten Symbol der rechten Seite in der Regel unifizieren läßt. Offensichlich stimmen die Werte für das Merkmal CAT überein. Für die Merkmale NUMBER und CASE müssen zwei Variablen ersetzt werden:

$$\begin{aligned} \langle 1 \rangle &= \textit{plural} \\ \langle 3 \rangle &= \textit{nominativ} \end{aligned}$$

Die Unifikation ist damit erfolgreich beendet. Sehen wir nun, ob sich das zweite Symbol der rechten Regelseite mit `Augen` unifizieren läßt. Wieder stimmen die Werte für das Merkmal CAT überein. Die Variable $\langle 1 \rangle$ für das Merkmal NUMBER wurde inzwischen durch den Wert *plural* ersetzt. Dieser stimmt mit dem Wert aus dem Lexikoneintrag für `Augen` überein. Bleibt schließlich die Variable $\langle 2 \rangle$, die ersetzt werden muß:

$$\langle 2 \rangle = \textit{neutral}$$

Damit ist auch diese Unifikation erfolgreich, die Regel kann also angewendet werden:

$$\begin{bmatrix} \text{CAT}: \textit{noun} \\ \text{NUMBER}: \textit{plural} \\ \text{GENDER}: \textit{neutral} \\ \text{CASE}: \textit{nominativ} \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT}: \textit{adjective} \\ \text{NUMBER}: \textit{plural} \\ \text{GENDER}: \textit{neutral} \\ \text{CASE}: \textit{nominativ} \end{bmatrix} \begin{bmatrix} \text{CAT}: \textit{noun} \\ \text{NUMBER}: \textit{plural} \\ \text{GENDER}: \textit{neutral} \\ \text{CASE}: \textit{nominativ} \end{bmatrix}$$

Wenn bei der Unifikation ein Merkmal nur in einer der beiden Strukturen vorkommt, wird dieses ins Ergebnis übernommen. So war das Merkmal CASE im Lexikoneintrag von `Augen` nicht besetzt (weil die Wortformen für `Auge` im Plural unabhängig vom Kasus gleich sind). Trotzdem war eine Unifikation möglich, das Merkmal wurde mit dem Wert *nominativ* besetzt.

Unifikation wurde hier sehr informell eingeführt. Wer eine formalere Definition sucht, wendet sich zum Beispiel an [All95, Seite 598ff].

Typen von Merkmalen

Im letzten Abschnitt wurden Merkmalsstrukturen in ihrer einfachsten Form motiviert und eingeführt. Bevor wir an das Erweitern dieses Konzeptes gehen, soll zunächst der Zweck von Merkmalen genauer analysiert werden.

Die für diese Arbeit entwickelten Merkmale sind in Tabelle 2.1 aufgeführt. Eine Auflistung der Kathegorien und mit welchen Merkmalen sie vorkommen findet sich in Tabelle 2.2. Es sei dabei hingewiesen, daß die Bezeichnungen pragmatisch verwendet wurden und sich nicht an linguistischen Theorien orientieren. Es hat sich bei der Entwicklung der Grammatik für diese Arbeit als sinnvoll ergeben, die Merkmale in drei Typen einzuteilen:

1. **Übereinstimmung** — Oft müssen gewisse grammatikalische Eigenschaften innerhalb eines Satzes oder einer Phrase übereinstimmen. In der obigen Regel wurde zum Beispiel die Übereinstimmung von Numerus, Genus und Kasus verlangt. Die Merkmale NUMBER, GENDER, CASE und auch PERSON sind somit von diesem Typ.

Tabelle 2.1: Merkmale

Merkmal	Typ	Beschreibung
SUBCAT	Genauere Bestimmung	Grammatische Unterkategorie
PERSON	Übereinstimmung	Person
CASE	Übereinstimmung	Fall
AGR	Übereinstimmung	Numerus und Genus
POS	Genauere Bestimmung	Position im Satz
DET	Übereinstimmung	Vorhandensein eines Artikels
DEF	Übereinstimmung	Bestimmtheit der Nominalphrase
RREQ	Anforderung	Relativer Bezug
RAGR	Übereinstimmung	Relativer Bezug
TAGR	Anforderung	Zeit
VAGR	Anforderung	Anforderungen des Verbes
SEMCAT	Genauere Bestimmung	Semantische Kategorie

Tabelle 2.2: Grammatische Kategorien und ihre Merkmale

Kat.	Beschreibung und Merkmale
`s`	Satz
`vp`	Verbalphrase (Satzteil) – SUBCAT PERSON AGR POS RREQ RAGR TAGR VAGR
`v`	Verb – SUBCAT PERSON VAGR AGR
`comp`	Argument für Verbalphrase – SUBCAT PERSON AGR RREQ RAGR VAGR
`np`	Nominalphrase – SUBCAT RREQ RAGR AGR TAGR SEMCAT
`n`	Nomen – CASE AGR DET DEF SEMCAT
`pn`	Name – SEMCAT
`det`	Artikel – CASE AGR DEF
`adj`	Adjektiv – CASE AGR DEF SEMCAT
`pro`	Pronomen – SUBCAT AGR CASE
`pp`	Präpositionalphrase – RREQ RAGR SEMCAT
`p`	Präposition – SEMCAT CASE DET AGR
`adv`	Adverb – SEMCAT
`junk`	Grammatisch irrelevante Adjunkte zu Verbalphrasen – SUBCAT
`sc`	Nebensatz – SUBCAT POS RREQ RAGR
`rc`	Relative Komponente – SUBCAT PERSON AGR POS RAGR VAGR
`tc`	Zeitphrase – SUBCAT TAGR
`tp`	zeitliche Präpositionalphrase – SEMCAT
`t`	Zeitwort – SUBCAT SEMCAT
`part`	Partikel – SUBCAT
`vcomp`	Verbkomplement – SUBCAT

2. **Genauere Bestimmung** — Andere Merkmale bestimmen lediglich einen Satzteil näher, um seine Verwendung gegen andere abzugrenzen. So legt CAT die grammatikalische Kategorie fest. Dies ermöglicht beispielsweise, daß Verben grundsätzlich anders behandelt werden können als Nomen oder Adjektive. Andere Beispiele der in dieser Arbeit verwendeten Grammatik sind SUBCAT, SEMCAT und POS.

3. **Anfordung** — Einige grammatische Phänomene lassen sich nicht lokal innerhalb einer Regel abhandeln. Wenn das Verb `sagen` im Satz auftaucht, muß es ebenfalls ein Subjekt (wer sagt?), ein Satzkomplement (was wird gesagt?) und eventuell ein Dativ-Objekt (wem wird etwas gesagt?) geben. Wegen der freien Wortstellung des Deutschen können diese Komponenten an vielen verschiedenen Positionen des Satzes erscheinen. Für diese Anforderungen des Verbes wird das Merkmal VAGR verwendet. Ein anderes Beispiel ist das Merkmal TAGR.

Der letzten Typ von Merkmalen läßt sich nur sehr umständlich durch den im letzten Abschnitt vorgeschlagenen Formalismus von Merkmalsstrukturen realisieren. Auch andere Überlegungen motivieren, einige Erweiterungen vorzunehmen. Dies soll im nächsten Abschnitt geschehen.

2.5 Erweiterung von Merkmalsstrukturen

Substrukturierung

Um die Formulierung von Grammatikregeln übersichtlicher zu gestalten, bietet es sich an, Merkmale, die gleich oder sehr ähnlich verwendet werden, zusammenzufassen. Ein Beispiel ist Person, Numerus und Genus eines Nomens oder einer Nominalphrase, die in einer sogenannten *agreement structure* (AGR) zusammengefaßt werden. Für den Lexikoneintrag von `er` sieht dies dann folgendermaßen aus:

$$\begin{bmatrix} \text{CAT: } \textit{pronoun} \\ \text{AGR: } \begin{bmatrix} \text{PERSON: } \textit{third} \\ \text{NUMBER: } \textit{singular} \\ \text{GENDER: } \textit{male} \end{bmatrix} \\ \text{CASE: } \textit{nominativ} \end{bmatrix}$$

Solch eine Substruktur wird in der Grammatik im Prinzip wie ein normales Merkmal behandelt. Die obige Adjektivregel vereinfacht sich damit zu:

$$\begin{bmatrix} \text{CAT: } \textit{noun} \\ \text{AGR: } \langle 1 \rangle \\ \text{CASE: } \langle 2 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT: } \textit{adjective} \\ \text{AGR: } \langle 1 \rangle \\ \text{CASE: } \langle 2 \rangle \end{bmatrix} \begin{bmatrix} \text{CAT: } \textit{noun} \\ \text{AGR: } \langle 1 \rangle \\ \text{CASE: } \langle 2 \rangle \end{bmatrix}$$

Paarbildung

Gelegentlich ist es sinnvoll, Paare von Werten als Merkmalseintrag zuzulassen. Werfen wir zunächst einen Blick auf die Präpositionalphrasen in den folgenden Sätzen:

5 `Ich fahre in München mit dem Zug.`
6 `Ich gehe ab Hamburg mit meiner Großmutter.`
7 `Er fährt von Nürnberg über Erlangen nach Bamberg mit einem Koffer.`
8 `Er schrieb das Buch mit meiner Hilfe.`

Alle vorkommenden Präpositionalphrasen sind adjunkt, das heißt der Satz wäre auch ohne sie grammatisch korrekt. Angenommen, wir wollten nun Regeln formulieren, die bestimmte Präpositionalphrasen als Adjunkte zulassen. Zum einen könnten wir zulassen, daß jeder beliebige Satz Präpositionalphrasen mit der Präposition `mit` enthalten darf. Zum anderen erlauben Sätze mit Bewegungsverben adjunkte Ortsangaben.

Die Verwendung von Präpositionalphrasen hängt also sowohl von der Präposition als auch von der Nominalphrase ab. Beides sollte also gleichermaßen in der Merkmalsstruktur kodiert sein. Dies ist ein Beispiel dafür, daß es sinnvoll sein kann, daß ein Merkmal ein Paar von Werten enthält.

Um das Beispiel abzuschließen, hier nun eine mögliche Merkmalsstruktur für die Präpositionalphrase `mit dem zug`:

$$\begin{bmatrix} \mathrm{CAT}: \textit{prepositionial phrase} \\ \mathrm{SEMCAT}: (\textit{mit}, \textit{zug}) \end{bmatrix}$$

Anforderungen

Ein wesentlicher Punkt, der die Sprachverarbeitung des Deutschen gegenüber dem Englischen erschwert, ist die freie Wortstellung. Für den englischen Satz

9 `He gives me the book`

gibt keine Möglichkeiten, Wörter umzustellen ohne ihn — zumindest grammatikalisch — zu ändern (In `He gives the book to me` ist das Dativobjekt `me` durch das Präpositionalobjekt `to me` ersetzt). Für die deutsche Übersetzung gibt es jedoch sechs Varianten:

10 `Er gibt mir das Buch`
11 `Er gibt das Buch mir`
12 `Das Buch gibt er mir`
13 `Das Buch gibt mir er`
14 `Mir gibt er das Buch`
15 `Mir gibt das Buch er`

Die einzige Regel scheint zu sein, daß das Verb an zweiter Stelle steht, während die anderen Satzkomponenten beliebig positioniert werden können.

Nun wäre es jedoch zu aufwendig für jede Konstellation eine eigene Regel aufzustellen. Dies gilt insbesondere wenn man bedenkt, daß zusätzliche adjunkte Präpositionalphrasen hinzugefügt werden können: `mit seiner rechten Hand`, `im Vorbeigehen`, `am Samstag` und so fort.

Für diese Arbeit wurde daher ein Formulismus entwickelt, in dem die einzelnen Komponenten nach und nach in die Satzstruktur eingebaut werden. Wie dies im einzelnen geschieht, ist aus der Beschreibung der Grammatik im Anhang A zu ersehen. In diesem Abschnitt soll nur darauf hingewiesen werden, welche Erweiterungen an den Merkmalsstrukturen vonnöten sind.

Ein Verb, das in die Satzstruktur eingebaut werden muß, stellt gewisse Anforderungen. Für das Verb `geben` könnte man diese so formulieren:

$$\text{VAGR:} \begin{bmatrix} \text{SUBJECT: need} \\ \text{DATIV OBJECT: may} \\ \text{ACCUSATIV OBJECT: need} \end{bmatrix}$$

Das Verb `geben` braucht ein Subjekt und ein Akkusativobjekt (need). Ein Dativobjekt ist möglich (may), aber nicht unbedingt notwendig. Diese Eigenschaften des Verbes sind in der *verb agreement* Substruktur (VAGR) angegeben.

Eine Satzkomponente kann einzelne dieser Anforderungen erfüllen. Dies wird entsprechend im VAGR Merkmal vermerkt. Die Formulierung für eine Satzkomponente, die das Subjekt — sagen wir: `er` — beinhaltet, sieht dann so aus:

$$\text{VAGR:} \begin{bmatrix} \text{SUBJECT: has} \end{bmatrix}$$

Wie bereits oben erwähnt, werden die Satzkomponenten nach und nach in die Satzstruktur eingebaut. Diese Satzstruktur sollte also nach dem Einbau von `er` und `gibt` so aussehen:

$$\text{VAGR:} \begin{bmatrix} \text{SUBJECT: satisfied} \\ \text{DATIV OBJECT: may} \\ \text{ACCUSATIV OBJECT: need} \end{bmatrix}$$

Die Subjektanforderung ist also erfüllt (satisfied), das Akkusativobjekt steht noch aus (need) und ein Dativobjekt ist weiterhin möglich (may). Nach dem Einbau aller Wörter sollte der Satz nur dann akzeptiert werden, wenn alle Anforderungen erfüllt wurden (complete).

Der eben beispielhafte beschriebene Mechanismus läßt sich nur sehr umständlich mit einfacher Unifikation realisieren. So sollte das Vorhandensein von zwei Subjekten zum Abbruch führen (has $\sqcup$ has $\mapsto$ *fail*), die Erfüllung einer Anforderung jedoch nicht (need $\sqcup$ has $\mapsto$ satisfied). Normale Unifikation würde jedoch ersteres akzeptieren (has $\sqcup$ has $\mapsto$ has) und bei letzterem fehlschlagen (need $\sqcup$ has $\mapsto$ *fail*).

In Tabelle 2.3 ist das gewünschte Verhalten des Unifikationsmechanismus dargestellt. *no value* bedeutet dabei, daß weder Anforderungen oder Vorhandensein dieser Satzkomponente aufgestellt wurde. Falls zwei Zustände nicht unifizierbar sind, ist dies durch *fail* angegeben. Ein $\bullet$ gibt an, daß dieser Zustand erlaubt ist, wenn die Struktur am Ende überprüft wird (complete). Es darf also kein Zustand vorhanden sein, der mit $\circ$ markiert ist: Weder unbefriedigte Anforderung, noch unangeforderte Satzkomponenten.

2.6 Weiteres zu Merkmalsstrukturen

Zum Abschluß des Themas Merkmalsstrukturen hier noch einige theoretische Anmerkungen.

Tabelle 2.3: Verarbeitung der Anforderungszustände

$\sqcup$	*no value*	need	may	has	satisfied
• *no value*	*no value*	need	may	has	satisfied
○ need	need	*fail*	*fail*	satisfied	*fail*
• may	may	*fail*	*fail*	satisfied	*fail*
○ has	has	satisfied	satisfied	*fail*	*fail*
• satisfied	satisfied	*fail*	*fail*	*fail*	*fail*

Mächtigkeit

In der ursprünglichen Form haben wir hier als Eintrag für ein Merkmal nur eine Variable oder ein Element aus einer endlichen Menge von Werten erlaubt. Mit dieser Einschränkung erhöhen wir die Mächtigkeit des Grammatikformalismus gegenüber kontextfreien Grammatiken nicht. Das heißt, daß eine kontextfreie Grammatik, deren Komponenten Merkmalsstrukturen sind, keine anderen Sprachen erzeugen kann als eine kontextfreie Grammatik mit elementaren Symbolen.

Dies gilt auch für die Erweiterungen, sofern sichergestellt ist, daß Paarbildung und Substrukturierung nicht rekursiv angewendet werden können. Oder anders formuliert: Die Menge möglicher Merkmalsstrukturen muß endlich bleiben.

Der Vorteil von Merkmalsstrukturen liegt in der erhöhten Effizienz der Notation: Durch die Einführung von Variablen können viele ähnliche elementare Regeln in eine einzige Regel gepackt werden. Dies vereinfacht die Formulierung von Regeln und beschleunigt das Parsing.

Kontextfreies Rückgrat

Der hier vorgestellte Mechanismus einer Unifikationsgrammatik läßt sich auf zweierlei Weise auf eine kontextfreie Grammatik abbilden. Wie oben beschrieben, ist sie eine gepackte kontextfreie Grammatik, da eine Regel, die Merkmalsstrukturen mit Variablen verwendet, mehrere kontextfreie Regeln mit verschiedenen Variablenbelegungen zusammenfaßt.

Unter einer zweiten Sichtweise ist eine Reduktion der Unifikationsgrammatik auf ein *kontextfreies Rückgrat* möglich. Wenn außer dem Kategorien-Merkmal CAT alle Merkmale weggelassen werden, erhält man wiederum eine kontextfreie Grammatik. Aus diesem Blickwinkel ist die tatsächliche Unifikationsgrammtik nur eine Verfeinerung der zugrundeliegenden kontextfreien Grammatik. Die besondere Bedeutung der Kategorien spiegelt sich auch in der Darstellung der Regeln im Anhang C wieder. Diese Sichtweise liegt auch dem Lernverfahren zugrunde - linguistisches Wissen ist auf der Ebene der Kategorien formalisiert.

Diese Bildung eines kontextfreien Rückgrates der Unifikationsgrammtik spielt jedoch keine Rolle bei ihrer Verwendung im Parsing, das im folgenden Abschnitt behandelt wird. Bei der Besprechung der Konstruktion neuer Regeln tritt sie jedoch deutlich hervor.

2.7 Chart Parsing

In den vorigen Abschnitten dieses Kapitels wurde ausführlich der Formalismus erläutert, in dem natürliche Sprachen beschrieben werden. Mit diesem Formalismus sollte es nun möglich sein, einen beliebige natürlichsprachliche Äußerung zu *parsen*. Parsing heißt, zu entscheiden, ob die Äußerung grammatisch wohlgeformt ist und wenn ja, sie ihn in ihre grammatikalischen Bestandteile aufzulösen.

Das hier vorgestellte *Chart Parsing* ist dadurch gekennzeichnet, daß Zwischenergebnisse (*Kanten*) in einer sogenannten *Chart* festgehalten werden. Diese Zwischenergebnisse sind begonnene oder abgeschlossene Regelanwendungen, zusammen mit der Information, welche Teile der Eingabe sie überspannen.

Dieser Begriff eines Chartparsers legt noch nicht die Parsingstrategie fest. Neben den klassischen Ansätzen *top down parsing* und *bottom up parsing* sind viele andere Agenda-getriebene Strategien möglich. Ein weiterführendes Buch zum Thema Chartparsing ist [Gör88].

Für diese Arbeit wird eine Variation des bottom up parsing verwendet, die auf den Ansätzen in [Cov94] aufbaut. Falls eine Äußerung nicht vollständig geparst werden kann, liefert bottom up parsing immerhin alle möglichen Teilparses. Dies ist wichtig, da neue Grammatikregeln gelernt werden sollen, die auf diesen Teilparses fußen.

Das Parsingverfahren durchläuft den Satz Wort für Wort und besteht aus der Aufeinanderfolge von Scanner- und Komplettierungsschritten. Der Scanner sucht zu einem Wort die entsprechenden Lexikoneinträge und schreibt sie als Kanten in der Chart. Der Komplementierer sucht Regeln, deren rechte Seite mit diesen neuen Kanten abschließt. Daraufhin sucht er, falls nötig, alte Kanten, die die rechte Regelseite vervollständigen. Findet er diese, ist die Regel anwendbar und die linke Seite wird als neue Kante in die Chart geschrieben. Der Komplementierer versucht nun mit dieser neuen Kante erneut Regeln anzuwenden und so fort.

Ein Beispiel

Um die Funktionsweise des bottom up Chartparsers zu verdeutlichen, wollen wir uns an dieser Stelle das Parsing des Satzes `er geht` mit Hilfe der kontextfreien Grammatik von Seite 8 detailliert verdeutlichen:

Regeln:
(A) SATZ → NOMINALPHRASE VERBALPHRASE
(B) NOMINALPHRASE → PRONOMEN
(C) VERBALPHRASE → VERB
Lexikon:
PRONOMEN → er
VERB → geht

Das Parsing nimmt nun folgenden Verlauf:

Wort `er`

Scanner Im Lexikon ist das Wort als `PRONOMEN` klassifiziert. Dies wird also als Kante 1 in die Chart geschrieben.

Komplettierer Regel B ist nun auf diese Kante anwendbar, keine weiteren Kanten sind vonnöten und die linke Seite der Regel `NOMINALPHRASE` kann nun als zweite Kante in die Chart geschrieben werden.
Auf diese neue Kante ist keine Regel vollständig anwendbar. Die Chart zu diesem Zeitpunkt ist in Abbildung 2.2 illustriert.

Abbildung 2.2: Chart nach dem Parsen von `er`.

2: NOMINALPHRASE	
1: PRONOMEN	
er	geht

Wort `geht`

- **Scanner** Im Lexikon ist `geht` als `VERB` klassifiziert. Dies wird als Kante 3 in der Chart vermerkt.
- **Komplettierer** Der Komplettierer sucht nach Regeln, die sich auf diese neue Kante anwenden lassen und findet Regel C. Somit wird als Kante 4 `VERBALPHRASE` in die Chart eingetragen.
  Da `VERBALPHRASE` als Abschluß der rechten Seite von Regel A auftaucht, wird nun nach einer Kante `NOMINALPHRASE` in der Chart gesucht und in Kante 2 gefunden. Regel A ist somit anwendbar und die fünfte Kante `SATZ` wird in die Chart geschrieben. Keine weiteren Regeln können angewendet werden. Die Chart ist in Abbildung 2.3 illustriert.

Abbildung 2.3: Chart nach dem Parsen von `er geht`.

5: SATZ	
2: NOMINALPHRASE	4: VERBALPHRASE
1: PRONOMEN	3: VERB
er	geht

Nach dem Ende des Parsings findet sich eine Kante `SATZ` in der Chart, die den gesamten Satz überspannt. Der Satz wurde also erfolgreich geparst. Es ist möglich, daß mehrere solche `SATZ`-Kanten erzeugt werden. Dies deutet auf Mehrdeutigkeiten in der Grammatik hin.

Ebenfalls ist es möglich, daß eine `SATZ`-Kante erzeugt wurde, die nicht den ganzen Satz überspannt. Bei der oben verwendeten Grammatik wäre dies zum Beispiel `er geht nach hause` oder `auch er geht`. Dies sind keine erfolgreichen Parses.

Probleme beim bottom up Parsing

Der Vorteil des Chartparsings liegt darin, daß keine Regelanwendungen wiederholt werden müssen, da alle Zwischenergebnisse als Kanten gespeichert sind.

Ein Problem des bottom up Parsings liegt darin, daß das Verfahren in eine Endlosschleife laufen kann, wenn es rekursive Regelanwendungen gibt. Gibt es beispielsweise zwei Regeln $\mathtt{A} \rightarrow \mathtt{B}$ und $\mathtt{B} \rightarrow \mathtt{A}$, so können diese beiden Regeln endlos hintereinander angewendet werden. Dies kann dadurch unterbunden werden, daß keine Kanten in die Chart geschrieben werden dürfen, die schon darin enthalten sind.

In dieser Arbeit gibt es diese Beschränkung nicht. Der Grund dafür ist, daß für das Lernverfahren die Parsebäume von Bedeutung sind. Zwei Kanten, die sich nur in ihren Parsebäumen unterscheiden, müssen daher unterschiedlich behandelt werden. Daher ist später beim Lernen von neuen Regeln streng darauf zu achten, daß es dadurch zu keinen rekursiven Anwendungen unärer Regeln kommen darf.

Kapitel 3

Verwandte Forschung

Die Arbeit des Philosophen ist das Zusammentragen von Erinnerungen zu einem bestimmten Zweck

Ludwig Wittgenstein
Philosophische Untersuchungen, 127

In diesem Kapitel wird zunächst auf die Unterschiede zwischen dem Deutschen und dem Englischen eingangen. Anschließend werden verschiedene statistische Ansätze zur Sprachverarbeitung vogestellt: n-Gram Modelle, probabilistische kontextfreie Grammatiken und schließlich die Erweiterung von Grammatiken, was die Grundlage dieser Arbeit ist.

3.1 Englisch vs. Deutsch

Die meiste Literatur auf dem Gebiet der maschinellen Sprachverarbeitung behandelt die englische Sprache. Für das Englische gibt es daher die besten Korpora und die besten Baumbanken geparster Sätze.

Zwischen dem Deutschen und dem Englischen gibt es wesentliche Unterschiede. So ist ein bedeutendes Forschungsthema für das Englische *Part of Speech Tagging*, das Zuweisen der grammatischen Kategorie (der *Tag*) zu Worten in Sätzen. Ein klassisches Beispiel ist:

1 `Time flies like an arrow.`

In diesem Satz hängt die Bedeutung entscheidend davon ab, welche Tags den Wörtern `flies` und `like` zugewiesen werden. `flies` ist entweder ein Nomen oder ein Verb, `like` kann ein Verb, eine Präposition und sogar ein Nomen sein. Es geben sich für diesen Satz daher zwei Lesarten: ,,Zeitfliegen mögen einen Pfeil" oder ,,Zeit fliegt wie ein Pfeil".

Im Deutschen ist dies jedoch kaum ein Problem. In der deutschen Sprache liefert die Morphologie wesentlich mehr Informationen. Dagegen gibt es die freie Satzstellung. Diese hat insbesondere zur Folge das es entferntere Bezüge zwischen Satzteilen gibt als im Englischen. Nehmen wir den Satz

2 `Den entscheidenden Hinweis gab in Erlangen mir trotz einigem Zögern Herr Weber, wo eine alljährliche Konferenz zu diesem Thema stattfindet.`

Zwischen dem Verb `gab` und den Subjekt des Satzes `Herr Weber` liegen drei andere Satzkomponenten, die bei der Prüfung auf Gleichheit in Numerus und Person übergangen werden müssen. Der Relativsatz ist ebenso weit von seinem Bezug `Erlangen` entfernt.

Wir wollen an dieser Stelle nicht weiter auf Unterschiede zwischen dem Englischen und dem Deutschen eingehen. Es ist jedoch wichtig im Kopf zu behalten, daß Methoden, die im Englischen erfolgreich sind, sich möglicherweise nicht so einfach auf das Deutsche übertragen lassen. Gerade die freie Satzstellung mag ein entscheidender Nachteil für lokale Verfahren wie kontextfreie Grammatiken oder n-Gram Modelle sein.

Ein gemeinsames Problem in beiden Sprachen ist der Umgang mit Mehrdeutigkeiten (*Ambiguitäten*). Für den Satz

3 `Können Sie mir Auskunft über einen Zug am Samstag in München geben?`

erzeugt ein Parser, der mehrdeutige Zuordnungen von Präpositionalphrasen nicht auflöst, bis zu sieben verschiedene Parses. Sie unterscheiden sich in ihrer Bedeutung etwa, ob die Auskunft am Samstag gegeben werden soll, oder ob von einem Samstagszug die Rede ist. Es können noch viel extremere Beispiele genannt werden, da die Anzahl der Parses exponentiell mit der Zahl der Präpositionalphrasen wächst.

Für das menschliche Sprachverstehen sind die meisten dieser Möglichkeiten jedoch sinnlos: etwa daß die Auskunft in München oder mittels eines Zuges gegeben werden soll. Das Auflösen dieser Mehrdeutigkeiten wird *Disambiguierung* gennant.

Die angegebene Mehrdeutigkeit ist eine *semantische Ambiguität*. Es gibt jedoch das Problem der Mehrdeutigkeit auf praktisch allen Ebenen der Sprachverarbeitung. Diese reichen von der Unterscheidung von Homonymen bei der Spracherkennung (`kam` oder `Kamm`?) bis hin zu Anaphernresolution bei der Pragmatik.

3.2 N-Gram Modelle über Wortfolgen

Nach dem Erfolg statistischer Verfahren auf dem Gebiet der Worterkennung, gibt es seit einiger Zeit Bemühungen, diese Methodik auf den nächsten Schritt zu übertragen: statistische Syntaxanalyse. Die Hoffnung ist, dadurch Probleme wie das Disambiguieren von Mehrdeutigkeiten mit der Hilfe von Wahrscheinlichkeiten aufzulösen. Als am wahrscheinlichsten gilt daher die Interpretation, die von den meisten Beispieldaten (der *Evidenz*) gestützt wird.

Als Verfahren zur Syntaxmodellierung sind in der Spracherkennung dabei noch einfache *n-Gram Modelle* über die Aufeinanderfolge von Wörtern Stand der Forschung. N-Gram Modelle machen Vorhersagen über die Wahrscheinlichkeiten eines Wortes basierend auf seinen Vorgängerwörtern. Dazu müssen Statistiken über die Folge von Wörtern durch die Auswertung eines großen Korpus von Sätzen angelegt werden. Die Statistiken können auch auf syntaktischen Kategorien beruhen, wie im Beispiel in Abbildung 3.1. (Für eine ausführliche Beschreibung siehe [ST94, Kapitel 7].)

Abbildung 3.1: Beispielhafter Ausschnitt aus einer Bigram-Datenbank

Die hier angegebenen Zahlen sind bedingte Wahrscheinlichkeiten $P(w_j \mid w_i)$. Die Werte geben also an, wie wahrscheinlich das Folgewort zu einer bestimmten Kategorie gehört, wenn gerade ein Artikel aufgetreten ist. Die Beispielwerte in der Tabelle sind willkürlich gewählt. Natürlich muß gelten: $\forall w_i : \sum_{w_j \epsilon W} P(w_j \mid w_i) = 1$.

Wahrscheinlichkeit	Folge von Kategorien	
	w_i	w_j
0.75	Artikel	Nomen
0.16	Artikel	Adjektiv
0.06	Artikel	Adverb
0.02	Artikel	Artikel
0.01	Artikel	Verb

Die Verwendung von solchen n-Gram Modellen ist, wie schon angedeutet, aus der Spracherkennung heraus motiviert. Worterkenner erzeugen aus einer akustischen Eingabe eine große Menge von Worthypothesen mit verschiedenen Anfangs- und Endzeitpunkten. Diese Menge wird als *Lattice* bezeichnet. Nun muß sehr effizient entschieden werden können, welche Wörter zusammenpassen können, so daß eine sinnvolle Äußerung festgestellt werden kann. N-Gram Modelle liefern diese Information sehr schnell: die entsprechenden Wahrscheinlichkeiten finden sich in einer Tabelle.

Dennoch hat dieses Vorgehen entscheidende Nachteile: Mit solchen n-Gram Modellen sind Entscheidungen auf Information aus einem sehr kleinen Fenster beschränkt. Die statistische Unabhängigkeitsannahme, daß das Vorkommen eines Wortes nur von den $n-1$ Vorworten abhängt, ist unzulässig und kann Ergebnisse stark verfälschen. Die Fensterbreite n der n-Gramme zu vergrößern ist schwierig: Die Größe des notwendigen Trainingskorpus wächst exponentiell mit n.

Auch ignoriert die Methode vollkommen grammatische Aspekte. In dem Satz

4 `Er sagte, daß er mit seiner Freundin nach Rom fahren will.`

kann als letztes Wort kaum etwas anderes als ein Modalverb stehen. Zudem ist die Verbform durch das Subjekt festgelegt: `wollte` oder `willst` sind ausgeschlossen. Die entsprechende Information aus dem Subjekt `er` ist jedoch sieben Wörter entfernt.

3.3 Probabilistische kontextfreie Grammatiken (PCFG)

N-Gramme über Wortfolgen haben keinerlei grammatische Plausibilität. Neben dem eben angesprochenen Versagen bei weit entfernten Bezügen, liefert ein solches Vorgehen keinerlei Hinweise für die nachfolgende Semantikkomponente. Hierzu ist eine Strukturierung des Satzes notwendig, wie sie von einer kontextfreien Grammatik geliefert wird.

Bei *probabilistischen kontextfreien Grammatiken* (*PCFG*[1]) werden Anwendungswahrscheinlichkeiten für die Ableitungsregeln ermittelt. Dies ist in Abbildung 3.2 illustriert. Mit Hilfe

[1]PCFG = Probabilistic Context-Free Grammar

dieser Werte ist Disambiguierung möglich: Bei verschiedenen Parses kann der wahrscheinlichste ermittelt werden.

Abbildung 3.2: Beispielhafter Ausschnitt aus einer PCFG

Die angegeben Zahlen sind bedingte Wahrscheinlichkeiten, wie wahrscheinlich es ist, daß das nichtterminale Symbol `NOMINALPHRASE` zu einer der rechten Seiten abgeleitet wird. Die Wahrscheinlichkeitswerte sind willkürlich gewählt und summieren sich zu 1.

Wahrscheinlichkeit	Ableitungsregel
0.47	NOMINALPHRASE → ARTIKEL NOMEN
0.17	NOMINALPHRASE → NOMEN
0.13	NOMINALPHRASE → PRONOMEN
0.09	NOMINALPHRASE → NOMINALPHRASE PRÄPOSITIONALPHRASE
0.06	NOMINALPHRASE → NAME
0.06	NOMINALPHRASE → ADVERB NOMINALPHRASE
0.02	NOMINALPHRASE → VERB

Die Wahrscheinlichkeit einer Regelanwendung hängt nicht nur von der grammatischen Kategorie ab. Als zusätzliche Information kann die Regel herangezogen werden, die zur Ableitung der Kategorie geführt hat. Nun kann die Ableitungswahrscheinlichkeit von einer beliebigen Zahl an vorher angewendeten Regeln abhängig gemacht werden. Man spricht dann auch hier von einem n-Gram Modell: einem n-Gram Modell über Ableitungsregeln.

Die Ableitungswahrscheinlichkeiten können mit Hilfe des *Inside-Outside Algorithmus* anhand von Beispielsätzen gewonnen werden. Für eine detailierte Beschreibung siehe [Web95]. Die notwendige Übertragung dieses Vorgehens auf Unifikationsgrammatiken, befindet sich jedoch noch in den Anfängen. Siehe [Web95] für eine Übersicht an möglichen Ansätzen.

PCFG leiden jedoch ebenfalls unter falschen statistischen Unabhängigkeitsannahmen, wie Magerman in seiner Arbeit betont [Mag94, Seite 19]. Dies gilt insbesondere für das Deutsche. So hängt die Wahrscheinlichkeit, ob in einem Satz eine Nominalphrase zu einem Subjekt abgeleitet werden kann nicht notwendigerweise von einer festen Anzahl an vorher angewendeten Regeln ab. Entscheidend ist, ob sich an anderer Stelle im Satz ein Subjekt befindet.

3.4 Grundlagen des Ansatzes zur Grammatikerweiterung

In seiner Dissertation führte Miles Osborne [Osb94] ein Lernverfahren zur Erweiterung von Grammatiken für das Englische ein. Dieses Verfahren ist Grundlage dieser Arbeit, das ein ähnliches Verfahren für das Deutsche vorstellt. Im folgenden wird Osbornes Verfahren kurz skizziert und Unterschiede zu dem in dieser Arbeit entwickelten Verfahren aufgezeigt.

Das Verfahren beginnt mit einer Grundgrammatik, die die grundlegendsten Sprachkonstrukte bereits abdeckt. Mit dieser wird nun versucht ein Traingskorpus zu parsen. Dabei wird natürlich ein Teil der Sätze bereits erfolgreich analysiert. Für die übrigen Sätzen sollen Regeln gefunden, so daß auch sie erfolgreich geparst werden können.

Für jeden Satz wird nun versucht neue Regeln zu finden, mit denen er schließlich erfolgreich geparst werden kann. Kandidaten für Regeln werden zunächst aufgrund der Chart nach dem mißglückten Parsen konstruiert. Da eine Vielzahl an Regeln vorgeschlagen wird, folgen zwei Schritte in denen Kandidaten aussortiert werden: ein modellbasierter Schritt und ein datengesteuerter Schritt.

Der sogenannte modellbasierte *Kritiker* beurteilt mögliche Regelkandidaten nach linguistischen Gesichtspunkten. Grundlage dieser Beurteilung ist ein *Grammatikmodell*, das hauptsächlich auf dem *head feature mechanism* aus der *x-bar theory* basiert (näheres dazu siehe [Win90]). Aber auch andere grammatische Theorien fanden darin Einfluß.

Regeln, die vom modellbasierten Kritiker akzeptiert wurden, werden nun von der datengesteuerte Komponente des Verfahrens beurteilt. Diese ist ein statistischer Algorithmus, mit dessen Hilfe vorgeschlagenen Regeln nach ihrer Anwendungshäufigkeit beurteilt werden und Regeln, die am häufigsten angewendet werden, am höchsten bewertet werden. Regel, deren Bewertung einen bestimmten Grenzwert übersteigen werden schließlich in die Grammatik übernommen.

Die mit diesen Regeln erweiterte Grammatik deckt mehr sprachliche Konstrukte ab, als die Grundgrammatik. Letztendliches Ziel ist es mit einer ausreichenden Menge an Trainingssätzen eine vollständige Grammatik zu erzeugen.

Übertragung des Verfahrens

Für die in dieser Arbeit vorgenommene Übertragung in das Deutsche mußten erhebliche Änderungen vorgenommen werden. Das einfache Grammatikmodell Osbornes war so nicht auf das Deutsche anwendbar. Es mußte eine neue modellbasierte Komponente des Lernverfahrens entwickelt werden. Auch der datengesteuerte Lernschritt wurde so nicht übernommen.

Ein Problem in Osbornes Arbeit wurde dabei angegangen: Osborne generiert mit seinem Verfahren sehr viele neue Regeln und hat zur Folge mit einer großen Übergenerierung zu kämpfen, d.h. es werden viele ungrammatische Wortfolgen schließlich von der entstandenen Grammatik akzeptiert. Auch gelingt es ihm nicht mehr, Sätze effizient vollständig zu parsen. Das Parsing wird jeweils nach der Erzeugung von 10 Parsen abgebrochen, unabhängig davon, ob anschließend sinnvollere Parses erzeugt werden könnten.

In dieser Arbeit wird die modellbasierte Komponente des Verfahrens stärker betont. Dadurch wurde stärker versucht, nur linguistisch plausible Regeln zu erzeugen und somit die Zahl an erzeugten Parses pro Satz niedrig zu halten. In dem folgenden Kapitel wird das Verfahren noch einmal ausführlich motiviert und im Überblick vorgestellt.

Kapitel 4

Überblick

Dieses Kapitel gibt einen ausführlichen Überblick über diese Arbeit. Das behandelte Problem wird motiviert, Kriterien für eine Lösung angegeben und das Verfahren skizziert.

4.1 Problem

Mit den im vorletzten Kapitel vorgestellten Formalismen ist es prinzipiell möglich, eine Grammatik für natürliche Sprachen wie das Deutsche schreiben. In der Praxis zeigt sich jedoch, daß dies ein sehr schwieriges Unterfangen ist. So setzt die manuelle Erstellung einer Grammatik nicht nur gute linguistische Kenntnisse voraus. Um alle Eigenheiten der Sprache zu fassen, bedarf es unzählige Regeln. Die schiere Größe einer solchen Grammatik kompliziert ihre Beherrschung erheblich.

Eine der komplexesten Grammatiken für natürliche Sprachen, die Alvey Natural Language Tools Grammar für das Englische, umfaßt ungefähr eintausend Unifikationsregeln [GBCB92]. Trotzalledem versagt diese bei 3,12 Prozent der Nominalphrasen in einer Untersuchung von Taylor et al. [TGB89] (zitiert nach [Osb94]).

Miles Osborne [Osb94, Seite 5f] gibt theoretische und logistische Gründe dafür an, daß alle handgeschriebene Grammatiken *untergenerieren*, also die natürliche Sprache nicht vollständig abdecken. Er zitiert auch aus einer Arbeit von Souter und O'Donoghue:

> *We are, then, somewhat cautious as to the ultimate value of manually building a large rule-based grammar, as there will always be some new sentence which contains structures not catered for in the grammar, so any parser using such a grammar will hardly be robust.* [1]

4.2 Lösung

Zielsetzung dieser Arbeit ist es, ein Verfahren zu entwickeln, das Grammatikregeln automatisch generiert. Ausgehend von einer *Grundgrammatik* sollen neue Regeln anhand von Beispielsätzen gelernt werden.

[1] Übersetzung: „Wir sind daher etwas vorsichtig was den letztendlichen Nutzen der manuellen Erstellung einer großen regelbasierten Grammatik angeht, weil es immer einige neue Sätze geben wird, die Strukturen enthält, für die in der Grammatik nicht vorgesorgt wurde. Deshalb wird jeder Parser, der solch eine Grammatik benutzt, kaum robust sein." [SO90]

Wenn ein Beispielsatz von der Grammatik nicht geparst werden kann, setzt ein Lernverfahren ein, das neue Grammatikregeln vorschlägt. Dies geschieht, bis der Satz mit Hilfe der Regeln geparst werden kann. Anschließend werden die neuen Regeln bewertet und die besten in die Grammatik übernommen.

Durch ein ausreichendes Korpus an Beispielsätzen wird durch das Verfahren eine bessere Grammatik automatisch generiert.

4.3 Kriterien

Um die Qualität der generierten Grammatik bewerten zu können, wollen wir uns an drei Kriterien orientieren: Untergenerierung, Übergenerierung und Plausibilität.

Untergenerierung

Das Problem der Untergenerierung ist die Hauptmotivation für diese Arbeit. Ein Grammatik soll die natürliche Sprache vollständig beschreiben. Gibt es wohlgeformte Sätze, die von der Grammatik nicht abgeleitet werden können, ist dies Untergenerierung.

Übergenerierung

Umgekehrt soll die Grammatik aber auch keine Wortketten ableiten können, die keine gültigen Sätze darstellen. Dies ist beispielsweise von großer Bedeutung, wenn die Grammatik als Filter für die Worterkennung bei gesprochener Sprache verwendet werden soll.

Plausibilität

Schließlich sollte der Parsebaum Aufschluß über die Satzstruktur liefern. Aufgabe des syntaktischen Sprachverarbeitungsschrittes ist es nicht nur, richtige von falschen Sätzen zu unterscheiden, sondern auch Sätze so zu gliedern, daß sie in einem semantischen Folgeschritt weiterverarbeitet werden können. So sollten beispielsweise Nominalphrasen oder Präpositionalphrasen als abgeschlossene Bestandteile ersichtlich sein.

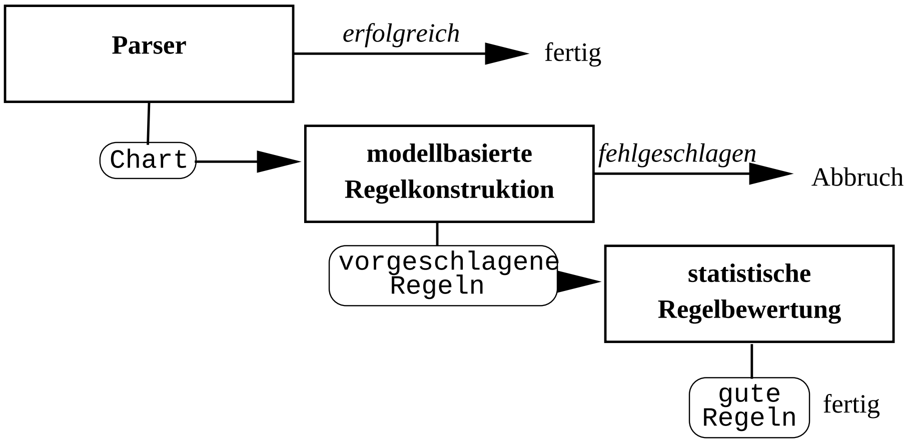
Effizienz

Ein zusätzliches Kriterium wäre noch Effizienz. Die Grammatik sollte schnelles Parsen bzw. Zurückweisen von Eingaben ermöglichen. Da dies jedoch erheblich vom verwendeten Parsingverfahren abhängt, betrachten wir dieses Kriterium nicht weiter.

Die ersten drei Kriterien sind aus Osbornes Arbeit [Osb94] übernommen. Es sind, wenn auch unter anderen Namen, anerkannte Qualitätsmaßstäbe für Grammatiken. Allen nennt sie beispielsweise generality, selectivity und understandability (Allgemeinheit, Abgrenzbarkeit und Verständlichkeit) [All95, Seite 44].

Die Bedeutung dieser Kriterien hängt von der Verwendung der Grammatik ab. Untergenerierung und Plausibilität sind von den Ansprüchen an die Strukturierung des Satzes für die Weiterverarbeitung in einer Semantikkomponente motiviert. Die Verwendung von einer Grammatik

Abbildung 4.1: Ablauf des Verfahrens



zur Unterstützung eines Spracherkenners setzt andere Schwerpunkte: Hier sind die Wahrscheinlichkeiten und das Ausschließen von Folgewörtern von größerer Bedeutung, also die Kriterien Untergenerierung und Übergenerierung.

4.4 Verfahren

Das Verfahren besteht aus drei Komponenten:

- ein bottom up Chartparser,
- modellbasiertes Lernen und
- datengesteuertes (oder statistisches) Lernen.

Das Verfahren ist in Abbildung 4.1 bildlich dargestellt. Der Parser erzeugt die Chart und entscheidet, ob ein Satz bereits von der Grammatik abgedeckt wird. Das *modellbasierte Lernen* benutzt die Kanten in der Chart und schlägt neue Regeln aufgrund von grammatikalischen Prinzipien vor. Das *statistische Lernen* bewertet die vorgeschlagenen Regeln und verwirft schlechte Regeln.

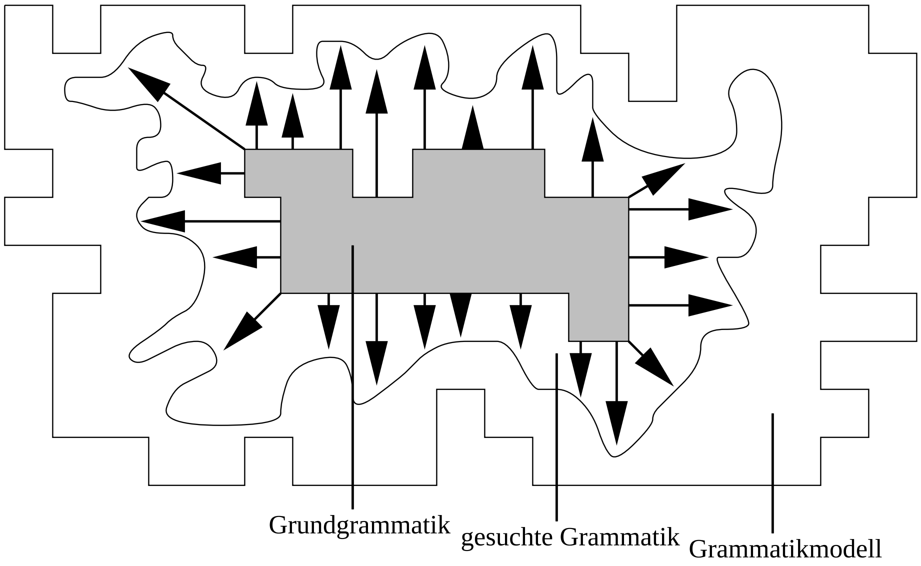
Der bottom up Parser wurde schon ausführlich im letzten Kapitel vorgestellt. In diesem Abschnitt soll nun das modellbasierte und datengesteuerte Lernen motiviert und eingeführt werden.

Modellbasiertes Lernen

Die Aufgabe des modellbasierten Lernes ist es, neue Regeln vorzuschlagen, die es erlauben einen vormals unparsebaren Satz zu verarbeiten.

Das Lernen basiert auf einem *Grammatikmodell*, das einen Rahmen für mögliche Regeln liefert. Dieses Grammatikmodell muß so allgemein sein, daß es die natürliche Sprache vollständig

Abbildung 4.2: Rolle von Grundgrammatik und Grammatikmodell



abdeckt. In anderen Worten: Nähme man alle Regeln, die das Grammatikmodell erlaubt, müssen sich damit alle wohlgeformten Sätze des Deutschen parsen lassen. All diese Regeln würden enorm übergenerieren, sie sind also als Grammatik nicht brauchbar.

Abbildung 4.2 illustriert die Rolle von Grammatikmodell und Ausgangsgrammatik bei der Generierung einer Grammatik für die natürliche Sprache. Die Grundgrammatik soll nach Bedarf erweitert werden. Die äußere Grenze ist die Menge aller möglichen Regeln, die das Grammatikmodell erlaubt.

Um auf die Kriterien im letzten Abschnitt zurückzukommen: Die Regelkonstruktion im modellbasierte Lernen geht gegen Untergenerierung der Grammatik an. Das Grammatikmodell wacht dabei über die Plausibilität der Regeln.

Statistisches Lernen

Aufgabe des datengesteuerten (oder statistischen) Lernens ist es, die vom modellbasierten Lernen vorgeschlagenen Regeln zu bewerten. Es abstrahiert dabei vom Inhalt der Regeln und betrachtet ihre Anwendungshäufigkeit. Je häufiger eine Regel angewendet wird, desto besser wird sie bewertet. Mit diesem Ansatz ist es möglich, Regelmäßigkeiten zu erkennen: Man kann allgemeine sprachliche Phänomene von eher obskuren Interpretationen unterscheiden, die nur auf wenige Sätze anwendbar sind.

In dem hier vorgestellten Lernverfahren wird das statistische Lernen vor allem angewendet, um die Zahl an neuen Regeln klein zu halten. Damit gibt es weniger Übergenerierung und die Effizienz der erzeugten Grammatik wird erhöht.

Verbindung beider Lernstrategien

Die Verbindung aus modellbasiertem und statistischem Lernen nutzt die Vorzüge der beiden Ansätze aus. Das linguistische motivierte modellbasierte Lernen erzeugt nur plausible neue Grammatikregeln und das statistische Lernen hält die Anzahl an neuen Regeln klein.

Das hier vorgestellte Verfahren betont im Gegensatz zu Osbornes Arbeit [Osb94] das modellbasierte Lernen. Das Leitmotiv des Verfahrens ist es, (fast) nur grammatikalisch plausible Regeln zu erzeugen. Der Nutzen der Verwendung einer Grammatik für die Sprachverarbeitung ist ja vor allem die Strukturanalyse des Satzes für die Semantik. Dabei ist die grammatikalische Plausibilität der Ergebnisse von entscheidender Bedeutung. Dagegen haben sich für das bloße Überprüfen von der Gültigkeit von Wortfolgen für die Spracherkennung Ansätze wie n-Gram-Modelle als erfolgreicher herausgestellt.

4.5 Korpus

Als Korpus wurden Sätze des experimentellen Zugauskunftssystems EVAR vom Institut für mathematische Maschinen und Datenverarbeitung (IMMD) der Universität Erlangen verwendet. Es handelt sich dabei um Aufzeichnungen gesprochener Sprache. Zum Beispiel:

1 `Grüß Gott, ich möchte nach Paris.`
2 `Ich will am Samstag nach Hamburg fahren.`
3 `Können Sie mir einen Zug nach Nürnberg für Samstag Mittag geben?`
4 `Gibt es auch eine Verbindung nach 8 Uhr abends?`

Das Korpus ist im Anhang D abgedruckt.

Kapitel 5

Konstruktion neuer Regeln

Dieses Kapitel beschreibt, wie neue Grammatikregeln vorgeschlagen werden. Dies geschieht durch Lernen anhand eines Grammatikmodells, das sicherstellen soll, daß neue Regeln grammatisch sinnvoll sind und sich im Einklang mit den vorhandenen Regeln in der Grammatik befinden.

Die Komplexität des Grammatikmodells steht dabei im Konflikt von zwei Zielsetzungen. Zum einen sollte es offen genug sein, um Regeln für alle unvorhersehbaren Grammatikkonstrukte zu ermöglichen. Zum anderen sollte es beschränkend genug sein, damit die Grammatik nicht durch eine Unzahl unplausibler Regeln verwildert.

5.1 Ablauf des Verfahrens

Zusammengefaßt nimmt die modellbasierte Regelkonstruktion folgenden Ablauf:

1. Parsen der Äußerung
2. Fertig, wenn erfolgreich geparst
3. Konstruktion aller möglichen Regeln anhand der Charts
4. Abbruch, wenn keine neuen Regeln vorgeschlagen
5. Vorläufige Aufnahme der neuen Regeln in die Grammatik
6. Zurück zum Schritt 1

Es werden also in einer Iteration der Regelkonstruktion alle Regeln vorgeschlagen, die aufgrund der Kanten in der Chart möglich und von dem Grammatikmodell erlaubt sind. Diese Regeln dürfen nicht schon in der Grammatik enthalten sein. Auch gilt eine Regel nicht als neu, wenn in der Grammatik schon eine allgemeinere Regel enthalten ist.

Das Verfahren hält, weil die Anzahl möglicher Regeln endlich ist. Es kommt also entweder zum Abbruch, weil keine neuen Regeln mehr vorgeschlagen werden können oder zum erfolgreichen Abschluß, wenn der Satz mit Hilfe der neuen Regeln vollständig geparst werden kann.

5.2 Ausgangssituation

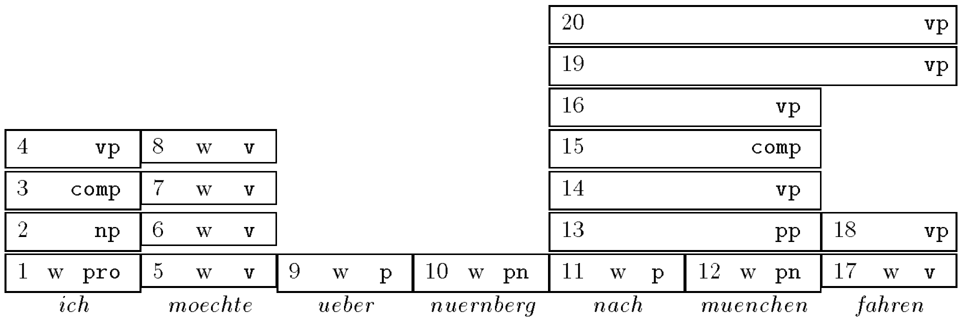
Das modellbasierte Lernen wird aktiviert, wenn eine Äußerung nicht geparst werden konnte. Betrachten wir folgenden Satz:

1 `Ich möchte über Nürnberg nach München fahren.`

Angenommen, der Satz konnte nicht geparst werden, weil in der Grammatik die Präpositionalphrase `über Nürnberg` nicht vorgesehen ist.[1] Die vom Parser erzeugte Chart ist in Abbildung 5.1 dargestellt. Die Kanten in der Chart bilden nun die Grundlage für Regelvorschläge.

Abbildung 5.1: Chart nach mißglücktem Parsen

Dieses Beispiel ist eine Chart, wie er mit einem bottom up Parser erzeugt worden sein könnte. Jede Kante ist durch einen Kasten dargestellt, der die Kantennummer und die syntaktische Kategorie beinhaltet. Kanten mit einem ,,w" sind Lexikoneinträge: so hat die Wortform *moechte* mehrere Einträge: es ist sowohl ein Hilfs- als auch Vollverb und es kann in der ersten oder dritten Person Singular konjugiert sein. Die Präpositionalphrase *nach muenchen* kann sowohl als Adjunkt oder als Argument in die Verbalphrase aufgenommen werden. Daher gibt es zwei `vp`-Kanten (Nummer 14 und 16), die diesen Satzteil überspannen.



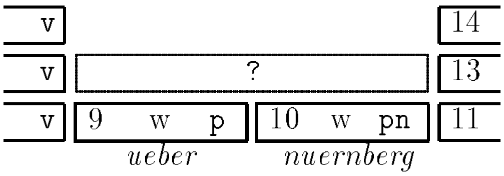
Für jeweils benachbarte Kanten in der Chart wird versucht, Regeln zu konstruieren. So fehlt für den Beispielsatz eine Regel, um die beiden Kanten 9 und 10 zusammenzufassen. Diese Regel muß nun gelernt werden. Bei erneutem Parsen lassen sich durch Anwendung der neuen Regeln neue Kanten erzeugen. Damit läßt sich die Eingabe eventuell schon vollständig parsen, ansonsten wird der Vorgang wiederholt.

Die Regelkonstruktion nimmt also die Kanten in der Chart als Quelle für neue Regeln. Da zum erfolgreichen Parses des Satzes zusätzlich Kanten zusammengefaßt werden müssen, müssen Regeln gefunden, die genau dies ermöglichen. Beispielsweise sollten die Kanten 9 und 10 zusammengefaßt werden, wie es in Abbildung 5.2 angedeutet ist. Dazu muß es eine Regel geben, die die Präposition `über` und den Namen `Nürnberg` ableitet, also grob folgende Form hat:

$$? \rightarrow \texttt{PRAEPOSITION NAME}$$

[1] Dieses Beispiel stimmt nicht mit der tatsächlichen in dieser Arbeit verwendeten Grundgrammatik überein (beschrieben in Anhang A). Es wurde aus didaktischen Gründen gewählt, weil sich daran das modellbasierte Lernen gut darstellen läßt.

Abbildung 5.2: Zusammenfassen zweier Kanten



(Die Konstituenten sind hier als atomare Begriffe geschrieben, obwohl sie in der dieser Arbeit zugrundeliegenden Grammatik Merkmalsstrukturen sind. Im folgenden sind jedoch zunächst nur die Kategorien von Bedeutung, weswegen wir zunächst diese vereinfachte Schreibweise gebrauchen.)

Für den Regelkonstruktor gibt es nun zwei fundamentale Fragestellungen:

- Welche Arten von Kanten können zusammengefaßt werden?
- Wie sieht die linke Regelseite aus?

Für das obige Beispiel (`über Nürnberg`) liegen die Antworten auf der Hand: Ja, sie können zusammengefaßt werden und auf der linken Regelseite muß eine Konstituente von der Kategorie `pp` (also eine Präpositionalphrase) stehen. In dem Lernverfahren müssen diese Fragen vom Grammatikmodell beantwortet werden.

5.3 Beschränkungen für Regeln

Wir sprechen zunächst die erste der genannten Fragen an: Welche Arten von Kanten können zusammengefaßt werden? Das Grammatikmodell muß hierzu sinnvolle Beschränkungen festlegen. Dies geschieht zum einen, um den Berechnungsaufwand nicht explodieren zu lassen, zum anderen um nur grammatisch stimmige Regeln zu lernen.

Die erste schwerwiegende Einschränkung des Grammatikmodells ist es, daß nur unäre und binäre Regeln erlaubt werden, d.h. auf der rechten Regelseite nur eine oder zwei Konstituenten stehen dürfen. Dadurch müssen bei der Zusammenfassung von Kanten auch nur einzelne Kanten oder Kantenpaare betrachtet werden.

In der Chart aus Abbildung 5.1 gibt es 20 Einzelkanten und 39 Kantenpaare. Ohne Einschränkungen gibt es also 59 mögliche rechte Regelseiten. Nun gibt es für jede diese noch eine Vielzahl möglicher linker Regelseiten. Ebenfalls kann die Regel verschieden allgemein gefaßt werden, was die Übergabe oder Übereinstimmung von Merkmalen angeht. So sind selbst mit der — durchaus erheblichen — Einschränkung auf nur unäre und binäre Regeln unzählige neue Regeln möglich.

Osborne [Osb94] wichtigste Beschränkungen sind *linear precedence rules (LP-Regeln)* und das *head feature convention* aus der *x-bar theory*, um die Menge an möglichen Regeln zu beschränken. LP-Regeln beschränken die Aufeinanderfolge von Konstituenten in Regeln. Beispielsweise gibt es keine sinnvolle Regel mit der rechten Seite `VERB ARTIKEL`.

Die head feature convention aus der X-Bar Theorie besagt, daß der Konstituent auf der linken Seite einen der Konstituenten auf der rechten Seite, der als *Regelkopf* bezeichnet wird, *dominiert*. Dominieren heißt dabei, daß bestimmte Merkmale übereinstimmen müssen. Mehr zum Thema X-Bar Theorie findet sich beispielsweise in [Win90]. In den folgenden Beispielen ist der Kopf unterstrichen:

$$\begin{array}{rcl}
\texttt{NOMINALPHRASE} & \rightarrow & \texttt{ARTIKEL}\ \underline{\texttt{NOMEN}} \\
\texttt{VERBALPHRASE} & \rightarrow & \underline{\texttt{VERB}}\ \texttt{VERBALPHRASE} \\
\texttt{NOMEN} & \rightarrow & \texttt{ADJEKTIV}\ \underline{\texttt{NOMEN}} \\
\texttt{SATZ} & \rightarrow & \underline{\texttt{VERBALPHRASE}}
\end{array}$$

Diese Einschränkungen sind sehr grob. Sie erlauben einerseits grammatisch unsinnige Regeln, wie beispielsweise die folgende.

$$\texttt{ADJEKTIV} \rightarrow \underline{\texttt{ADJEKTIV}}\ \texttt{NOMEN}$$

Sie ist möglich, weil LP-Rules die Aufeinanderfolge von `ADJEKTIV` und `NOMEN` erlauben und nach der head feature convention ein `ADJEKTIV` ein `ADJEKTIV` dominieren darf: Andererseits verbieten diese Einschränkungen Regeln, wie sie beispielsweise für das Satzende benötigt werden:

$$\begin{array}{rcl}
\texttt{VERBALPHRASE} & \rightarrow & \texttt{PRAEPOSITIONALPHRASE} \\
\texttt{VERBALPHRASE} & \rightarrow & \texttt{NOMINALPHRASE} \\
\texttt{VERBALPHRASE} & \rightarrow & \texttt{ADJEKTIV}
\end{array}$$

Für diese Arbeit wurde daher ein verfeinertes Schema entwickelt, das eher den Ansprüchen des Deutschen entspricht.

5.4 Regeltypen

Bevor wir zur Beschreibung des Grammatikmodells kommen, müssen wir jedoch erst noch einmal ausholen und mit einer Analyse beginnen, welche Typen von Regeln in der Grundgrammatik (Anhang A) vorkommen und als sinnvolle neue Regeln zu erwarten sind.

Die Regeln in der Grundgrammatik lassen sich in vier Typen aufteilen, die wir im folgenden zunächst beschreiben wollen.

Argumentregeln sind vor allem notwendig, wenn ein Argument in eine Phrase aufgenommen werden soll. Beispiele sind:

$$\begin{array}{rcl}
\texttt{VERBALPHRASE} & \rightarrow & \texttt{SUBJEKT}\ \texttt{VERBALPHRASE} \\
\texttt{NOMEN} & \rightarrow & \texttt{ADJEKTIV}\ \texttt{NOMEN}
\end{array}$$

Der Konstituent auf der linken Seite stimmt mit einem der Konstituenten auf der rechten Seite völlig überein. Mit dem anderen stimmt er in einigen Merkmalen überein: im ersten Beispiel wäre dies Numerus und Person, im zweiten Numerus, Genus und Kasus.

Adjunktregeln ähneln stark den Argumentregeln mit der Ausnahme, daß keinerlei Merkmale übereinstimmen müssen. Beispiel:

$$\texttt{VERBALPHRASE} \rightarrow \texttt{ADVERB}\ \texttt{VERBALPHRASE}$$

Kompositionsregeln sind notwendig, wenn zwei Konstituenten zu einer zusammengefaßt werden. Beispiele:

$$\begin{aligned} \texttt{PRAEPOSITIONALPHRASE} &\rightarrow \texttt{PRAEPOSITION NOMINALPHRASE} \\ \texttt{NEBENSATZ} &\rightarrow \texttt{KONDITIONAL VERBALPHRASE} \\ \texttt{VERB(INF)} &\rightarrow \texttt{VERB(PART) VERB(INF)} \end{aligned}$$

Bei all diesen Beispielen werden Merkmale aus den Konstituenten auf der rechten Regelseite in die Konstituenten auf der linken Regelseite übernommen, die meist einer Kategorie angehört, die mit keiner der beiden Konstituenten auf der rechten Regelseite übereinstimmt. Für diesen Typ von Regeln kann kein allgemeines Schema angegeben werden. So wird das letzte Beispiel für Konstrukte wie `angegeben werden` benötigt, wo die Informationen über Anforderungen (also das Merkmal VAGR) aus der ersten Konstituente, die Informationen über grammatische Verwendung (wie das Merkmal AGR) aus der zweiten übernommen werden.

Überführungsregeln sind alle unären Regeln. Hier gibt es viele Spezialfälle, wie das erste der folgenden Beispiele:

$$\begin{aligned} \texttt{SATZ} &\rightarrow \texttt{VERBALPHRASE} \\ \texttt{VERBALPHRASE} &\rightarrow \texttt{PRAEPOSITIONALPHRASE} \end{aligned}$$

Eine große Gruppe dieses Typs sind Satzenderegeln, wie sie auf Seite 32 angegeben sind. Dies sind im Prinzip Argument- oder Adjunktregeln, denen die übereinstimmende Konstituente auf der rechten Seite fehlt. So gleicht das zweite Beispiel der Adjunktregel

$$\texttt{VERBALPHRASE} \rightarrow \texttt{PRAEPOSITIONALPHRASE VERBALPHRASE}$$

außer daß die Konstituente `VERBALPHRASE` auf der rechten Seite fehlt.

In dem Verfahren von Miles Osborne [Osb94] werden ausschließlich Adjunktregeln gelernt. In dieser Arbeit sollen hingegen Regeln aller vier Regeltypen gelernt werden.

5.5 Lernen von Kompositions- und Überführungsregeln

Die meisten Kompositions- und Überführungsregeln in der Grundgrammtik sind elementar für die Struktur der Grammatik. Es sind eher technische Regeln wie beipielsweise

$$\texttt{SUBJEKT} \rightarrow \texttt{NOMINALPHRASE}$$

(Diese Regel übernimmt Merkmale aus der Nominalphrasen-Notation in die Verbalphrasen-Notation.) Es wäre sehr schwierig diese Regeln durch ein automatisches Verfahren zu lernen. Und dies wäre auch nicht sehr sinnvoll: diese Regeln sorgen für die linguistische Plausibilität der Grammatik, was nicht allein durch Syntax begründet ist.

Die Fälle von Kompositions- und Überführungsregeln, die wir lernen wollen, lassen sich durch wenige übergeneralisierende *Musterregeln* abdecken. Eine typisches Muster für Kompositionsregeln ist beispielsweise

$$\begin{bmatrix} \text{CAT: } pp \\ \text{SEMCAT: } (\langle \mathit{1} \rangle, \langle \mathit{2} \rangle) \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT: } p \\ \text{SEMCAT: } \langle \mathit{1} \rangle \end{bmatrix} \begin{bmatrix} \text{CAT: } pn \\ \text{SEMCAT: } \langle \mathit{2} \rangle \end{bmatrix}$$

Dieses Muster erlaubt Konstrukte wie `nach Erlangen`, `von Nürnberg` oder `mit Fritz`. Es deckt aber auch unerwünschte Konstrukte wie `am Berlin` ab. Muster übergeneralisieren, sie sind daher nicht Teil der Grundgrammatik, sondern des Grammatikmodells.

Beispiel

Wir wollen nun anhand eines Beispiels exemplarisch zeigen, wie Muster vom Lernverfahren benutzt werden, um neue Regeln vorzuschlagen. Der Beispielsatz vom Beginn dieses Kapitels

1 `Ich möchte über Nürnberg nach München fahren.`

konnte nicht geparst werden, weil für das Konstrukt `über Nürnberg` keine Regel in der Grundgrammatik existiert. In der Chart (siehe Abbildung 5.1 auf Seite 30) konnten daher die Kanten 9 und 10 nicht zusammengefaßt werden.

Das Lernverfahren versucht nun eine Musterregel zu finden, die diese beiden Kanten zusammenfaßt und findet sie in dem obigen Muster. Durch Besetzung der Variablen mit den Merkmalseinträgen aus den Kanten wird folgende Regel vorgeschlagen:

$$\begin{bmatrix} \text{CAT}: pp \\ \text{SEMCAT}: (ueber, stadt) \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT}: p \\ \text{SEMCAT}: ueber \end{bmatrix} \begin{bmatrix} \text{CAT}: pn \\ \text{SEMCAT}: stadt \end{bmatrix}$$

Mit dieser Regel läßt sich nun die Präpositionalphrase `über Nürnberg` parsen.

5.6 Lernen von Argument- und Adjunktregeln

Grundsätzlich ist eine Unzahl an Argument- und Adjunktregeln vorstellbar. Wir wollen nun einschränken, welche Kategorien durch Adjunkt- und Argumentregeln zusammengefaßt werden dürfen.

So wissen wir, daß beispielsweise eine Regel mit den Kategorien

`ADJEKTIV` $\rightarrow$ `ADJEKTIV NOMEN`

keine sinnvolle Regel darstellt. Miles Osborne [Osb94] realisiert diese Einschränkung mit einer Reihe von LP-Regeln (siehe Seite 31). Das in dieser Arbeit vorgestellte Verfahren benutzt die Tabellen 5.1 für Adjunktregeln und 5.2 für Argumentregeln.

Die Tabellen basieren auf den grammatischen Kategorien. Sie wurden von Hand erstellt, wobei linguistisches Wissen und Kenntnisse über die Grundgrammatik einflossen. Dies ist im Vergleich zu Osborne ein weniger elegantes und damit schwierigeres Verfahren, das auch willkürlicher und fehleranfälliger ist. Eine Alternative wäre es, die Tabelle aufgrund von einer kleinen Zahl von verallgemeinerten Prinzipien automatisch zu erstellen.

Adjunkte oder Argumente können entweder links oder rechts an die Kopfkategorie angefügt werden, wie Abbildung 5.3 veranschaulicht. Eine „linke" Adjunktregel in diesem Sprachgebrauch ist eine Regel, die links zur Kopfkategorie die Anfügung eines Adjunktes erlaubt. Die Regel

`VERBALPHRASE` $\rightarrow$ `ADVERB VERBALPHRASE`

Tabelle 5.1: Erlaubte Adjunktregeln

In den Zeilen befinden sich die möglichen Adjunkten, in den Spalten sind die jeweiligen Kopfkategorien aufgeführt. ,,l” und ,,r” stehen für ,,linke” und ,,rechte” Adjunktregeln. So steht der Eintrag in der Zeile `adv` und der Spalte `vp` für Adjunktregeln mit den Kategorien `vp -> adv vp`, wie auch in Abbildung 5.3 dargestellt. Gibt es keinen Eintrag, sind keine Adjunktregeln mit diesen Kategorien möglich.

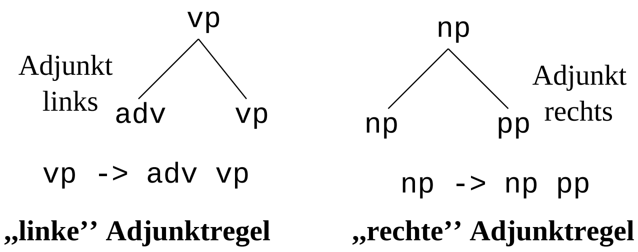
	vp	np	v	pn	adj	t	n
adv	l	r			l	l	
junk	l						
n				lr			
np		r					
pn	r						
pp	l	r					
tp		l					l

Tabelle 5.2: Erlaubte Argumentregeln

In den Zeilen befinden sich die möglichen Argumente, in den Spalten sind die jeweiligen Kopfkategorien aufgeführt. ,,l” und ,,r” stehen für ,,linke” und ,,rechte” Argumentregeln.

	adj	comp
np	l	
vp		l

Abbildung 5.3: Linke und rechte Adjunktregeln



ist also eine linke Adjunktregel: Ein Adverb kann der Verbalphrase vorangestellt werden, es befindet sich links davon. Als „rechte" Adjunkt- bzw. Argumentregeln werden Regeln bezeichnet, die rechts zur Kopfkategorie Adjunkte bzw. Argumente erlauben.

In den Tabellen finden sich die Einträge „l" für links und „r" für rechts. So erlaubt der Eintrag „r" in der Zeile np und der Spalte pp Regeln vom Typ

NOMINALPHRASE $\rightarrow$ NOMINALPHRASE PRAEPOSITIONALPHRASE

Das adjungierte Präpositionalphrase steht also rechts auf der rechten Regelseite, der Regelkopf (die Nominalphrase) wird auf die linke Regelseite übernommen.

Merkmale in Adjunkt- und Argumentregeln

Für die Behandlung von Merkmalen in konstruierten Regeln sei an die Tabellen 2.2 und 2.1 auf Seite 11 erinnert. Dort ist für jede Kategorie angegeben, welche Merkmale sie haben kann.

Da die Kategorien nur bestimmte Eigenschaften haben, können bei deren Merkmalsstrukturen auch nur bestimmte Merkmale besetzt sein. So hat zum Beispiel bei einer Verbalphrase das Merkmal für Kasus (CASE) keinerlei sinnvolle Bedeutung. Die in Tabelle 2.2 aufgeführten Merkmale sind die maximal erlaubten Merkmale. Für die jeweilige Kategorie dürfen keine anderen als die angegebenen Merkmale in einer Regel vorkommen. Daher können sie auch nicht in den Merkmalsstrukturen von Kanten auftauchen.

Die Konstituenten in den Regeln übernehmen jeweils die Merkmalen aus den Kanten. Die Konstituente auf der linken Regelseite übernimmt die Merkmale aus dem Regelkopf der rechten Regelseite.

Alle bisherigen Erläuterungen betreffen sowohl Adjunkt- als auch Argumentregeln. Der Unterschied zwischen beiden ist, daß wir bei Argumentregeln zusätzlich verlangen, daß die beiden Merkmalsstrukturen in Übereinstimmungs- und Anforderungsmerkmalen (siehe Tabelle 2.1) übereinstimmen.

Dies bedeutet für Argumentregeln zum einen eine zusätzliche Einschränkung. Zum anderern kann ein Argument zusätzliche Informationen über Merkmale an die linke Regelseite weiterreichen. Ein Adjunkt hingegen ist immer nur ein bloßes Hinzufügen.

Beispiel

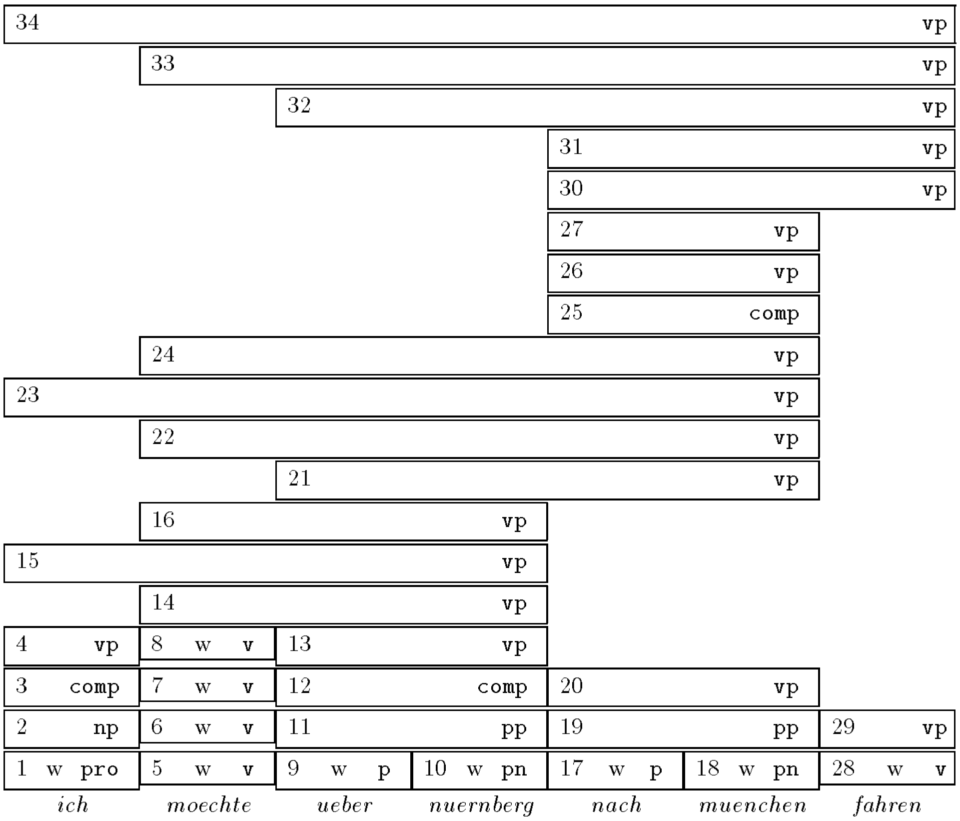
An dieser Stelle wollen wir die Verwendung der eben beschriebenen Einschränkungen anhand eines Beispiels veranschaulichen. Wir betrachten weiterhin den Satz

1 `Ich möchte über Nürnberg nach München fahren.`

Der Satz läßt sich auch mit der neuen Kompositionsregel aus dem vorigen Abschnitt und einer nicht weiter erläutertenweiteren Regel noch nicht vollständig parsen. Nach der ersten Iteration des Verfahrens kam das Verfahren also noch nicht zum Abschluß und ein weiterer Regelkonstruktionsschritt muß folgen. In Abbildung 5.4 ist die Chart nach dem erneutem Parsen dargestellt.

Abbildung 5.4: Chart nach dem zweiten Parseversuch

Wie schon zur ersten Chartabbildung 5.1 auf Seite 30 angemerkt, gibt dieses Beispiel einen realistischen Lauf des Lernverfahrens wieder. Obwohl nur zwei neue Regeln einbezogen werden, hat sich die Zahl der Kanten fast verdoppelt. Meist werden wesentlich mehr neue Regeln vorgeschlagen, was zu einer Vervielfachung der Kanten führt.



Der Satz konnte immer noch nicht geparst werden, weil die Präpositionalphrase `über Nürnberg` (Kante 11) nicht richtig in die Satzstruktur (Kante 30) eingebaut werden konnte.[2] Die entsprechende Adjunktregel fehlt in der Grammatik.

Tabelle 5.1 erlaubt Adjunktregeln, die Präpositionalphrasen links zu Verbalphrasen hinzufügen. Somit können wir die beiden Kanten 11 und 30 als Grundlage für einen neuen Regelvorschlag benutzen (SUBCAT: *inf* in der Verbalphrase bedeutet dabei, daß die Verbalphrase mit einem Verb im Infinitiv endet):

$$\begin{bmatrix} \text{CAT: } vp \\ \text{SUBCAT: } inf \\ \text{POS: } midfield \\ \text{VAGR: [SUBJ: need]} \end{bmatrix} \rightarrow$$

[2]Sie konnte allerdings als ein Präpositionalobjekt (Kante 12) in die Satzstruktur eingebaut werden, was schließlcih zu einer Verbalphrase führte, die den gesamte Satz überspannt (Kante 34). Dieses Präpositionalobjekt wurde jedoch nicht von dem Verb `fahren` angefordert, weswegen diese Verbalphrase nicht als Satz akzeptiert wurde. Mehr zur Grammatik findet sich in Anhang A.

$$\begin{bmatrix} \text{CAT}: \textit{pp} \\ \text{SEMCAT}: (\textit{ueber}, \textit{stadt}) \end{bmatrix} \begin{bmatrix} \text{CAT}: \textit{vp} \\ \text{SUBCAT}: \textit{inf} \\ \text{POS}: \textit{midfield} \\ \text{VAGR}: [\text{SUBJ}: \text{need}] \end{bmatrix}$$

Mit dieser weiteren neuen Regel kann der Satz schließlich vollständig erfolgreich geparst werden.

5.7 Generalisieren von Regeln

Wir haben nun ein Verfahren vorgestellt, mit dem verschiedene Regeltypen gelernt werden können. In diesem abschließenden Abschnitt wird das Problem des Grades der Allgemeinheit von Regeln angesprochen und die Regelkonstruktion für Adjunkt- und Argumentregeln verfeinert.

Das Problem des Grades der Allgemeinheit von Regeln läßt sich sehr gut an der zuletzt dargestellten Regel veranschaulichen. Diese Regel ist zu speziell: Sie erlaubt das Hinzufügen von Präpositionalphrasen vom Typ (*ueber,stadt*) nur zu Infinitivkonstruktionen (SUBCAT) im Mittelfeld (POS), die in Verbindung mit einem Modalverb nur ein Subjekt brauchen und dies noch nicht aufgetreten ist (VAGR, noch kein AGR).

Diese Regel wird sehr selten anwendbar sein. Sinnvoller wäre eine allgemeinere Regel, die Präpositionalphrasen vom Typ (*ueber,stadt*) im Mittelfeld jedes Satzes erlaubt. Sie hätte die Form:

$$\begin{bmatrix} \text{CAT}: \textit{vp} \\ \text{SUBCAT}: \langle 1 \rangle \\ \text{POS}: \textit{midfield} \\ \text{VAGR}: \langle 2 \rangle \\ \text{AGR}: \langle 3 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT}: \textit{pp} \\ \text{SEMCAT}: (\textit{ueber}, \textit{stadt}) \end{bmatrix} \begin{bmatrix} \text{CAT}: \textit{vp} \\ \text{SUBCAT}: \langle 1 \rangle \\ \text{POS}: \textit{midfield} \\ \text{VAGR}: \langle 2 \rangle \\ \text{AGR}: \langle 3 \rangle \end{bmatrix}$$

In dieser Regel sind für die Merkmale SUBCAT und VAGR Variablen anstatt den Werten aus den Kanten eingesetzt. Es ist also lediglich Übereinstimmung verlangt. Ebenso wird Übereinstimmung im Genus und Numerus (AGR) verlangt, da dieses Merkmal im allgemeinen Fall ja mit Werten besetzt sein kann.

Wie können wir nun dem Lernsystem mitteilen, daß es gegebenfalls eine allgemeinere Regel lernen soll? Diese Frage ist sehr grundsätzlich und schwierig: Das Lernsystem hat nur ein Beispiel mit einem neuen Phänomen. Bei diesem Einzelfall ist es überhaupt nicht abzusehen, ob es sich dabei um einen exotischen Sonderfall oder ein allgemeines Prinzip handelt.

Eine Möglichkeit wäre es, diese Entscheidung einem späteren Regelspezialisierer und -generalisierer zu überlassen, der ein großes Korpus von Sätzen auswerten kann. Solch ein Folgeschritt ist jedoch recht kompliziert, zumal es bei einem Parse mit mehreren neuen Regeln kaum mehr nachvollziehbar sein kann, welche Regeln zu allgemein oder zu speziell sind.

In dieser Arbeit wird eine andere, überraschend einfache und wirkungsvolle Methode angewendet: Erstens, alle möglichen Merkmale einer Kategorie müssen in der vorgeschlagenen Regel vorkommen. Dadurch ist in unserem Beispiel das Vorkommen von AGR sichergestellt.

Zweitens, Übereinstimmungs- und Anforderungsmerkmale (siehe Tabelle 2.1 auf Seite 11) werden in der vorgeschlagenen Regel nicht mit Werten aus den Kanten, sondern mit Variablen besetzt. Dadurch müssen diese Werte bei der Regelanwendung übereinstimmen, im Beispiel sind dies die Merkmale SUBCAT, VAGR und AGR. Merkmale zur näheren Bestimmung werden mit den Werten aus den Kanten besetzt.

Wir lösen also das Problem der Allgemeinheit von Regeln im Vertrauen auf die linguistischen Prinzipen des Grammatikmodells: Übereinstimmungsmerkmale charakterisieren Eigenschaften, die in den Komponenten eines Satzteiles gleich sein müssen. Dieses Prinzip läßt sich auf Anforderungsmerkmale übertragen. Merkmale zur näheren Bestimmung geben hingegen spezielle Eigenschaften einer Komponente an, die nicht im Bezug zu benachbarten Komponenten stehen.

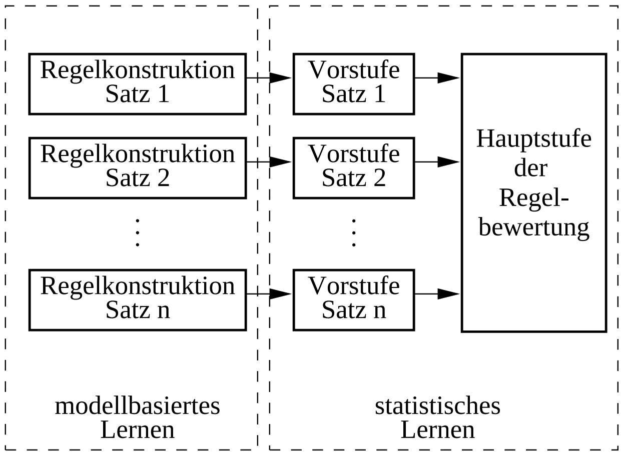
Kapitel 6

Bewertung von Regeln

Das letzte Kapitel erläuterte, wie neue Regeln konstruiert werden. Beim Ablauf des Lernverfahrens werden viele Regeln vorgeschlagen. Aufgabe der statistischen Bewertung ist es nun, die Zahl der neuen Regeln möglichst gering zu halten. Dies geschieht zum einen, um unsinnige Regeln auszusortieren, die noch nicht einmal in erfolreichen Parses der Sätze angewendet werden konnten, für die sie vorgeschlagen wurden. Zum anderen führen neue Regeln zu zusätzlichen Parses, die nicht unbedingt für plausible syntaktische Mehrdeutigkeiten stehen. Für das effiziente Parsen einer Grammatik ist es jedoch unerläßlich, die Zahl der Regeln klein zu halten, da jede zusätzliche Mehrdeutigkeit die Zeitdauer des Parsings vervielfacht.

Hauptzielsetzung der statischen Bewertung ist es, die Zahl an neuen Regeln so klein wie möglich zu halten. Der eigentlichen statistischen Bewertung wurde daher eine Vorstufe vorgeschalten, die dafür sorgt, daß für jeden Satz nur die Regeln zunächst erhalten werden, die notwendig sind. Dieser erste Schritt orientiert sich an der Verwendung der neukonstruierten Regeln beim Parsen des Satzes, für den sie vorgeschlagen wurden. Im zweiten Schritt, der Hauptphase der statistischen Regelbewertung, wird die Verwendung von Regeln im gesamten Korpus betrachtet. Ein graphischer Überblick über das Zusammenspiel von Regelkonstruktion, Vorstufe und Hauptstufe der statistischen Regelbewertung ist in Abbildung 6.1 gegeben.

Abbildung 6.1: Zusammenspiel von modellbasiertem und statistischem Lernen



6.1 Vorstufe: Bewertung im Satzkontext

Zunächst werden die Regeln im Satzkontext bewertet. Um diesen Lernschritt zu veranschaulichen, hier zunächst ein beispielhaftes Ablaufprotokoll:

```
| ?- learn(536).
(ich,suche,zuege,muenchen,nuernberg,etwa,um,mittag)
8 neue Regeln, 1 Parse.
Parse 1 benutzt 508 504 505.
```

Obwohl für den Satz acht neue Regeln vorgeschlagen wurden, wurden bei dem einzigen Parse nur drei davon verwendet. Die übrigen fünf wurden ebenfalls aufgrund von Kanten in der Chart nach dem erfolglosen ersten Parsevesuch vorgeschlagen. Sie führten jedoch zu keinem erfolgreichen Parse. Sie werden daher verworfen. Als *gute Regeln* nach der Bewertung im Satzkontext gelten also nur die drei verwendeten Regeln.

In der Regel werden jedoch nach erfolgreichem Lernen mehrere Parses erzeugt, wie das folgende Beispiel illustriert:

```
| ?- learn(516).
(koennten,sie,mir,eine,verbindung,nach,emmerich,grenzbahnhof,geben)
7 Regeln wurden vorgeschlagen.
Parse 1 benutzt 501 504.
Parse 2 benutzt 504.
Parse 3 benutzt 501 504.
Parse 4 benutzt 504.
Parse 5 benutzt 501 504.
Parse 6 benutzt 504.
```

In diesem Fall wird neben der Regel `504` auch die Regel `501` vorgeschlagen, aber nicht in allen Parses benutzt. Daß es auch ohne diese Regel geht, zeigen die Parses 2, 4 und 6. Wie anfangs des Kapitels dargestellt, ist es das Ziel der statistischen Bewertung, die Zahl der neuen Regeln klein zu halten. Da die Regel `501` zum erfolgreichen Parsen des Satzes nicht benötigt wird, wird sie von dem Lernverfahren an dieser Stelle verworfen.

Der Algorithmus zur Bewertung im Satzkontext arbeitet folgendermaßen:

1. Finde die minimale Anzahl von unterschiedlichen neuen Regeln, die in einem erfolgreichen Parse verwendet wurden.

2. Betrachte nur die Parses mit dieser Zahl an unterschiedlichen neuen Regeln.

3. Übernehme alle neuen Regeln aus diesen Parses als *gute Regeln*.

In der obigen Beschreibung ist von der Zahl der *unterschiedlichen* neuen Regeln gesprochen. Eine neue Regel kann in einem Parse mehrfach angewendet werden, was aber auf den Algorithmus keinen Einfluß haben soll. Wenn beispielsweise ein Parse nur eine neue Regel anwendet, diese jedoch zweimal, ist die Anzahl der Anwendungen unterschiedlicher neuer Regeln dennoch nur 1.

Für den obigen Beispielsatz `516` arbeitet die eben beschriebene Vorstufe der statistischen Bewertung folgendermaßen:

```
Minimale Regelzahl 1.
Gute Regel 504.
1 Regel uebernommen, 3 Parses.
```

Unnötige syntaktische Mehrdeutigkeiten

In dem letzten Beispiel fällt auf, daß drei Parses erzeugt wurden, obwohl keine Mehrdeutigkeit in dem Satz zu erkennen ist. Der Grund für die verschieden Parses liegt in der neu eingeführten Regel: sie erlaubt das Adjungieren von Nomen `Grenzbahnhof` an Namen `Emmerich`. Nun kann jedoch die Wortform `Grenzbahnhof` den Kasus Nominativ, Dativ und Akkusativ haben. Alle drei Worformen tauchen also in der Chart auf. Die eingeführte Regel betrachtet den Kasus des Nomens nicht weiter, weil CASE ein Übereinstimmungsmerkmal ist (siehe 2.1). Die Folge sind drei Parses, die sich nur in der verwendeten Wortform unterscheiden. Während des Parsevorgangs nach der Einführung der neuen Regeln wurden insgesamt 1794 Kanten erzeugt. Diese unnötige Mehrdeutigkeit hat zu diesem erheblichen Aufwand beigetragen.

Diese Fehleranalyse führt zu dem Dilemma jedes Lernverfahrendesigners: Soll er aufgrund dieses gewonnenen Wissens das Lernverfahren so verfeinern, daß es in diesem Spezialfall richtig reagiert? Soll er das Verfahren so manipulieren, daß es genau das lernt, was der Designer schon weiß? Dann kann ja kaum mehr von *Lernen* gesprochen werden, da das Wissen vom Designer ja bereits (wenn auch in indirekter Form) in das Verfahren eingeflossen ist.

Grenzen der Bewertung im Satzkontext

Die Grenzen der statistischen Bewertung im Satzkontext zeigt das folgende Beispiel auf:

```
| ?- learn(542).
(ich,brauche,informationen,ueber,zuege,nach,bamberg)
9 Regeln wurden vorgeschlagen.
Parse 1 benutzt 501.
Parse 2 benutzt 502.

Minimale Regelzahl 1.
Gute Regeln 501 502.
2 Regeln uebernommen, 2 Parses.
```

Die beiden erfolgreichen Parses nach der Regelkonstruktion verwenden jeweils eine andere Regel. Für die statistische Bewertung im Satzkontext können die beiden Regeln jedoch nicht weiter unterschieden werden, sie werden somit beide zunächst übernommen.

Eine genauere Betrachtung dieser Regeln ergibt, daß Regel `501` die Präpositionalphrase `ueber zuege nach bamberg` der Nominalphrase `informationen` zuordnet, während Regel `502` diese Präpositionalphrase als Adjunkt zur Verbalphrase auffaßt. Die semantisch sinnvollere der beiden Regeln ist also die erste. Dieses Wissen ist dem Lernverfahren jedoch nicht zugänglich.

Über ein größeres Korpus hinweg ist es jedoch wahrscheinlich, daß Regel `501` häufiger vorgeschlagen und verwendet wird. Sie würde von dem Lernverfahren auch in den folgenden Fällen konstruiert:

1 `Geben Sie mir Auskunft über Zugverbindungen in Fürth.`
2 `Er machte mir einen Vorschlag über eine IC-Fahrt nach Schwabach.`
3 `Ich hätte gerne Informationen über Nachtzüge.`

`Auskunft`, `Information` und `Vorschlag` haben alle SEMCAT:*auskunft*, `Zug`, `Zugverbindung`, `IC-Fahrt` und `Nachtzug` haben SEMCAT:zug. Hingegen haben die Verben `brauchen`, `geben`, `machen` und `haben` unterschiedliche semantische Kategorien. Während die Regel `501` für alle Sätze vorgeschlagen werden würde, würde jeweils eine andere Regel für die Verwendung der Präpositionalphrase als Adjunkt zur Verbalphrase vorgeschlagen werden. Diesen Umstand nutzt die statistische Bewertung über ein größeres Korpus aus.

6.2 Hauptstufe: Bewertung über ein größeres Korpus

Die satzübergreifende Bewertung baut auf den Ergebnissen der Bewertung im Satzkontext auf. Es werden nur noch Regeln betrachtet, die von dem vorherigen Schritt als *gute Regeln* erachtet wurden. Nun wird untersucht, wie oft eine neue Regel von mehreren Sätzen benutzt wurde. Je öfter die Regel verwendet wird, desto höher die Bewertung.

Formal läßt sich die Bewertung von Regeln folgendermaßen darstellen. *GuteRegeln(Satz)* ist dabei die Menge an Regeln, die von der Vorstufe als gute Regeln für *Satz* erklärt wurden.

$$Bewertung(Regel) = \#\{Satz | Regel \in GuteRegeln(Satz)\}$$

Mit Hilfe dieser Regelbewertungen wird nun versucht, aus mehreren Parses zu einem Satz die besten herauszusuchen. Die Bewertung eines Parses ist umso höher, desto bessere neue Regeln er enthält. Parses werden formal folgendermaßen bewertet:

$$Bewertung(Parse_n^{Satz}) = \prod_{Regel_i^{Satz} \in NeueRegeln} Bewertung(Regel_i^{Satz})$$

Dies erlaubt nun wiederum eine Konzentration auf Regeln, die in *guten Parses* verwendet wurden. Somit gibt es eine geringere Zahl von Regeln, die neu bewertet werden können und das Vorgehen wird iteriert. Das Verfahren konvergiert, wenn sich die Zahl der Regeln nicht weiter verringert.

Folgendes Beispiel illustriert die Ausgangssituation nach der ersten satzübergreifenden Bewertung von Regeln:

```
> ev 541
Regel 501 hat Bewertung 2.
Regel 514 hat Bewertung 3.
Regel 525 hat Bewertung 6.
Parse 1 benutzt 501 525, Bewertung 12.
Parse 2 benutzt 501 506 525, Bewertung 0.
Parse 3 benutzt 514 525, Bewertung 18.
Parse 4 benutzt 514 506 525, Bewertung 0.
Parse 5 benutzt 514 503 525, Bewertung 0.
Parse 6 benutzt 514 507 525, Bewertung 0.
Parse 7 benutzt 514 507 506 525, Bewertung 0.
Parse 8 benutzt 514 503 507 525, Bewertung 0.
Parse 9 benutzt 514 507 503 525, Bewertung 0.
```

Nach der Bewertung im Satzkontext wurden nur die drei Regeln `501`, `515` und `525` als *gute Regeln* eingestuft. Diese erhielten eine Bewertung entsprechend ihrer Verwendung in allen Sätzen des Korpus. So wurde `501` in zwei Sätzen des Korpus verwendet und erhält somit die Bewertung 2. Alle Regeln, die nicht zu den *guten Regeln* zählen, werden mit 0 bewertet.

Die Bewertung eines Parses ist das Produkt der Bewertungen der verwendeten neuen Regeln. So lautet für Parse 3 die Berechnung

$$\begin{aligned} Bewertung(Parse_3^{541}) &= Bewertung(Regel_{514}^{541}) \times Bewertung(Regel_{525}^{541}) \\ &= 3 \times 6 \\ &= 18 \end{aligned}$$

Die beiden verbleibenden Parses nach der Bewertung im Satzkontext (Parse 1 und 3) erhalten in dem Beispiel zwei unterschiedliche Bewertungen: 12 und 18. Das Verfahren beschränkt sich nun auf die Parses mit der höchsten Bewertung, also 18. Dies ist nur der Parse 3.

Die im Parse 3 vorkommenden Regeln (`514` und `525`) gelten nach diesem erster Iteration einer satzübergreifenden Bewertung weiterhin als *gute Regeln*. Damit reduziert sich insgesamt die Zahl der neuen Regeln von 28 auf 2, die Zahl der Parses von 9 auf 1; ein ideales Ergebnis:

```
Maximale Bewertung 18.
Parse mit maximaler Bewertung 3
Nach wie vor Gute Regeln 514 525.
2 Regeln uebernommen, 1 Parse.
```

Noch einmal zusammengefaßt stellt sich der Algorithmus zur satzübergreifenden Bewertung folgendermaßen dar:

1. Zähle für jeder der verbleibenden *guten Regeln*, in wie vielen Sätzen des Korpus sie verwendet wurde (d.h. als *gute Regel* eingestuft wurde.).
2. Diese Zahl sei die *Bewertung der Regel*. Alle übrigen Regeln werden mit 0 bewertet.
3. Bilde für jeden Satz *Bewertungen der Parse* aus dem Produkt der Bewertungen der verwendeten *guten Regeln*. Ein Parse wird mit 0 bewertet, wenn er neue Regeln benutzt, die nicht zu den *guten Regeln* des Satzes gehören.
4. Berechne die maximale Bewertung eines Parses innerhalb eines Satzes und verwerfe alle Parses, die nicht diese Bewertung haben. Die Parses mit der maximalen Bewertung sind die *guten Parses*.
5. Erkläre alle konstruierten Regeln in den verbleibenden Parses zu *guten Regeln*
6. Beende des Verfahren, wenn sich die Menge der *guten Regeln* nicht geändert hat. Ansonsten zurück zu Schritt 1.

Der Algorithmus ist in Abbildung 6.2 formal angegeben.

6.3 Andere Bewertungsstrategien

Die vorgestellte statistische Bewertung von konstruierten Regeln ist ein recht einfaches Verfahren. In diesem Abschnitt sollen kurz einige Alternativen dargestellt werden.

Abbildung 6.2: Algorithmus für die Hauptstufe der statistischen Bewertung

Der Algorithmus übernimmt aus dem modellbasierten Lernen:

Parses(Satz) – die Menge an Parses, die für *Satz* erzeugt wurden
NeueRegeln – die Menge aller vorgeschlagenen Regeln
$Parse_i^j$ – Parse i von Satz j
$Regel_i^j$ – Regel i von Satz j

und aus der Vorstufe des statistischen Lernens:

GuteRegeln(Satz) – die Menge der guten Regeln für *Satz*
GuteRegeln – die Menge aller guten Regeln

```
loop
   forall Regel ϵ NeueRegeln
      Bewertung(Regel) := #{ Satz | Regel ϵ GuteRegeln(Satz) }
   forall Satz
      forall Parse ϵ Parses(Satz)
         Bewertung(Parse_n^Satz) := ∏_{Regel_i^Satz ϵ NeueRegeln} Bewertung(Regel_i^Satz)
   MaximaleBewertung = max(Bewertung(Parse))
   forall Satz
      NeueGuteRegeln(Satz) := { Regel ϵ GuteRegeln(Satz) |
                 Parse(Satz) benutzt Regel
                 Bewertung(Parse(Satz)) = MaximaleBewertung }
   exit unless ∃ Satz : GuteRegeln(Satz) ≠ NeueGuteRegeln(Satz)
   forall Satz
      GuteRegeln(Satz) := NeueGuteRegeln(Satz)
   GuteRegeln := ⋃_Satz GuteRegelnSatz
end loop
```

Die Menge *GuteRegeln* enthält schließlich alle Regeln, die letztendlich der Grammatik hinzugefügt werden.

Regelkontext

In dem vorgestellten Lernverfahren wird keinerlei Kontext der Regelanwendung berücksichtigt. Es wird nur betrachtet, ob eine Regel in einem Satz angewendet wurde oder nicht. Als zusätzliche Information könnte man nun noch die davor oder danach angewendeten Regeln einfließen lassen. Dies ließe eine differenziertere Regelbewertung zu.

Ohne die Berücksichtung benötigt man allerdings wesentlich weniger Daten. Hier gilt das selbe, was schon über n-Gramme in Abschnitt 3.1 auf Seite 21 gesagt wurde: Die Größe des benötigten Korpus steigt exponential mit der Anzahl an Vorregeln, die berücksichtigt werden sollen.

Statistisches Maß aus der Semantik

Hilfreich wäre vor allem ein statistisches Maß, das angibt, wieviel *Sinn* die mittels der konstruierten Regeln erzeugten Strukturanalysen des Satzes machen. So wäre zum Beispiel neben dem bloßen Satz auch seine Struktur — in Form eines vereinfachten Parsebaumes — als Eingangsdaten für das Lernverfahren sehr wertvoll zur Aussortierung der konstruierten Regeln. Führen konstruierte Regeln nicht zu einer semantisch überzeugenden Strukturanalyse, schwächt dies die Bewertung dieser Regeln.

Eine andere Möglichkeit, wie dieses Maß erzeugt werden könnte, wäre eine nachgeschaltete semantische Stufe der Regelbewertung. Diese bewertet die erzeugten Parsebäume nach ihrer semantischer Plausibilität und liefert diese Information an das Verfahren zur Regelbewertung zurück.

6.4 Grenzen syntaktischer Sprachverarbeitung

Es gibt grundsätzliche Grenzen syntaktischer Sprachverarbeitung, die auch von einem Lernverfahren für Grammatikregeln nicht gelöst werden. So sind einige Mehrdeutigkeiten erst durch eine semantische oder pragmatische Stufe der Sprachverarbeitung auflösbar. Zur Veranschaulichung sei ein Blick auf folgendes Beispiel gerichtet:

1a `Günter fährt nach Paris.`
1b `Günter fährt nach Paris und München, Hans fährt nur nach München.`
1c `Günter fährt mit dem Zug nach Paris, Hans fliegt mit dem Flugzeug.`
2 `Hans möchte auch nach Paris fahren.`

Worauf bezieht sich das `auch` in dem letzten Satz? Dies hängt entscheidend von dem gegebenen Kontext ab und hat entsprechende semantische Konsequenzen:

- mit dem Kontext 1a bezieht es sich auf `Hans`:
  `Günter fährt nach Paris, auch Hans will es.`
- mit dem Kontext 1b bezieht es sich auf `nach Paris`:
  `Hans fährt nach München, will auch nach Paris.`
- mit dem Kontext 1c bezieht es sich auf `fahren`:
  `Hans fliegt zwar mit dem Flugzeug, aber will auch fahren.`

Diese Mehrdeutigkeit des Bezuges von `auch`, die ja zu verschiedenen Parsebäumen führt, kann uns soll nicht von der syntaktischen Phase der Sprachverarbeitung aufgelöst werden. Alle drei Alternativen sind plausibel und müssen an die semantische Phase der Sprachverarbeitung übergeben werden.

Kapitel 7

Experimente

In der Sprache berühren sich Erwartung und Erfüllung

Ludwig Wittgenstein
Philosophische Untersuchungen, 445

Experimente sind der Punkt, wo eine durchdachte Idee auf die Realität trifft. Die Experimentierumgebung wurde von vornherein beschränkt: bestimmte grammatische Konstrukte wurden ausgeklammert. Genaueres siehe im Anhang A, in welchem die Grammatik erläutert ist.

Dies geschah jedoch ausschließlich, um den Aufwand zur Entwicklung von Grundgrammatik und Grammatikmodell in machbaren Grenzen für diese Arbeit zu halten. Ebenfalls wurde damit der Komplexitätsgrad der Charts geringer gehalten, um die Möglichkeiten des experimentellen Systems nicht zu sprengen.

Für gesicherte statistische Aussagen ist auch die Größe des Trainingskorpus von 140 Sätzen etwas zu gering. Dennoch lassen sich aus den Ergebnissen die Möglichkeiten und Schwierigkeiten des Lernverfahrens ablesen.

Der Code wurde hauptsächlich in nichtkompilierten Prolog implementiert. Details sind in Anhang B erläutert. Die Experimente wurden auf einer SUN Sparcstation 10 durchgeführt.

7.1 Korpus

Die folgenden Experimente wurden mit einem Korpus durchgeführt, das bei anderen Forschungsarbeiten des Instituts für Mathematische Maschinen und Datenverarbeitung (IMMD) an der Universität Erlangen bereits verwendet wurde. Es stammt aus dem Projekt EVAR, das ein Spracherkennungs und -verarbeitungssystem zur Bearbeitung von Zugauskunftsanfragen zum Ziel hat.

Das Korpus ist ein schriftliches Protokoll von tatsächlichen Anfragen. Dabei wurden viele unterschiedliche Formulierungen ausgewählt. Er ist für das Lernverfahren dieser Arbeit ein sehr schweres Korpus: Mündliche Anfragen haben oft ungewöhnlich Formulierungen, die in schriftlicher Sprache nicht gebräuchlich sind. Er enthält auch Äußerungen, die schlichtweg ungrammatikalisch sind, wie zum Beispiel:

1 `Ja dann so am Vormittag am späteren Vormittag vielleicht.`

Von den 140 Äußerungen wurden 29 per Hand als ungrammatikalisch eingestuft. Das Lernverfahren, das zum Ziel hat, nur grammatikalisch plausible Regeln zu lernen, sollte diese Sätze nicht erfolgreich parsen können.

Außerdem ist die experimentelle Grundgrammatik von Anfang an beschränkt (siehe A.8), es wurden aus dem ursprünglichen Korpus Sätze entfernt, die eines der folgenden sogenannten *Nicht-Wörter* enthielten:

`schnellsten, und, zwischen, bin, ist, sind, waere, waers, wars, wenns`

7.2 Qualität der Grundgrammatik

Von den 111 grammatikalischen Sätzen konnten 40 (36 Prozent) von der Grundgrammatik erfolgreich geparst werden. Keine der ungrammatikalischen Äußerungen wurde erfolgreich geparst.

Ein Richtweiser für die Qualität der Grundgrammatik ist die Anzahl der Parses pro Satz. Je mehr Parses pro Satz erzeugt werden, desto mehr unterschiedliche Syntaxanalysen des Satzes werden von der Grammatik erzeugt. Da jedoch nur wenige Analysen syntaktisch plausibel sind, ist ein Gradmesser für die Qualität der Grammatik eine niedrige Anzahl von Mehrdeutigkeiten.

In Tabelle 7.2 ist die Anzahl der Parses pro Satz für die 40 erfolgreich geparsten Sätze angegeben. Für 75 Prozent der Sätze wurden nur ein oder zwei Parses produziert. Dies ist ein Hinweis dafür, daß die Grundgrammatik die syntaktische Form der Sätze richtig modelliert.

Tabelle 7.1: Anzahl der Parses pro Satz

Parses	Anzahl Sätze
1	17
2	13
3	3
4	3
5	1
6	2
12	1

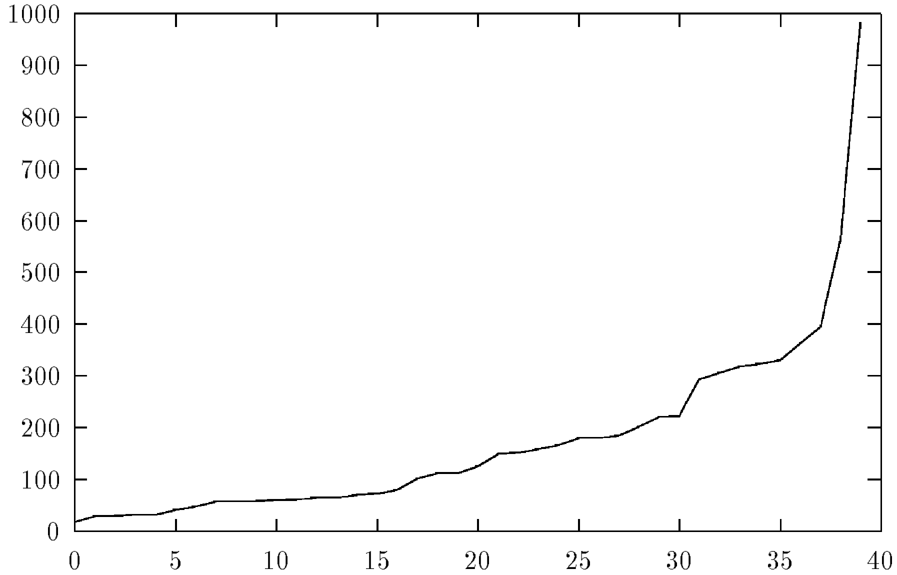
Einen Eindruck über das Verhalten des Parsings mit einer Unifikationsgrammatik wie der Grundgrammatik gibt das Diagramm in Abbildung 7.1. Das Diagramm listet die Anzahl der Charteinträge pro Satz der Größe nach sortiert auf. Für die Hälfte der Sätze wurden weniger als 120 Charteinträge erzeugt. Das Parsing dieser Sätze mittels des implementierten Systems dauerte in diesen Fällen nur wenige Sekunden. Andererseits gab es einige Ausreißer: beim Parsing eines der Sätze wurden fast 1000 Charteinträge erzeugt.

7.3 Qualität des Lernverfahrens

Von den 71 grammatikalischen Sätzen, für die die Regeln der Grundgrammatik nicht ausreichten, konnten 45 (63 Prozent) von der Grundgrammatik erfolgreich geparst werden. Aber auch mit

Abbildung 7.1: Anzahl der Charteinträge (ursprünglich geparste Sätze)

Dargestellt sind die Anzahl an Charteinträgen, die für jeden der 40 Sätze beim Parsen erzeugt wurden, die bereits von der Grundgrammatik geparst werden konnten. Die Sätze sind nach ihrer Anzahl an Charteinträge sortiert.



den gelernten Regeln wurden keine der ungrammatikalischen Äußerungen erfolgreich geparst. Für 3 Sätze dauerte das Lernverfahren länger als 25 Stunden und wurde abgebrochen. Diese Ergebnisse sind zum Überblick in Tabelle 7.2 zusammengefaßt.

Tabelle 7.2: Grundlegendes Ergebnis der Experimente

Ergebnis	Anzahl
ursprünglich geparste Sätze	40
durch das Lernverfahren geparste Sätze	45
nicht geparste grammatikalisch wohlgeformte Sätze	23
ungrammatikalische Sätze	29
Lernverfahren zu langsam	3
insgesamt	140

Das Lernverfahren dauert wesentlich länger als der Versuch mit der Grundgrammatik, die Sätze zu parsen. Da zusätzliche Regeln vorgeschlagen werden, werden erheblich mehr Charteinträge erzeugt. Die Abbildung 7.2 gibt einen Eindruck davon.

Um die Rolle der statistischen Regelbewertung abschätzen zu können, betrachten wir nun die Anzahl an Regeln, die für jeden Satz vorgeschlagen werden und letztendlich als *gute Regeln* übernommen werden.

Abbildung 7.3 gibt zunächst an, wieviele neue Regeln für jeden Satz vom modellbasierten Lernen vorgeschlagen werden. Die Anzahl reicht von 1 bis 28, im Durchschnitt sind es 13,6.

In Abbildung 7.4 ist angegeben, wieviele Parses pro Satz erzeugt wurden.

Abbildung 7.2: Anzahl der Charteinträge (beim Lernen)

Dargestellt sind die Anzahl von Charteinträgen, die für jeden der 97 Sätze beim letzten Parseschrittes des Lernverfahrens erzeugt werden. In dem Diagramm sind alle Sätze berücksichtigt, also auch die ungrammatikalischen und jene für die das Lernverfahren fehlschlägt. Die Sätze sind wie auch in Abbildung 7.1 nach ihrer Anzahl an Charteinträge sortiert.

Die Kurve hat eine ähnliche Erscheinung wie die in Abbildung 7.1, allerdings werden wesentlich mehr Charteinträge erzeugt: bis über 10,000.

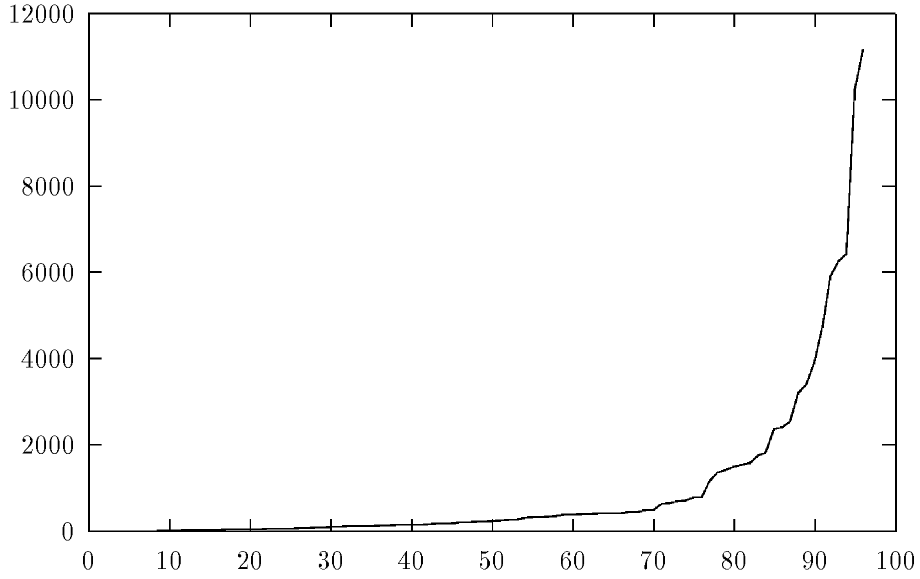


Abbildung 7.3: Anzahl der vorgeschlagenen Regeln

Dargestellt sind die Anzahl an Regeln, die für jeweils einem der 45 Sätze vorgeschlagen wurden, die nach dem Lernverfahren erfolgreich geparst werden konnten. Wie auch in den vorigen Diagrammen wurden die Sätze sortiert, um eine übersichtliche Darstellung zu gewinnen.

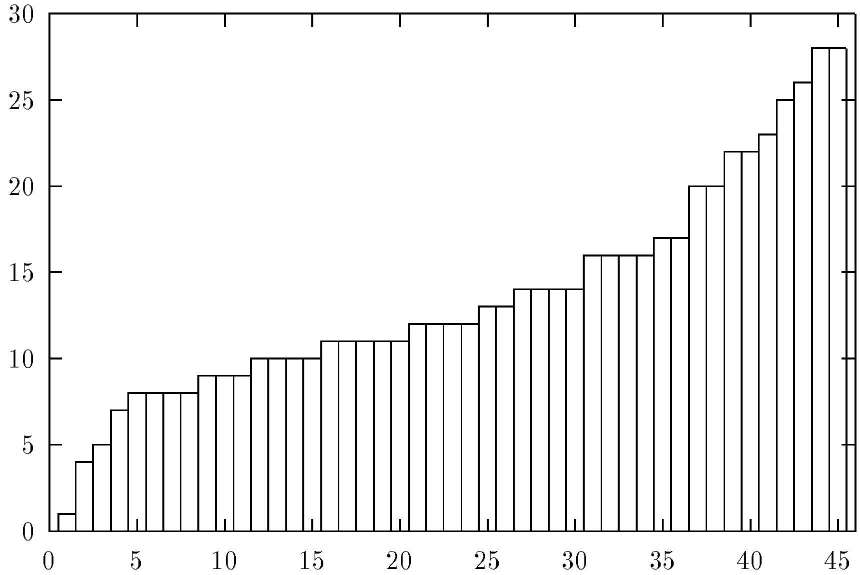
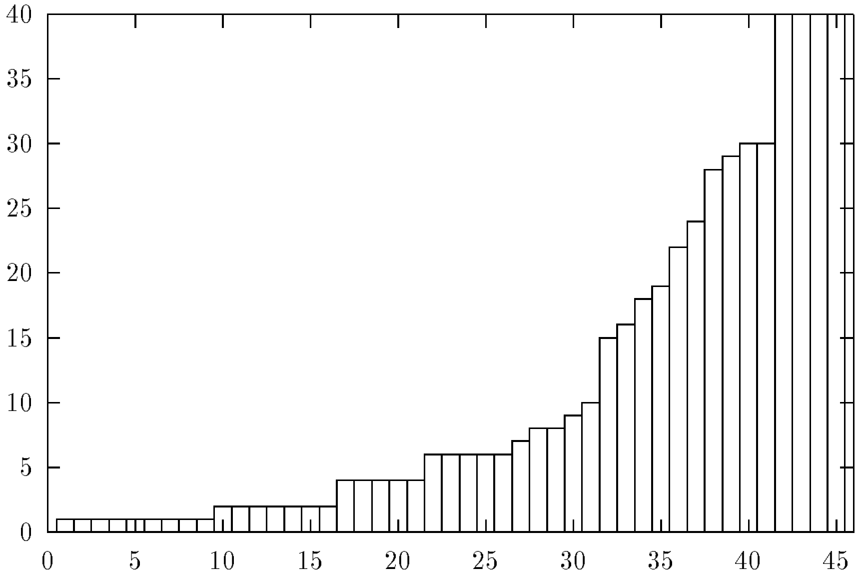


Abbildung 7.4: Anzahl der Parses mit den vorgeschlagenen Regeln

Dargestellt sind die Anzahl an Parses pro Satz nach der Regelkonstruktion und vor der statistischen Regelbewertung. Die Daten sind sortiert wie in Abbildung 7.3. Für die vier Sätze mit den meisten Parses lauten die Anzahlen 60, 96, 200 und 624.



Regelbewertung im Satzkontext

Bei der Regelbewertung im Satzkontext wird bereits ein Großteil der vorgeschlagenen Regeln aussortiert, wie Tabelle 7.5 verdeutlicht. Die durchschnittliche Regelzahl sinkt nach diesem Schritt des Lernverfahrens von 13,6 auf 1,9.

Regelbewertung über das gesamte Korpus

Nach der Regelbewertung im Satzkontext gibt es 55 verschiedene Regeln, die jetzt über das ganze Korpus hinweg bewertet werden. Dies reduziert die Anzahl der Regeln noch einmal, wie aus Abbildung 7.6 zu ersehen ist. Die durchschnittliche Zahl an gelernten Regeln pro Satz liegt somit bei 1,8. Es wurden 4 Regeln aussortiert, es gibt nun noch 51 verschiedene.

In Abbildung 7.7 ist angegeben, wieviele Parses pro gelerntem Satz nach der statistischen Bewertung erzeugt wurden. Im Durchschnitt sind dies 2,3 pro Satz.

Das Korpus an gelernten Sätzen in diesem Experiment (insgesamt nur 45) ist jedoch zu klein, um die Wirksamkeit dieses Schrittes wirklich entfalten zu lassen. Tabelle 7.3 gibt eine Übersicht, in wie vielen Sätzen gute Regeln nach der Vorstufe der statistischen Regelbewertung angewendet wurden. Die meistbenutzte gelernte Regel findet gerade einmal 6mal Verwendung – für eine befriedigende statistische Auswertung ist dies natürlich zu wenig.

Zeitverhalten

In Einzelfällen dauert das Lernverfahren extrem lange, wie aus Tabelle 7.4 zu ersehen ist. Dies liegt an dem explosiven Anwachsen der Zahl der Parses bei mehreren lokalen Mehrdeutigkeiten.

Abbildung 7.5: Anzahl der guten Regeln nach Bewertung im Satzkontext

Dargestellt sind die Anzahl an guten Regeln pro Satz wie in Abbildung 7.3

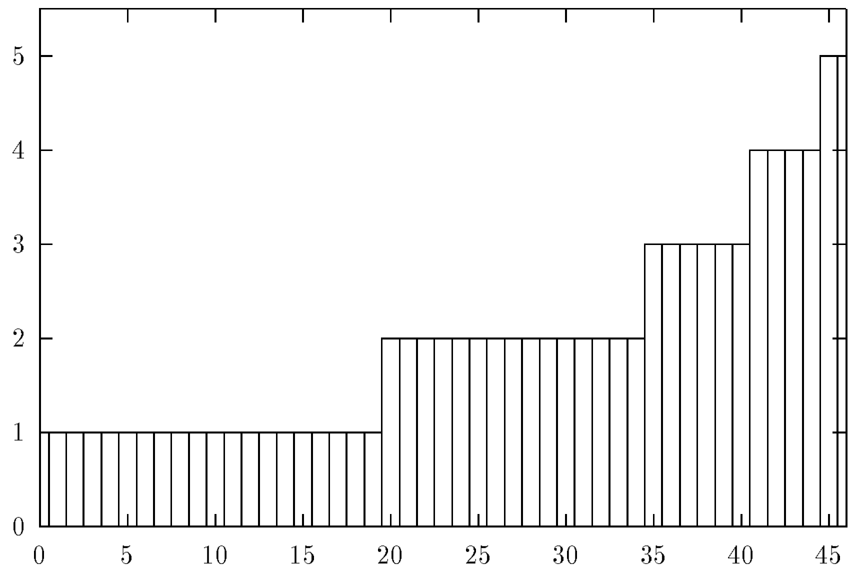


Abbildung 7.6: Anzahl der guten Regeln nach Bewertung über gesamtes Korpus

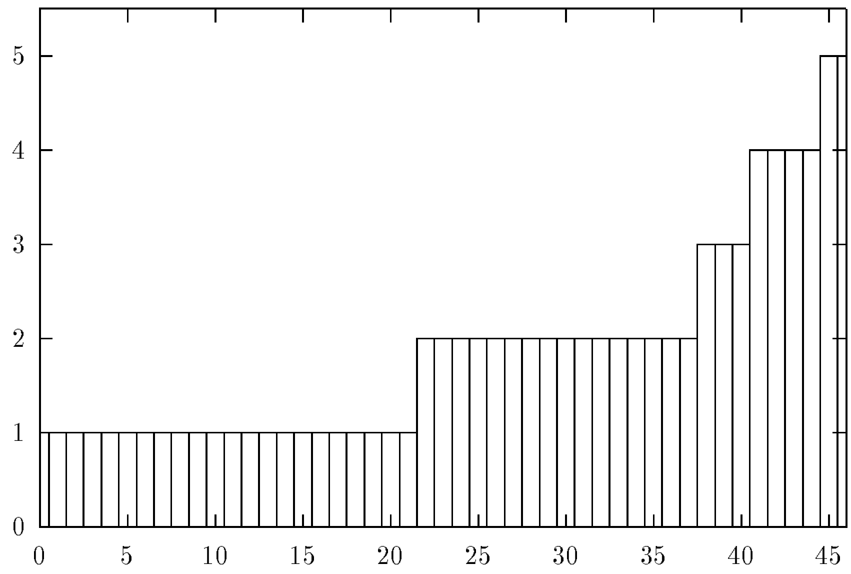


Abbildung 7.7: Anzahl der Parses pro Satz nach Bewertung über gesamtes Korpus

Die Darstellung ist sortiert wie z.B. aus Abbildung 7.3 gewohnt.

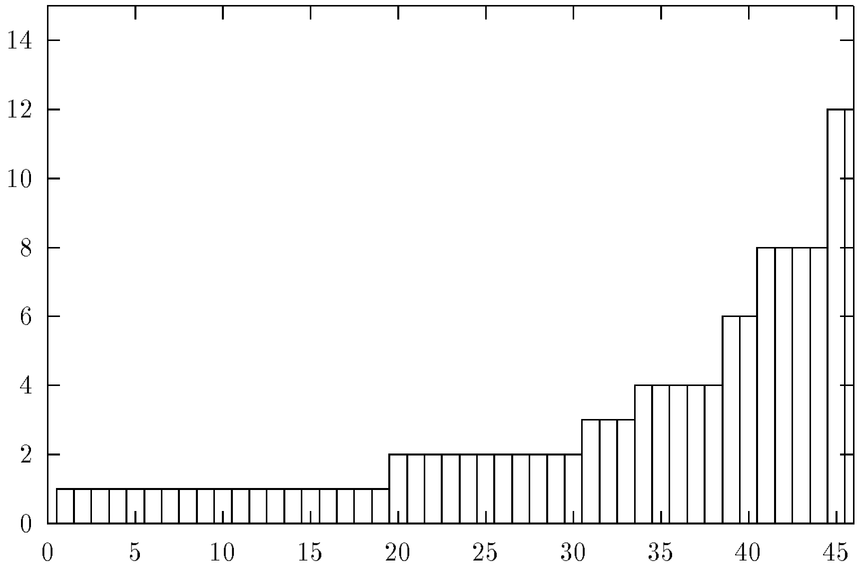


Tabelle 7.3: Anwendung von gleichen Regeln in Sätzen

Wie oft angewendet	Anzahl Regeln
1 mal	34
2 mal	13
3 mal	6
4 mal	1
6 mal	1

So werden beispielsweise für den Satz `518`[1] insgesamt 620 Parses erzeugt, das Verfahren benötigte in der verwendeten Implementierung 22 Stunden für diesen einen Satz.

In der Tabelle 7.4 sind alle Sätze berücksichtigt, also auch jene, die nicht erfolgreich gelernt werden konnten. In der Regel ist das Verfahren schnell genug, dreiviertel der Sätze wurden in weniger als 20 Minuten bearbeitet. Je länger die Sätze jedoch sind, mit denen das Verfahren konfrontiert wird, desto mehr Kanten werden beim ersten Parsen erzeugt und entsprechend mehr neue Regeln werden auch vorgeschlagen, mit den erwähnten Folgen. Es sollten daher nicht zu lange Sätze in das Lernkorpus aufgenommen werden.

Tabelle 7.4: Dauer des Lernverfahrens

Zeitdauer	Anzahl der Sätze
bis 20 min	72
20 min bis 3 Stunden	16
3 bis 10 Stunden	7
10 bis 20 Stunden	1
20 bis 30 Stunden	1
mehr als 30 Stunden	3

Vergleicht man diese Zeiten jedoch mit Osbornes Arbeit [Osb94], zeigt sich ein wesentlicher Vorteil des durchgeführten grammatisch strengeren Ansatzes. Osborne hat weniger Beschränkungen für die vorgeschlagenen Regeln. Es wird dadurch eine wesentlich größere Grammatik erzeugt, für die vollständiges Parsen nicht mehr möglich ist. Osborne bricht das Parsing von Sätzen ab, sobald 10 Parses erzeugt wurden. Dies scheint jedoch kein sinnvolles Vorgehen, da dadurch wertvolle Informationen verloren gehen und es nicht einzusehen ist, warum die ersten zehn Parses gegenüber den später erzeugten bevorzugt werden sollten.

Fehleranalyse

Von insgesamt 111 grammatisch wohlgeformten Sätzen konnten 23 durch das implementierte Experimentalsystem nicht geparst werden. Dies ist vor allem auf das unzureichend entwickelte zugrundeliegende lingustische Modell zurückzuführen. So ist eine Schwäche der Grammatik offenbar die unzureichende Behandlung von Demonstrativpronomen wie `hier`, `den` oder `diese`. Dies führte bei 10 Sätzen zum Fehlschlagen des Verfahrens.

7.4 Zusammenfassung

Von den 111 grammatisch wohlgeformten Sätzen des Testkorpus wurden 40 bereits von der Grundgrammatik geparst, 45 konnten nach Regelkonstruktion erfolgreich geparst werden. Bei 23 Sätzen versagte das Verfahren. Für 3 Sätze dauerte das Lernverfahren länger als 100 Stunden und wurde abgebrochen.

Im Sinne der in Kapitel 4 definierten Kriterien läßt sich dieses Ergebnis wie folgt beurteilen:

[1] `518 koennen sie mir sagen wann ich spaetestens in muenchen losfahren muss wenn ich noch vor zehn uhr in augsburg sein will`

Untergenerierung wurde erfolgreich reduziert. Nach dem Lernverfahren konnten doppelt so viele Sätze geparst werden wie vorher.

Übergenierung ist durch das Lernverfahren nicht erkennbar aufgetreten: Keine der ungrammatischen Äußerungen wurde mit den gelernten Regeln geparst.

Plausibilität kann nur indirekt beurteilt werden: Die niedrige Zahl an erzeugten Parses pro Satz deutet aber darauf hin, daß die Plausibilität der Grammatik weitgehend erhalten blieb.

Die Wirkung der beiden Phasen der statistischen Bewertung von Regeln wurde positiv demonstriert. Durch die statistische Kompenente konnte die Zahl der vorgeschlagenen Regeln und damit die Vielfachheit von resultierenden Parses deutlich reduziert werden. Dadurch bleibt die Grammatik auch nach dem Hinzufügen gelernter Regeln relativ kompakt.

Kapitel 8

Bewertung

Die Zielrichtung dieser Arbeit war es, menschliche Sprache mit all ihren Ungenauigkeiten und okskuren Sonderfällen in ein Korsett der Ordnung zu zwängen. So ein Vorhaben muß natürlich scheitern. Genausowenig wie ein Mensch jemals eine Sprache *vollkommen* beherrschen wird, wird auch ein Computerprogramm immer fehlerhaft arbeiten. Das Maß für den Erfolg ist der Grad der Verfehlung.

8.1 Erfolge

Ein Lernverfahren zur automatischen Grammatikerweitung für die deutsche Sprache wurde beschrieben und seine Effektivität durch erfolgreiche Experimente anhand einer exemplarischen Implementierung demonstriert. Die Effektivität aller Schritte des Lernverfahrens konnte gezeigt werden.

Das Verfahren erhielt eine kleine und kompakte Grammatik. Dadurch konnte in den meisten Fällen ein effizientes Parsen von Sätzen durchgeführt werden. Es mußte – im Gegensatz zu Osbornes Verfahren – nicht vor den unnötig erzeugten Mehrdeutigkeiten kapituliert werden.

Die Fehlerquote des Experimentiersystems betrug 32 Prozent, 23 von 71 grammatisch wohlgeformten Sätzen konnten nicht von dem Lernsystem erfolgreich geparst werden. Diese recht hohe Quote kann auf das noch nicht ausgereifte zugrundeliegende grammatische Modell zurückgeführt werden.

Das Lernverfahren fußt auf zwei Vorgehensweisen, um Regeln zu generieren: der modellbasierten und der statistischen Regelbewertung. Beide haben ihre Tücken: Das modellbasierte Lernen kann zu eng für die Mannigfaltigkeit der sprachlichen Erscheinungen angelegt sein. Das statistische Lernen kann ebenfalls zu falschen Ergebnissen führen: Regeln, die häufig verwendet werden, müssen nicht richtig sein. Die Kombination aus beiden scheint jedoch eine gute Herangehensweise zu sein. Für ein letztendliches Urteil ist jedoch ein größeres Testkorpus notwendig.

8.2 Grenzen und Probleme

Man kann das Lernverfahren auch folgendermaßen betrachten: Es ist ein Suchverfahren, das eine adäquate Grammatik für die deutsche Sprache finden soll. Als Untergrenze ist die Grundgrammatik gegeben, die Obergrenze sind alle Regeln, die von dem Grammatikmodell erlaubt werden.

Die Qualität des Lernverfahrens hängt immer von der Gestaltung dieser beiden Grenzen ab: Sind sie falsch gesetzt, reduziert dies den Erfolg des Lernverfahrens.

So werden in dem Lernverfahren keine neuen Merkmale oder Kategorien erzeugt. Dies ist eine echte Einschränkung: Um zum Beispiel Regeln einzuführen, die nur einmal innerhalb einer Phrase vorkommen dürfen, aber an beliebiger Stelle, ist ein Merkmal notwendig, das die Verwendung dieser Regel kontrolliert.

Ein schwerwiegendes Problem ist der Grad der Allgemeinheit von Regeln. Wenn ein unbekanntes sprachliches Phänomen in einem Satz vorkommt, ist es zunächst unklar, ob es nur in genau diesem Zusammenhang vorkommen kann oder ein Beispiel für ein allgemeineres grammatisches Prinzip ist. Die statistische Regelbewertung bewertet Regeln ja nur nach ihrer Häufigkeit, nicht ihrem Grad an Allgemeinheit. Sie wird zudem immer allgemeinere Regeln bevorzugen — diese können ja öfters angewendet werden als speziellere. Es wurde ein einfaches Vorgehen zur Behandlung dieses Problems vorgeschlagen (siehe Abschnitt 5.7), das recht überzeugend funktioniert. Ob es allerdings das letzte Wort dazu darstellt, ist fraglich.

Problematisch ist auch die Frage des linguistischen Wissens, das in das Verfahren gesteckt werden muß. Im Gegensatz zu handgeschriebenen Grammatiken stehen nun zwar zwei Ebenen zu Verfügung auf denen dieses Wissen in das System einfließen kann. Dennoch ist es unklar, ob linguistisches Wissen überhaupt in ausreichender Menge in eine solche Form gebracht werden kann. Für das Schreiben von Grammatiken von Hand scheint dies nicht möglich zu sein. Ob dieses Lernverfahren einen genügend vereinfachten Formulalismus zur Verfügung stellt kann zumindest bezweifelt werden.

Ein grundsätzliches Problem von Unifikationsgrammatiken ist das ungünstige Verhalten bei Disambiguitäten. Jede Mehrdeutigkeit bedeutet eine Vervielfachung des Rechenaufwandes. Dies bedeutet selbst bei Disambiguitäten, die nur lokal existieren und die sich in einem größeren Kontext auflösen, einen großen Overhead.

Überhaupt scheint die Verwendung von Unifikationsgrammatiken für die deutsche Sprache Probleme zu bereiten: Im Deutschen gibt es viele Bezüge zwischen entfernt liegenden Satzteilen, die sich nur sehr umständlich mit Grammatikregeln fassen lassen, die ja immer nur lokale Bezüge definieren. Dies ist in dem Fall des vorgestellten Lernverfahrens noch verschärft problematisch, da es nur maximal binäre Regeln zuläßt.

8.3 Ausblick

Das vorgestellte Lernverfahren läßt sich in vielerlei Hinsicht verbessern und erweitern. So sind beispielsweise das Grammatikmodell und die Grundgrammatik, die ad hoc entwickelt und ständig überarbeitet wurden, sicherlich noch nicht ausgereift. Mit den gemachten Erfahrungen (siehe auch Anhang A) sollte es aber möglich sein, hier Konstruktionen zu entwickeln, die linguistisches Wissen besser berücksichtigen.

Die statistische Regelbewertung läßt sich ebenfalls noch verfeinern. Einige Gedanken wurden ja bereits weiter oben (Abschnitt 6.3) dazu angerissen. So könnte der Regelkontext bei der Bewertung berücksichtigt werden. Auch könnten zusätzliche semantische Bewertungen in Form von vorgegebenen Strukturierungen des Satzes oder einem nachgestellten Bewertungsschritt einfließen.

Die Effizienz des Parsings ist ein großes Problem, wie die Experimente gezeigt haben (siehe Abschnitt 7.3). Da für eine Bewertung der konstruierten Regeln alle Parses erzeugt werden müssen, führt dies bei vielen Mehrdeutigkeiten zu einem explosiven Anstieg des Berechnungsaufwandes. Dies ließe sich entscheidend reduzieren, indem Mehrdeutigkeiten, die sich nur lokal auswirken, nicht in verschiedenen Charteinträgen, sondern entsprechend gepackt in nur einer Kante berücksichtigt werden. So unterscheiden sich oft viele Charteinträge nur in ihrer Entstehungsgeschichte, nicht aber ihrer Merkmalsstruktur. Beim Parsen spielt jedoch nur die Merkmalsstruktur ein Rolle.

Bisher findet nur positive Information in dem Verfahren Berücksichtigung: Aufgrund von zu parsenden Sätzen wird die Grammatik erweitert. Hilfreich wäre auch Berücksichtigung negativer Information: Wenn die erweiterte Grammatik Wortfolgen fälschlicherweise als grammatisch wohlgeformte Sätze einstuft, sollte eine Komponente des Verfahrens einschreiten und dieses Urteil zur Regelbewertung heranziehen.

Neben diesem Ansatz für Grammatikregeln sind auch Lernverfahren für andere Daten eines sprachverarbeitenden Systems möglich: So könnte eine semantische Hierarchie für Lexikoneinträge nach ihrer Verwendung mit statistischen Methoden entwickelt werden. Vielleicht ist sogar die Generierung von Lexikoneinträgen unbekannter Wörter anhand ihrer Verwendung möglich.

59

Literaturverzeichnis

[All95] James Allen. *Natural Language Understanding.* Benjamin Cummings Publishing Company, 1995.

[And89] John R. Anderson. *Kognitive Psychologie.* Spektrum der Wissenschaft Verlagsgesellschaft, Heidelberg, 1989.

[Bra87] Ivan Bratko. *Prolog.* Addison-Wesley, 1987.

[Cov94] Michael A. Covington. *Natural Language Processing for Prolog Programmers.* Prentice Hall, 1994.

[Den91] Daniel C. Dennett. *Consciousness Explained.* Little, Brown and Company, 1991.

[Dre92] Herbert Dreyfus. *What Computers Still Can't Do.* MIT Press, 1992.

[Eps92] Robert Epstein. Can machines think? *AI Magazine*, Summer:80–95, 1992.

[GBCB92] Clare Grover, Ted Briscoe, John Carroll, and Bran Boguraev. The alvey natural language tools grammar (third release). Technical report, University of Cambridge Computer Laboratory, 1992.

[GM89] Gerald Gazdar and Chris Mellish. *Natural Language Processing in Prolog.* Addison-Wesley, 1989.

[Gör88] Günter Görz. *Strukturanalyse natürlicher Sprache.* Addison-Wesley, 1988.

[Har62] Justus Hartnack. *Wittgenstein und die moderne Philosophie.* Urban Bücher, 1962.

[LJ80] George Lakoff and Mark Johnson. *Methaphors We Live By.* The University of Chicago Press, 1980.

[Mag94] David M. Magerman. *Natural Language Parsing as Statistical Pattern Recognition.* PhD thesis, Stanford University, February 1994.

[O'K90] Richard A. O'Keefe. *The Craft of Prolog.* MIT Press, Cambridge, Massachusetts, 1990.

[Osb94] Miles Osborne. *Learning Unification-Based Natural Language Grammars.* PhD thesis, University of York, September 1994.

[Shi85] S. Shieber. Evidence against the contexte-freeness of natural language. *Linguistics and Philosophy*, 8:333–343, 1985.

[SN58] Herbert Simon and Allen Newall. Heuristic problem solving: The next advance in operations research. *Operations Research*, 6(January-February), 1958.

[SO90] Clive Souter and Tim O'Donoghue. Probabilistic parsing in the COMMUNAL project. Technical Report 90.2, University of Leeds School of Computer Studies, 1990.

[ST94] Ernst Günter Schukat-Talamazzini. Automatische Spracherkennung. Habilitationsschrift, Universität Erlangen, September 1994.

[TGB89] L.C. Taylor, C. Grover, and E.J. Briscoe. The syntactic regularity of english noun phrases. In *Proceedings, 4th European Assosiation for Computational Linguistics*, pages 256–263, 1989.

[Tur50] Alan Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.

[Web95] Hans Weber. *LR-inkrementelles probabilistisches Chartparsing von Worthypothesenmengen mit Unifikationsgrammatiken: Eine enge Kopplung von Suche und Analyse.* PhD thesis, Universität Hamburg, 1995.

[Wei66] Joseph Weizenbaum. Eliza – a computer program for the study of natural language communication between man and machine. *Communications of the Association for Computing Machinery*, 9:36–45, 1966.

[Win90] Patrick H. Winston. *Artificial Intelligence at MIT.* MIT Press, 1990.

[Wit60] Ludwig Wittgenstein. *Schriften.* Suhrkamp, 1960.

Anhang A

Die verwendete Grundgrammatik

Wir wollen in unserem Wissen vom Gebrauch der Sprache eine Ordnung herstellen: eine Ordnung zu einem bestimmten Zweck, eine von vielen Ordnungen, nicht die *Ordnung.*

Ludwig Wittgenstein
Philosophische Untersuchungen, 102

Das Erstellen einer Grammatik für eine natürliche Sprache ist ein mühseliges Unterfangen. Es tauchen immer wieder Ausnahmen und skurrile Besonderheiten auf, die das Aufstellen von speziellen Regeln erfordern. Oft stellt sich die Frage, wieviel Aufwand getrieben werden soll, um diese Spezialfälle eng einzugrenzen. Oder ob es nicht besser ist, ein gewisses Maß an Übergeneralisierung zuzulassen, um die Grammatik übersichtlicher zu gestalten.

Zudem ist eine Grammatik nie fertig: es gibt immer etwas hinzuzufügen und zu ergänzen. So ist auch die hier vorgestellte Grundgrammatik bei weitem nicht vollständig. Aber gerade dies ist ja die Motivation für die vorliegende Arbeit: Das vorgestellte Verfahren soll die Grammatik automatisch vervollständigen. Hauptaufgabe der Grundgrammatik ist ausschließlich die grundlegende Struktur vorzugeben.

Diese Grundgrammatik wurde vor allem im Hinblick auf die Anforderungen des Lernsystems entworfen. Neben semantischer Plausibilität war effiziente Verwendbarkeit das Leitmotiv. Die Grammatik und der verwendete Formalismus sind pragmatisch entwickelt worden, sie basieren auf keiner linguistischen Theorie. So wird beispielsweise mit dem Merkmal SUBCAT meist etwas anderes bezeichnet. Hier wird es als Unterkategorie verwendet.

A.1 Satzschema

Für das Englische sind verbzentrierte Ansätze üblich. Links vom Verb liegt das Subjekt, rechts Objekte und andere Adjunkte. Dieser Ansatz läßt sich nicht einfach auf das Deutsche übertragen. Hier ist nur die Position des Verbes festgelegt, das zudem aufgespalten werden kann. Das *Satzschema* des Deutschen ist in Abbildung A.1 (nach Anregung von Hans Weber [mündlich]) angegeben.

Abbildung A.1: Satzschema

	VORF.	VERB 1	MITTELFELD	VERB 2	NACHFELD
	0/1 K.		0/∞ Komponenten		
Aussage	ich	fahre	nach Nürnberg		
Frage	was	willst	du | da		
Aussage	ich	will	Schuhe	kaufen	
Frage		kaufst	du | wirklich | Schuhe		
Aussage	ich	brauche	unbedingt | Schuhe		und zwar schöne
Befehl		Geh	doch | endlich		

Bei Befehls- und normalen Fragesätzen ist das Vorfeld leer, bei Fragesätzen mit Fragewort und Aussagesätzen befindet sich dort genau eine Komponente. Die übrigen Komponenten befinden sich im Mittelfeld. Mit *Komponenten* sind dabei Subjekt, Objekte, Präpositionalphrasen und Adjunkte gemeint. Komponenten, die notwendig für den Satz sind, werden als *Argumente* bezeichnet.

Wie kann nun dieses Satzschema in einer Grammatik verwirklicht werden? Abbildung A.2 gibt einen Eindruck, wie ein typischer Parsebaum aussieht. Der Satz wird als Verbalphrase aufgebaut. Dies geschieht mittels Regeln, die auf der linken Seite die Satzteile in die Verbalphrase aufnehmen. Die jeweilige Position im Satz wird dem Merkmal POS vermerkt. (Die Bezeichnung *Verbalphrase* ist aus dem Englischen übernommen, obwohl sie — wie oben bemerkt — für das Satzschema des Deutschen nicht unbedingt angemessen ist.)

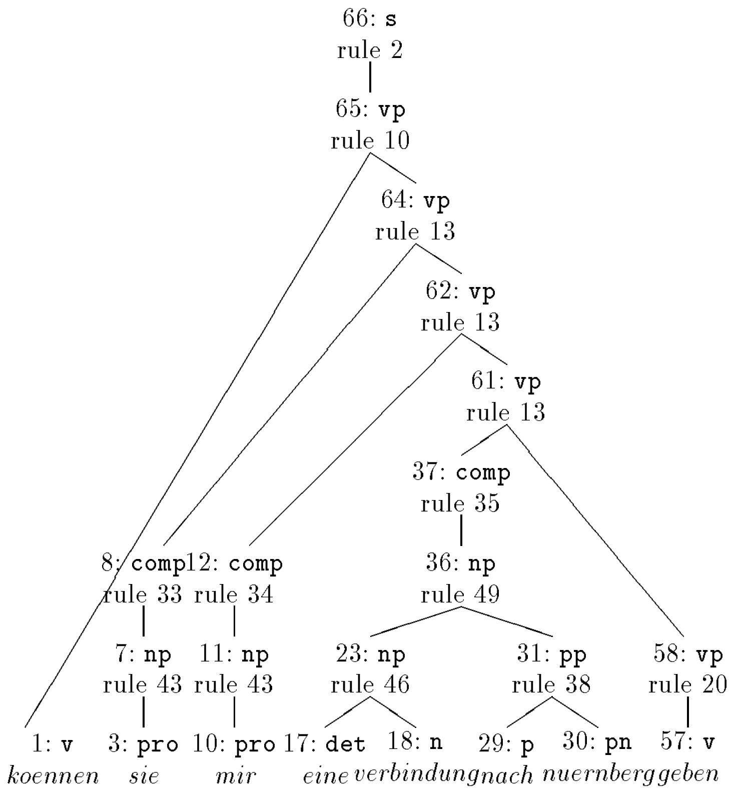
Dies sei an dieser Stelle durch ein Beispiel verdeutlicht. Die Regel, die ein Vollverb (SUBCAT: *fin*) in der Position VERB 1 in die Verbalphrase einbaut, sieht folgendermaßen aus:

$$\begin{bmatrix} \text{CAT}: vp \\ \text{SUBCAT}: fin \\ \text{PERSON}: \langle 2 \rangle \\ \text{AGR}: \langle 3 \rangle \\ \text{POS}: verb \\ \text{RREQ}: \langle 7 \rangle \\ \text{RAGR}: \langle 8 \rangle \\ \text{TAGR}: \langle 9 \rangle \\ \text{VAGR}: \langle 10 \rangle \\ \text{SEMCAT}: \langle 11 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT}: v \\ \text{SUBCAT}: fin \\ \text{PERSON}: \langle 2 \rangle \\ \text{AGR}: \langle 3 \rangle \\ \text{VAGR}: \langle 10 \rangle \\ \text{SEMCAT}: \langle 11 \rangle \end{bmatrix} \begin{bmatrix} \text{CAT}: vp \\ \text{SUBCAT}: fin \\ \text{PERSON}: \langle 2 \rangle \\ \text{AGR}: \langle 3 \rangle \\ \text{POS}: midfield \\ \text{RREQ}: \langle 7 \rangle \\ \text{RAGR}: \langle 8 \rangle \\ \text{TAGR}: \langle 9 \rangle \\ \text{VAGR}: \langle 10 \rangle \\ \text{SEMCAT}: \langle 11 \rangle \end{bmatrix}$$

Ein Verb hat zwei Merkmale, die mit den Komponenten des Satzes übereinstimmen müssen: zum einen Numerus und Person, die im AGR-Merkmal kodiert sind, zum anderen die verlangten Argumente, die sogenannte *Valenz* des Verbes (Merkmal VAGR). Ein Vollverb liefert beide diese Merkmale. Bei einer Hilfsverbkonstruktion kommt das AGR-Merkmal vom Hilfsverb und

Abbildung A.2: Ein typischer Parsebaum der Grammatik

An diesem Parse kann man das Satzschema gut erkennen: Die Verbalphrase wird aufgebaut, indem sie linksseitig Verben und Komponenten aufnimmt. Einzelne Komponenten können schlicht aus einem Pronomen (wie *sie*), oder aus größeren Konstrukten wie eine Nominalphrase mit Präpositionalphrasenadjunkt bestehen (*eine verbindung nach nuernberg*). Die Zahlen vor den Kategorien sind die Nummern der Charteinträge.



das VAGR-Merkmal vom Verb im Infinitiv. Gleiches gilt für Modalverbkonstruktionen. (Auf die übrigen Merkmale wie TAGR wird weiter unten ausführlich eingegangen. An dieser Stelle ist nur wichtig, daß diese weitere Eigenschaften beschreiben, die im Satz übereinstimmen müssen.)

Valenz des Verbes

Die Valenz des Verbes gibt vor, welche Argumente verlangt werden. Notwendige Komponenten können neben dem Subjekt verschiedene Arten von Objekten sein. Zwei Beispiele:

$$\texttt{geben} \qquad \mathrm{VAGR}: \begin{bmatrix} \mathrm{SUBJ}: & \mathrm{need} \\ \mathrm{OBJ1}: & \mathrm{need} \\ \mathrm{OBJ2}: & \mathrm{may} \end{bmatrix}$$

$$\texttt{wollen} \qquad \text{VAGR:} \begin{bmatrix} \text{SUBJ: need} \\ \text{PPOBJ: need}(\textit{nach}, \textit{stadt}) \end{bmatrix}$$

Das Verb `geben` hat ein optionales Dativobjekt, die angegebene Valenz für das Verb `wollen` beschreibt die Verwendung des Verbes in Sätzen wie zum Beispiel:

1 `Ich will nach Nürnberg.`

Neben dem Typ des Argumentes können also auch deren semantischen Kategorien festgelegt werden. Dies ist in diesem Fall nötig, da nicht beliebige Präpositionalphrasen als Objekt möglich sind. Tabelle A.1 gibt eine Übersicht über alle Argumenttypen.

Tabelle A.1: Typen von Argumenten für die Verbalphrase

Merkmal	Beschreibung	Beispiel
SUBJ	Subjekt	<u>ich</u> fahre
OBJ1	Akkusativ Objekt	liest <u>das Buch</u>
OBJ2	Dativ Objekt	gibt <u>mir</u>
OBJ3	Genitiv Objekt	<u>bedarf</u> der Pflege
PPOBJ	Präpositionalphrase als Objekt	fährt <u>nach Erlangen</u>
SCOMP	Nebensatz	sagt, <u>daß sie glücklich ist</u>
VCOMP	Verbpartikel	fährt <u>weg</u>

Argumentregeln

Die Verwendung des Anforderungsmechanismus der erweiteren Merkmalsstrukturen (siehe Abschnitt 2.5) ermöglicht nun, daß die Anforderungen des Verbes über den Satz hinweg erfüllt werden können. Das Verb sammelt somit seine Argumente ein. Die Regel, die Argumente im Mittelfeld einsammelt, hat folgende Gestalt:

$$\begin{bmatrix} \text{CAT: } \textit{vp} \\ \text{SUBCAT: } \langle 1 \rangle \\ \text{AGR: } \langle 2 \rangle \\ \text{POS: } \textit{midfield} \\ \text{TAGR: } \langle 3 \rangle \\ \text{VAGR: } \langle 4 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT: } \textit{comp} \\ \text{AGR: } \langle 2 \rangle \\ \text{VAGR: } \langle 4 \rangle \end{bmatrix} \begin{bmatrix} \text{CAT: } \textit{vp} \\ \text{SUBCAT: } \langle 1 \rangle \\ \text{AGR: } \langle 2 \rangle \\ \text{POS: } \textit{midfield} \\ \text{TAGR: } \langle 3 \rangle \\ \text{VAGR: } \langle 4 \rangle \end{bmatrix}$$

Das Merkmal VAGR ist in allem Merkmalsstrukturen der Regel mit derselben Variablen $\langle 4 \rangle$ belegt. Damit kann der Anforderungsmechanismus das Hinzufügen des Arguments registrieren. Auch das AGR-Merkmal muß zwischen Verbalphrase und Argument übereinstimmen. Dieses Merkmal des Arguments besitzt jedoch nur werden, ob Verb und Subjekt in Numerus und Person übereinstimmen.

Bildung von Argumenten für Verbalphrasen

Argumente für Verbalphrasen sind alle von der Kategorie `comp`. Dies hat technische Gründe, die mit der Umwandlung von Merkmalen in das Anforderungsmerkmal VAGR zu tun haben. Dies geschieht jeweils mit einer Regel, beispielsweise für das Subjekt:

$$\begin{bmatrix} \text{CAT}: \textit{comp} \\ \text{SUBCAT}: \textit{mid} \\ \text{AGR}: \langle 1 \rangle \\ \text{VAGR}: \begin{bmatrix} \text{SUBJ}: \text{has}\langle 2 \rangle \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT}: \textit{np} \\ \text{SUBCAT}: \textit{nom} \\ \text{AGR}: \langle 1 \rangle \\ \text{SEMCAT}: \langle 2 \rangle \end{bmatrix}$$

Die semantische Kategorie der Nominalphrase SEMCAT wird in das Anforderungsmerkmal VAGR übernommen. Die Nominalphrase muß im Nominativ sein (SUBCAT: *nom*). Die Regeln für die verschiedenen Objektarten sind ähnlich, nur die Übernahme des AGR-Merkmals findet nicht statt.

Die Subkategorie des Arguments beschränkt, in welcher Position des Satzes das Argument eingebaut werden kann. Im Falle des Subjekts ist dies Vor- und Mittelfeld. Im Gegensatz dazu können Satzkomplemente (wie `daß sie glücklich ist`) nur im Vor- und Nachfeld vorkommen, ein Verbkomplement nur am Schluß des Satzes.

Adjunktregeln

Adjunkte werden ähnlich wie Argumente in die Verbalphrase eingebaut. Typische Adjunkte sind Adverben oder Präpositionalphrasen. Folgende Beispielregel behandelt die Aufnahme von Präpositionalphrasen wie `mit deiner Hilfe` im Vorfeld.

$$\begin{bmatrix} \text{CAT}: \textit{vp} \\ \text{SUBCAT}: \langle 1 \rangle \\ \text{AGR}: \langle 2 \rangle \\ \text{POS}: \textit{prefield} \\ \text{TAGR}: \langle 3 \rangle \\ \text{VAGR}: \langle 4 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT}: \textit{pp} \\ \text{SEMCAT}: (\textit{mit}, \langle 5 \rangle) \end{bmatrix} \begin{bmatrix} \text{CAT}: \textit{vp} \\ \text{SUBCAT}: \langle 1 \rangle \\ \text{AGR}: \langle 2 \rangle \\ \text{POS}: \textit{verb}1 \\ \text{TAGR}: \langle 3 \rangle \\ \text{VAGR}: \langle 4 \rangle \end{bmatrix}$$

A.2 Nominalphrasen

Als Nominalphrasen bezeichnet man Konstrukte, die als Subjekt oder Objekt in einem Satz dienen können. Nominalphrasen können auch Teil von Präpositionalphrasen sein. Im folgenden Satz sind zusammenhängende Nominalphrasen unterstrichen:

2 `Hans wohnte mit seiner alten Mutter, die ihn liebte,`
`in einem großen Haus an der Straßengabelung.`

Nominalphrasen können aus einzelnen Worten wie Pronomen oder Namen bestehen. Aber auch komplexere Strukturen sind möglich: Präpositionalphrasen und Relativsätze können angehängt sein. Die häufigsten Nominalphrasen bestehen aus einem Nomen, eventuell vorangestellten Adjektiven und meist einem Artikel. Die folgende Regel faßt ein Nomen und ein Artikel zu einer Nominalphrase zusammen:

$$\begin{bmatrix} \text{CAT: } np \\ \text{SUBCAT: } \langle 1 \rangle \\ \text{AGR: } \langle 2 \rangle \\ \text{SEMCAT: } \langle 4 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT: } det \\ \text{CASE: } \langle 1 \rangle \\ \text{AGR: } \langle 2 \rangle \\ \text{DEF: } \langle 3 \rangle \end{bmatrix} \begin{bmatrix} \text{CAT: } n \\ \text{CASE: } \langle 1 \rangle \\ \text{AGR: } \langle 2 \rangle \\ \text{DEF: } \langle 3 \rangle \\ \text{SEMCAT: } \langle 4 \rangle \end{bmatrix}$$

Der Kasus (Merkmal CASE), der zwischen Artikel und Nomen übereinstimmen muß, wird zur grammatische Subkategorie der Nominalphrase. Dies ist motiviert durch den Umstand, daß der Kasus in Nomina ein Übereinstimmungsmerkmal ist, für die Nominalphrase jedoch ein Bestimmungsmerkmal (siehe dazu 2.1 auf Seite 11).

Das Merkmal DEF gibt die Bestimmtheit des Artikels wieder. Diese ist für die Flexion der Artikel wichtig, wie das folgende Beispiel verdeutlicht:

3 `Der große Mann lacht.`
4 `Ein großer Mann lacht.`

Nomina

In dieser Grammatik werden als Nominalphrasen nur Konstrukte bezeichnet, die selbstständige Komponeten eines Satzes sein können. So ist `großartige Frau` keine Nominalphrase, da ein vorangestellter Artikel oder ein Demonstrativpronomen fehlt. Das genannte Beispiel gilt hier als Nomen. Zur Illustration hier die Regel, die ein Adjektiv als ein Adjunkt zu einem Nomen ermöglicht:

$$\begin{bmatrix} \text{CAT: } n \\ \text{CASE: } \langle 1 \rangle \\ \text{AGR: } \langle 2 \rangle \\ \text{DET: } \langle 3 \rangle \\ \text{DEF: } \langle 4 \rangle \\ \text{SEMCAT: } \langle 5 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT: } adj \\ \text{CASE: } \langle 1 \rangle \\ \text{AGR: } \langle 2 \rangle \\ \text{DEF: } \langle 4 \rangle \end{bmatrix} \begin{bmatrix} \text{CAT: } n \\ \text{CASE: } \langle 1 \rangle \\ \text{AGR: } \langle 2 \rangle \\ \text{DET: } \langle 3 \rangle \\ \text{DEF: } \langle 4 \rangle \\ \text{SEMCAT: } \langle 5 \rangle \end{bmatrix}$$

Das Merkmal DET gibt an, ob ein Artikel zur Vervollständigung des Nomens zu einer Nominalphrase notwendig ist. Einige Nomina wie `Reichtum` oder `Auskunft` brauchen keinen Artikel. In der ersten Regel dieses Abschnittes trat dieses Merkmal nicht auf, da ein Artikel immer möglich ist.

A.3 Präpositionalphrasen

Präpositionalphrasen werden zumeist aus einer Präposition (wie `in`) und einer Nominalphrase (wie `die Stadt`) gebildet. Sie können aber auch mit einem Namen (`in Nürnberg`) oder einem Verb (`mit Umsteigen`) gebildet werden. Insgesamt ist die Bildung von Präpositionalphrasen recht einfach. Ihre Tücken zeigen sie, wenn es um Disambiguierung geht, also die Auflösung von Mehrdeutigkeiten, was zumeist nur auf semantischer Ebene geht. In dieser Grundgrammatik wird die Disambiguierung mittels einer semantischen Kategorie unterstützt, die im folgenden erklärt wird.

A.4 Die Rolle der semantischen Kategorie

An dieser Stelle sei auf die Rolle des Merkmals SEMCAT näher eingegangen. Damit wird die semantische Kategorie von Verben (auch *mood* oder *theme* [All95] genannt), Nomina, Nominalphrasen und Präpositionalphrasen angegeben. Dadurch ist ein gewisser Grad an semantischer Disambiguierung möglich. Betrachten wir dazu die folgenden beiden Sätze:

5 `Ich bin im Zug nach München.`
6 `Ich bin im Klassenzimmer nach München.`

Nach rein grammatischen Kriterien kann nicht unterschieden werden, ob die Präpositionalphrase `nach München` an eine Nominalphrase angehängt werden kann. Sie könnte auch in einem zeitlichen Sinne als Adjunkt zur Verbalphrase verwendet sein. Durch die semantische Kategorie ist eine solche Unterscheidung möglich: Eine Präpositionalphrase mit SEMCAT *(nach,ort)* darf in der vorliegenden Grammatik nur an eine Nominalphrase angehängt werden, wenn dessen SEMCAT mit *zug* besetzt ist.

Nicht jedes Wort hat eine eigenständige semantische Kategorie. Beispielsweise haben in dieser Grammatik die Wörter `Zug`, `Zugverbindung`, `IC-Fahrt` usw. den selben Merkmalswert. Dies ist sinnvoll, wie das obige Beispiel verdeutlicht.

Dennoch ist dieses Vorgehen viel zu vereinfachend. So haben Adjektive auf die semantische Kategorie eines nachfolgenden Nomens in der Grammatik keinen Einfluß. Dies reicht offenkundig nicht aus, wie das folgende Beispiel veranschaulicht:

7 `Ich bin in dem fliegenden Klassenzimmer nach München.`

Semantik ist voller Feinheiten, so daß sie keinesfalls durch ein einzelnes Merkmal erfaßt werden könnte. Dieses Vorgehen würde zu einer schier endlosen Menge an Merkmalswerten und demzufolge Grammatikregeln führen. Dennoch hat es einen gewissen Nutzen: Das Merkmal SEMCAT sortiert die Mehrzahl unsinniger Konstrukte aus.

A.5 Zeitphrasen

Es lassen sich grundsätzlich mindestens drei verschiedene Arten von Zeitangaben unterscheiden:

1. Uhrzeitangaben: `um acht Uhr`
2. Tageszeitangaben: `am Morgen`
3. Tagesangaben: `am Sonntag`

Jede dieser Angaben darf in einer Zeitbeschreibung höchstens einmal vorkommen. Allein mit diesem linguistischen Prinzip, können wir die Zeitphrasen in dem folgenden Satz richtig zuordnen:

8 `Am Samstag ist der Zug um acht Uhr um neun Uhr gefahren.`

In diesem Beispiel ist klar, daß der Zug eine Stunde Verspätung hatte. In der Grammtik ist dies mittels des Merkmals TAGR (*time agreement*) gelöst. TAGR ist eine Substruktur mit den drei Merkmalen DAY (Tagesangabe), TOD (*time of day*, Tageszeitangabe) und TIME (Uhrzeitangabe). Die Substruktur ähnelt dem *verb agreement* VAGR, wobei die Merkmale auf den Merkmalswert *may* initialisiert werden. Jede der Angaben kann also einmal vorkommen, muß aber nicht.

Skurrilitäten

Es gibt noch diverse Eigenarten und Skurrilitäten bei Zeitangaben, wie vielleicht folgendes Beispiel verdeutlicht:

9 `Ich fahre um acht Uhr.`
10 `Ich fahre um acht.`
11 `Ich fahre um zwanzig Uhr.`
12 `* Ich fahre um zwanzig.`

Der letzte Satz klingt ungewohnt, ja falsch. Die Variante ohne `Uhr` ist offenbar nur bei Zeitangaben von ein bis zwölf Uhr anwendbar. Wir wollen an dieser Stelle diese Eigenheiten jedoch nicht weiter vertiefen.

Zusammenfassung von Zeitangaben

Wie können nun Zeitphrasen in einem Satz vorkommen? Betrachten wir die folgenden drei Sätze:

13 `Am Montag um acht Uhr stehe ich auf.`
14 `Am Montag stehe ich um acht Uhr auf.`
15 `Ich stehe am Montag um acht Uhr auf.`

In der Grammatik erlauben wir vor dem Verb nur eine Komponente. Daher müssen wir im ersten Satz `am Montag um acht Uhr` als eine Komponente betrachten. Im zweiten Beispielsatz sind aber die Tages- und Uhrzeitangabe aufgespalten, folglich zwei einzelne Komponenten. Im letzten Beispiel ist schließlich beides möglich. Beides als Möglichkeiten zuzulassen, würde jedoch zu einer völlig unnötigen syntaktischen Ambiguität führen, die im Falle unseres Beispiels zwei Parses erzeugt.

In dieser Grammatik wurde folgender Lösungsweg eingeschlagen: Die einzelnen Zeitphrasen (Kategorie `tp`) werden entweder zu Einzelzeitkomponenten (Kategorie `tc` SUBCAT: *single*), die nur aus einer Zeitphrase bestehen, oder zu Mehrfachzeitkomponenten (Kategorie `tc` SUBCAT: *multiple*), die aus mindestens zwei Zeitphrasen bestehen. Vor dem Verb dürfen nur Mehrfachzeitkomponeten stehen, im Mittelfeld nur Einzelzeitkomponenten. Nominalphrasen dürfen nur Mehrfachzeitkomponenten aufnehmen.

A.6 Nebensätze

In den Haupsatz können Nebensätze in sehr unterschiedlichen Formen eingeschoben sein. In der Grundgrammatik werden sechs verschiedene Nebensatzarten unterschieden (in Klammern ist jeweils kursiv der Merkmalswert des Merkmals SUBCAT angegeben):

1. Hauptsatz (*main*) – Der Nebensatz hat die Grammatik eines normalen Hauptsatzes, eventuell mit vorangestellten Einleitungswort. Beispiel: `Ich verließ sie,` <u>`aber ich liebte sie so sehr`</u>.

2. Klassische Nebensatz (*sub*) – Das Verb befindet sich am Schluß des Nebensatzes. Beispiel: `Ich verließ sie, als ich sie so sehr liebte`.

3. Zu-Konstruktion (*zu*) – Eine Konstruktion, die mit „zu" und einem Verb im Infinitiv abschließt. Dies ist die einzigste Nebensatzkonstruktion, bei der die Valenz des Verbs nicht erfüllt werden muß. So fehlt im folgenden Beispiel das Subjekt: `Ich verließ sie mit der Gewißheit, sie so sehr zu lieben`.

4. Relativsatz (*rel*, *wq*) – Ein Nebensatz in klassischer Nebensatzform, der sich auf eine Nominalphrase in Hauptsatz bezieht. Beispiel: `Ich verließ die Frau, die ich so sehr liebte.`

5. Satzkomplement (*scomp*) – Ein Nebensatz in klassischer Nebensatzform, der vom Verb verlangt wird. Beispiel: `Ich sagte ihr nicht, daß ich sie so sehr liebte`.

6. Konditionale Reihung (*cond*) – Erste Hälfte einer Reihung zweier Halbsätze, die jeweils mit einem Verb beginnen. Beispiel: `Verlasse ich sie jetzt, vergibt sie mit nie.`

7. Satzergänzung (*wq*) – Ein Nebensatz ähnlich einem Relativsatz, der sich jedoch auf kein Nomen, sondern auf den Hauptsatz als ganzes bezieht. Beispiel: `Ich verließ sie aus Liebe, was sie mir nie glauben wird`.

Ein Nebensatz läßt sich in zwei Teile aufspalten: die einleitenden Phrase (oft nur ein Wort wie `aber`), und der Restnebensatz. Abbildung A.3 gibt einen Überblick über die sieben Nebensatztypen in einer etwas formaleren Darstellung.

Abbildung A.3: Nebensätze

Typ	einleitende Phrase	Restnebensatz
main	`part` [SUBCAT:] *main*	Hauptsatzform, POS: *prefield*
sub	`part` [SUBCAT:] *sub*	Nebensatzform, POS: *midfield*
zu	- / `part` [SUBCAT:] *um*	„zu"-Satzform, POS: *midfield*
rel	`rc` [SUBCAT:] *rel*	Nebensatzform, POS: *midfield*
scomp	`part` [SUBCAT:] *scomp*	Nebensatzform, POS: *midfield*
cond	-	Hauptsatzform, POS: *midfield*
wq	`rc` [SUBCAT: *wq*]	Nebensatzform, POS: *midfield*

Relativsätze

Besondere Schwierigkeiten bereiten Relativsätze, da deren einleitende Phrase (hier `rc` *relative component* genannt) Bezüge zum Hauptsatz hat. Diese Bezüge werden mit der Anforderungsstruktur RREQ und der Übereinstimmungsstruktur RAGR formalisiert. An jede Nominalphrase im Hauptsatz kann ein Relativsatz gekoppelt sein. Die Phrase muß nicht dem Relativsatz unmittelbar vorangestellt sein:

16 `Ich fahre mit dem Zug zu meinem Freund nach Nürnberg, der am Samstag um acht Uhr abfährt.`

Aus syntaktischen Gründen könnte sich der Relativsatz sowohl auf `Zug` als auch auf `Freund` beziehen, aus semantischen Gründen bezieht er sich jedoch auf das entferntere Nomen `Zug`.

Das Vorhandensein eines Relativsatzes wird in RREQ vermerkt, indem der Merkmalswert auf *has* gesetzt wird. Die Nominalphrase im Hauptsatz, auf die sich der Relativsatz bezieht, setzt den Merkmalswert auf *need*, was zu *satisfied* unifiziert wird. Die Nominalphrase und das Relativpronomen müssen in Genus und Numerus übereinstimmen: das Merkmal RAGR enthält diese Information.

A.7 Sonstiges

Es gibt eine Reihe von Phrasen, die insbesondere in gesprochener Sprache beliebig in einen Satz gestreut werden können, ohne daß sie irgendetwas zum Informationsgehalt betragen. Dazu gehören Einleitungen wie `grüß Gott` oder `also` sowie Wörter wie `ja` inmitten des Satzes. Sie werden in der Kategorie `junk` zusammengefaßt und können von Verbalphrasen adjungiert werden.

A.8 Grenzen der Grammatik

Die verwendete Grammatik erhebt keinen Anspruch auf linguistische Vollständigkeit. Sie ist jedoch komplex genug, um das Lernverfahren zu testen. Sie ist unvollständig in zweierlei Hinsicht: Einerseits fehlen Regeln für Sonderfälle, andererseits wurden einige grundlegende Sprachkonzepte nicht berücksichtigt. Das Lernverfahren soll ja die Regeln für Sonderfälle konstruieren können.

Es wird jedoch nicht vollständig neue Sprachkonzepte erlernen können, für die mehr Information aus dem Lexikon bereitgestellt werden müßte. Das Verfahren in der hier vorgestellten Form deckt also nicht vollständig die deutsche Sprache ab. Dies ist jedoch kein grundsätzliches Problem: Die entsprechenden Sprachkonzepte können noch implemetiert werden.

Fehlende Sprachkonstrukte

Folgende Sprachkonstrukte sind in dem experimentellen Verfahren nicht berücksichtigt:

Konjunktionen — Konjunktionen wie

```
... und ...
zwischen ... und ...
... oder ...
entweder ... oder ...
```

bedeuten einigen Aufwand für den hier verwendeten Grammatikformalismus: Es können nur maximal binäre Regeln verwendet werden, diese Konstrukte lassen sich jedoch am einfachsten durch Regeln mit drei oder vier Konstituenten auf der rechten Seite realisieren. Das ist kein prinzipielles Problem, da solche Regeln durch mehrere binären Regeln ersetzt werden können. Es bedeutet aber, daß die Grammatik dadurch verkompliziert auf aufgebläht wird.

Zustandsaussagen — Sätze mit dem Verb `sein` wie

```
Der Hund ist tot.
Rot ist grün.
Der Mann ist ein Hund.
Der Hund ist in München.
```

wurden ebenfalls nicht in die Grammatik aufgenommen. Der Einbau dieser Satzform bedarf einer Änderung der Verbanforderungsstruktur VAGR, die nicht allzu schwierig sein sollte. Da Sätze dieser Art in dem Testkorpus kaum vorkamen wurden sie aus Zeitgründen nicht weiter beachtet.

Komparativ, Superlativ — Ebenfalls aus Zeitgründen ignoriert wurden diese Formen des Adjektivs. Deren Berücksichtigung dürfte allerdings keinerlei Probleme bereiten.

Verbzeiten, Konjunktiv — Es wurden keine Merkmale für Verbzeiten oder die Unterscheidung zwischen Konjunktiv und Indikativ eingeführt. Dies war in der verwendeten Domäne nicht notwendig, da alle Sätze im Präsens vorlagen. Eine Hinzufügung dieser Merkmale ist allerdings kein Problem für die Grammatik.

Anhang B

Implementierung

Die implementierte Experimentierumgebung besteht aus einer Vielzahl von Programmen, die in Prolog, C und Perl geschrieben wurden. Prolog wurde für den Chartparser verwendet, da die darin enthalteten Unifikationsmechanismen das Parsen von Sätzen mit Unifikationsgrammatiken erheblich unterstützt. Die mehr statistischen Verfahren wurden C geschrieben, was eine effizientere Programmierung erlaubt. Ebenfalls in C wurden einige Visualisierungsroutinen geschrieben, die TEX-Files als Output hatten. Dadurch ist die Kontrolle von Parsebäumen und -charts beim Erstellen der Grammatik erheblich angenehmer und einfacher möglich. Perl-Skripte runden schließlich das Zusammenspiel der verschiedenen Teile der Experimentierumgebung ab.

Diese Anmerkungen zur Implementierung sollen helfen, den Quellcode verständlicher zu machen, damit er für eigene Projekte verwendet werden kann. Er ist frei verwendbar, wenn eine Referenz zu dieser Diplomarbeit erfolgt.

Bei der Programmierung des Chartparser hat mir insbesondere das Buch „Natural Language Processing for Prolog Programmers" [Cov94] geholfen. Ein anderes Buch zu diesem Thema ist [GM89]. Bei Prolog war [Bra87] und [O'K90] von Hilfe.

Der Code gliedert sich in drei Segmente: Der Chartparser, die Entwicklungsumgebung zur Erstellung der Grundgrammatik, des Lexikons und des Grammatikmodells, das Lernverfahren sowie die Experimentierumgebung.

B.1 Chartparser

Files:

```
chart.pl
rules.pl
lexicon.pl
input_ok.pl
gm.pl
```

Das Herz des Lernverfahrens ist der Chartparser, der sich im File `chart.pl` befindet. Er folgt stark Vorschlägen von Michael Covington [Cov94]. Es handelt sich dabei ein bottom up Parser, der ohne aktive Kanten arbeitet.

Das Parsen eines Satzes wird mit der Funktion `?- parse(`*Satznummer*`).` angestoßen. Der Parser erzeugt nun Charteinträge in der Form:

```
chart(Kategorie, Merkmalsstruktur, Anfangsposition, Endposition, Regel,
      Kante1, Kante2, Nummer).
```

Nummer ist dabei die fortlaufende Nummer des Charteintrags. *Regel* gibt an, mit welcher Regel der Eintrag aus den Kanten *Kante1* und *Kante2* gewonnen wurde. Diese Angaben sind nicht notwendig für das Parsen, helfen aber einen Überblick über den Parsevorgang zu erhalten.

Die *Anfangsposition* und *Endposition* geben an, welchen Bereich des Satzes der Charteintrag überdeckt. Schließlich bleiben die *Kategorie* und die *Merkmalsstruktur*. Die Merkmalsstruktur ist eine Liste mit den einzelnen Merkmalen und einem zusätzlichen SEM-Merkmal, das der Parsebaum für diesen Charteintrag enthält.

Die Ausgabe des Parsekommandos sind die vollständigen Parses, also Charteinträge von der Kategorie `s`, die sich vom Anfang bis zum Ende des Satzes erstrecken.

Korpus

Das Korpus befindet sich in dem File `input_ok.pl`. Die Sätze bestehen dabei aus einem Paar: der Satznummer und der Liste der Wörter im Satz, also zum Beispiel:

```
input(499,[ich,moechte,gerne,nach,paris]).
```

Lexikon

Das Lexikon befindet sich in dem File `lexicon.pl`. Teile des Lexikons sind automatisch von der Morphologiekomponente erzeugt. Ein Lexikoneintrag hat die Form

```
lexicon(Wörter, Kategorie, Merkmalsliste).
```

Der erste Parameter ist entweder ein einzelnes Wort oder eine Liste von Wörter, die die selben syntaktischen und (grob) semantischen Eigenschaften haben. Die Merkmalsliste besteht aus Paaren von Merkmalsnamen und Merkmalswerten. Diese Form ist wesentlich einfacher zu lesen als die der Merkmalsstrukturen. Ein Beispiel:

```
lexicon([fahre,komme,geht,halte,laufe],
   v,[subcat:fin,semcat:bewegen,agr:[number:sing],person:first,
      vagr:[subj:[need,_]]]).
```

In der Merkmalsliste müssen nicht alle Merkmale vorkommen, sondern nur jene, denen ein Wert zugewiesen wird. Dies macht diese Angaben kompakter und übersichtlicher. Die Lexikoneinträge werden von der Funktion `compile_lexicon` (in der Datei `chart.pl`) in ein Format umgewandelt, das der Funktionsweise des Chartparsers mehr entgegenkommt:

```
wordform(Wort, [Kategorie, Merkmalsstruktur], Nummer).
```

Grammatik

Die Grammatik befindet sich in dem File `rules.pl`. Das Format der Regeln ähnelt stark den Lexikoneinträgen:

`grammar([`*Kategorie der linken Seite*`,` *Merkmalsliste der linken Seite*`],`
`[[`*erste Kategorie der rechten Seite*`,` *erste Merkmalsliste der rechten Seite*`],`
`[`*zweite Kategorie der rechten Seite*`,` *zweite Merkmalsliste der rechten Seite*`]]).`

Wie auch beim Lexikon werden die Regeln durch eine Funktion, `compile_grammar`, für den Chartparser in ein passenderes Format umgewandelt. Das resultierende Format der Regeln (`rule()`) ist analog zu dem der Lexikoneinträge.

B.2 Entwicklungsumgebung

Zusätzliche Files:

```
p
pretty.c
nowords.pl
input.pl
morph.pl
lexicon_add.pl
n
```

Um die Entwicklung der Grammatik zu unterstützen, wurde ein Tool geschrieben, daß die doch recht kryptischen Prologterme für Regeln und Charteinträge in ein visuell ansprechendes Format umwandelt. Dies geschiet mit Hilfe von LATEX: Das Tool erzeugt Dateien in einem Format, das von LATEXweiterverarbeitet werden kann, und schließlich in Postscript umgewandelt wird, so daß es mittels des Postscriptviewers Ghostview betrachtet werden kann.

Die Umwandlungsroutine wurde in C geschrieben und befindet sich in dem File `pretty.c`, kompiliert zu `pretty`. Um die oben beschriebenen Umwandlungen zu realisieren, gibt es das Shell-Skript mit dem schlichten Namen `p`.

Bei einem Aufruf des Skripts ohne zusätzliche Parameter wird der Chart des letzten durchgeführten Parse dargestellt. Dies ist möglich, weil während des Parsens die Datei `chart.log` angelegt wurde, die die Charteinträge mit zusätzlichen Informationen enthält. Das Skript erzeugt ein Dokument, das alle Charteinträge mit allen hilfreichen Daten, die Bäume vollständiger Parse und eine graphische Übersichtsdarstellung aller Kanten enthält.

Mit der Option `t` kann eine Titelzeile angegeben werden, die im Dokument neben dem Datum erscheint. Zusätzlich werden alle Darstellungen durchnummeriert und in der Datei `parse.inf` geloggt.

Beschränkung des Korpus

Da bestimmte grammatikalische Phänomene von vornherein ausgeschlossen werden, kann nicht das gesamte Korpus an anfänglich vorgegebenen Sätzen betrachtet werden. Mit Hilfe der Funktion `filter` (in `chart.pl`) werden Sätze, die Un-Wörter enthalten aussortiert. Das ursprüngliche Korpus input.pl wird so in den verwendeten Korpus `input-ok.pl` und die übrigen Sätze `input-noword.pl` aufgespalten. Die Datei `nowords.pl` enthält die Wörter, die ausgeschlossen werden.

Entwicklung des Lexikons

Die Entwicklung des Lexikons ist teilweise durch eine Morphologiekomponente unterstützt. Diese ist in Prolog geschrieben und befindet sich im File `morph.pl`. Für Adjektive und Nomen werden Wortformen mit den entsprechenden Endungen gebildet und in die Datei `lexicon_gen.pl` gespeichert. Für die Endungen von Nomen wurde auf die Deklinationstabellen aus dem Duden zurückgegriffen.

Es wäre zukünftig wünschenswert, die Morphologiekomponente auf Verben auszuweiten. Dies war bei dem verwendeten Korpus jedoch nicht nötig, da fast ausschließlich Verben im Präsens, erste oder dritte Person Singular vorkamen. Verben und alle übrigen Lexikoneinträge wurden daher von Hand in das File `lexicon-add.pl` geschrieben. Das Perl-Skript `cl` faßt sie mit den automatisch generierten Lexikoneinträgen aus `lexicon-gen.pl` in das Lexikonfile `lexicon.pl` zusammen.

Für die Entwicklung des Lexikons ist die Routine `wordcheck(`*Startsatz*`,`*Endsatz*`)` (im File `chart.pl`) wertvoll: Sie sucht alle Wortformen ohne Lexikoneintrag aus dem Korpus heraus und gibt sie auf dem Bildschirm aus.

Entwicklung der Grammatik

Bei der Entwicklung der Grammatik ist das Visualisierungstool `p` hilfreich. Mit der Funktion `report_rules.` in `chart.pl` wird die Datei `report.inf` geschrieben, die von dem Tool gelesen werden kann. Bei einem Aufruf mit dem Parameter `r` gibt es die Regeln in graphischer Form aus. Die Regeln sind zudem durchnummeriert, was deren Verwendung in Parses klarer erkennen läßt. Mit Hilfe des Perl-Skiptes `n` wird diese Nummerierung auch in das File `rules.pl` eingetragen, in dem die Grammatik erstellt wird.

B.3 Lernverfahren

Zusätzliche Files:

```
gm.pl
parse
learn
statistics.c
statistics
eval
```

Das Grammatikmodell für das Lernverfahren befindet sich in dem File `gm.pl`. Es besteht zum einen aus Festlegungen für die Merkmalen und zum anderen aus Beschränkungen zur Art der lernbaren Regeln. So muß für jede Kategorie festgelegt werden, welche Merkmale sie haben kann (vergleiche Tabelle 2.2 auf Seite 11). Dies geschieht in der Form:

`req_feature(`*Kategorie*`,` *Liste von Merkmalen*`).`

wobei die Liste einfach aus den Namen der Merkmalen besteht. Zu jedem Merkmal muß nun angegeben sein, ob Merkmalswerte in neu konstruierte Regeln aufgenommen werden (Vergleiche Seite 38 und Tabelle 2.1 auf Seite 11). Wie dies dem Grammatikmodell mitgeteilt wird, illustrieren die beiden folgenden Beispiele. Bei SUBCAT wird der Merkmalswert in neukonstruierte Regeln übernommen, bei AGR nur die Übereinstimmung festgelegt.

```
feature_pass(subcat,A,A).
feature_pass(agr,_,_).
```

Das Format für Beschränkungen unterscheidet sich zwischen Argument- und Adjunktregeln einerseits und Kompositions- und Überführungsregeln andererseits. Im ersten Fall geschieht dies in folgender Form:

`adj(`*Position*`,` *Kopfkategorie*`,` *Merkmalsliste*`,` *Kategorien*`).`
`arg(`*Position*`,` *Kopfkategorie*`,` *Merkmalsliste*`,` *Kategorien*`).`

Position gibt dabei an, ob das Adjunkt bzw. Argument links oder rechts aufgenommen wird. Zu der *Kopfkategorie* können noch zusätzliche Werte von Merkmalen in einer *Merkmalsliste* angegeben werden, die die möglichen Regelkonstruktionen beschränken. Schließlich wird festgelegt, welche *Kategorien* Adjunkte bzw. Argumente haben können.

Für Kompositions- und Überführungsregeln werden Musterregeln angegeben, deren Form der der Regeln der Grundgrammatik entspricht. Der einzige Unterschied ist, daß die Prädikate `lc()` und `tf()` statt `grammar()` heißen.

Ablauf des Lernverfahrens

Das Lernverfahren ist in drei Schritte aufgeteilt: das Parsen der Sätze, das Konstruieren neuer Regeln und die statistische Auswertung der Regeln. In dem Unterverzeichnis `Experimente` werden Dateien abgelegt, die die angefallenen Daten zwischen diesen Schritte festhalten.

Das Parsen aller Sätze in der Datei `input_ok.pl` wird mit dem Perl-Skript `parse` gestartet. Während des Ablaufs werden in dem Unterverzeichnis `Experimente` folgende Dateien angelegt:

- `to_be_parsed.inf` enthält die Satznummern der Sätze, die geparsed werden sollen.
- `parses.inf.`*Satznummer* enthält für jeden Satz die Satznummer, die Anzahl der Charteinträge nach dem Parsen, die Länge des Satzes in Wörtern und die Parsebäume im Format des SEM-Merkmals.
- `to_be_learned.inf` enthält die Sätznummern der Sätze, die nicht geparsed werden konnten und anschließend gelernt werden müssen.

Das Konstruieren neuer Regeln für die Sätze in `to_be_learned.inf` wird mit dem Skript `learn` ausgeführt. Dies ist der zeitaufwendigste Teil des Lernverfahrens. Wiederum werden eine Reihe von Dateien während des Ablaufs in dem Unterverzeichnis `Experimente` angelegt:

- `parses.learn.`*Satznummer* enthält für jeden Satz die Satznummer, die Nummer der ersten neukonstruierte Regel (immer 501), die Anzahl der Charteinträge nach dem letzten Parsen und die Parsebäume.
- `rules.inf.`*Satznummer* enthält die vorgeschlagenen Regeln für den jeweiligen Satz.

Schließlich führt das Skript `good_rules` die statistische Auswertung der vorgeschlagenen Regeln aus. Es behilft sich dabei des C-Programmes `statistics.c` (kompiliert zu `statistics`). Als Ausgabe erzeugt es für jede Iteration des Lernverfahrens eine Datei `good_rules_eval.`*Schritt* im Unterverzeichnis `Experimente` an, die für jeden Satz die Regelnummer und Bewertung enthält.

B.4 Experimentierumgebung

Zusätzliche Files:

```
learn_fast
run_learn_fast
stat
ev
```

Die Konstruktion neuer Regeln ist der zeitaufwendigste Schritt des Lernverfahrens. Dies bereitet insbesondere Schwierigkeiten, da er für einige Sätze zwar nur Sekunden dauert, für andere, längere Sätze hingegen Tage. Um in kurzer Zeit zumindest einen Einschätzung über das Verhalten des Lernverfahrens in diesem Schritt zu erhalten, gibt es das Skript `learn_fast`. Diesem wird ein Parameter übergeben, der die Rechenzeit für einen Satz auf eine Höchstdauer in Sekunden beschränkt. Falls diese Zeitspanne überschritten wird, bricht der Prozess ab und es wird der nächste Satz bearbeitet.

Mit dem Skript `run_learn_fast` kann die Konstruktion von Regeln für einen einzelnen Satz ausgeführt werden. Als Paramater muß die Satznummer und die maximale Anzahl von Sekunden für die Dauer des Prozesses angegeben werden. Anschließend ist ein genaueres Betrachten der Chart nach dem Lernen mit `p c` und der gelernten Regeln mit `p r` möglich.

Wie sich das Lernverfahren auf einzelne Sätze auswirkt, kann auch mit den Skripten `stat` und `ev` ermittelt werden. Mit der Satznummer als Parameter geben sie aus, welche der vorgeschlagenen Regeln von welchen Sätzen verwendet werden. `stat` gibt dazu noch die guten Regeln nach der statistischer Auswertung im Satzkontext aus. `ev` liefert die statistische Auswertung nach einer Iteration der statistischen Auswertung im Korpuskontext.

Anhang C

Alle Regeln der Grundgrammatik

$$1 \bullet \texttt{s} \rightarrow \texttt{vp}$$

$$\begin{bmatrix} \textsc{subcat}: \langle 1 \rangle \\ \textsc{person}: \langle 3 \rangle \\ \textsc{agr}: \langle 4 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \langle 1 \rangle \\ \textsc{person}: \langle 3 \rangle \\ \textsc{agr}: \langle 4 \rangle \\ \textsc{pos}: \mathit{prefield} \\ \textsc{rreq}: \mathit{complete} \\ \textsc{vagr}: \mathit{complete} \end{bmatrix}$$

$$2 \bullet \texttt{s} \rightarrow \texttt{vp}$$

$$\begin{bmatrix} \textsc{subcat}: \langle 1 \rangle \\ \textsc{person}: \langle 3 \rangle \\ \textsc{agr}: \langle 4 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \langle 1 \rangle \\ \textsc{person}: \langle 3 \rangle \\ \textsc{agr}: \langle 4 \rangle \\ \textsc{pos}: \mathit{verb} \\ \textsc{rreq}: \mathit{complete} \\ \textsc{vagr}: \mathit{complete} \end{bmatrix}$$

$$3 \bullet \texttt{vp} \rightarrow \texttt{junk}\ \texttt{vp}$$

$$\begin{bmatrix} \textsc{subcat}: \langle 1 \rangle \\ \textsc{person}: \langle 3 \rangle \\ \textsc{agr}: \langle 4 \rangle \\ \textsc{pos}: \mathit{prefield} \\ \textsc{rreq}: \langle 8 \rangle \\ \textsc{ragr}: \langle 9 \rangle \\ \textsc{tagr}: \langle 10 \rangle \\ \textsc{vagr}: \langle 11 \rangle \\ \textsc{semcat}: \langle 12 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \mathit{init} \end{bmatrix} \begin{bmatrix} \textsc{subcat}: \langle 1 \rangle \\ \textsc{person}: \langle 3 \rangle \\ \textsc{agr}: \langle 4 \rangle \\ \textsc{pos}: \mathit{prefield} \\ \textsc{rreq}: \langle 8 \rangle \\ \textsc{ragr}: \langle 9 \rangle \\ \textsc{tagr}: \langle 10 \rangle \\ \textsc{vagr}: \langle 11 \rangle \\ \textsc{semcat}: \langle 12 \rangle \end{bmatrix}$$

$4 \bullet \texttt{vp} \rightarrow \texttt{sc vp}$

$$\begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \langle 3\rangle \\ \text{AGR}: \langle 4\rangle \\ \text{POS}: \mathit{prefield} \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{VAGR}: \langle 11\rangle \\ \text{SEMCAT}: \langle 12\rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{SUBCAT}: \mathit{sub} \end{bmatrix} \begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \langle 3\rangle \\ \text{AGR}: \langle 4\rangle \\ \text{POS}: \mathit{verb} \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{VAGR}: \langle 11\rangle \\ \text{SEMCAT}: \langle 12\rangle \end{bmatrix}$$

$5 \bullet \texttt{vp} \rightarrow \texttt{sc vp}$

$$\begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \langle 3\rangle \\ \text{AGR}: \langle 4\rangle \\ \text{POS}: \mathit{prefield} \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{VAGR}: \langle 11\rangle \\ \text{SEMCAT}: \langle 12\rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{SUBCAT}: \mathit{zu} \end{bmatrix} \begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \langle 3\rangle \\ \text{AGR}: \langle 4\rangle \\ \text{POS}: \mathit{verb} \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{VAGR}: \langle 11\rangle \\ \text{SEMCAT}: \langle 12\rangle \end{bmatrix}$$

$6 \bullet \texttt{vp} \rightarrow \texttt{sc vp}$

$$\begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \langle 3\rangle \\ \text{AGR}: \langle 4\rangle \\ \text{POS}: \mathit{prefield} \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{VAGR}: \langle 11\rangle \\ \text{SEMCAT}: \langle 12\rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{SUBCAT}: \mathit{cond} \end{bmatrix} \begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \langle 3\rangle \\ \text{AGR}: \langle 4\rangle \\ \text{POS}: \mathit{verb} \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{VAGR}: \langle 11\rangle \\ \text{SEMCAT}: \langle 12\rangle \end{bmatrix}$$

$7 \bullet \texttt{vp} \rightarrow \texttt{rc vp}$

$$\begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \langle 3\rangle \\ \text{AGR}: \langle 4\rangle \\ \text{POS}: \mathit{prefield} \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{VAGR}: \langle 11\rangle \\ \text{SEMCAT}: \langle 12\rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{SUBCAT}: \mathit{wq} \\ \text{PERSON}: \langle 3\rangle \\ \text{AGR}: \langle 4\rangle \\ \text{VAGR}: \langle 11\rangle \end{bmatrix} \begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \langle 3\rangle \\ \text{AGR}: \langle 4\rangle \\ \text{POS}: \mathit{verb} \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{VAGR}: \langle 11\rangle \\ \text{SEMCAT}: \langle 12\rangle \end{bmatrix}$$

$8 \bullet$ `vp` $\rightarrow$ `comp vp`

$$\begin{bmatrix} \textsc{subcat}: \langle 1 \rangle \\ \textsc{person}: \langle 3 \rangle \\ \textsc{agr}: \langle 4 \rangle \\ \textsc{pos}: \mathit{prefield} \\ \textsc{rreq}: \langle 8 \rangle \\ \textsc{ragr}: \langle 9 \rangle \\ \textsc{tagr}: \langle 10 \rangle \\ \textsc{vagr}: \langle 11 \rangle \\ \textsc{semcat}: \langle 12 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{person}: \langle 3 \rangle \\ \textsc{agr}: \langle 4 \rangle \\ \textsc{rreq}: \langle 8 \rangle \\ \textsc{ragr}: \langle 9 \rangle \\ \textsc{vagr}: \langle 11 \rangle \end{bmatrix} \begin{bmatrix} \textsc{subcat}: \langle 1 \rangle \\ \textsc{person}: \langle 3 \rangle \\ \textsc{agr}: \langle 4 \rangle \\ \textsc{pos}: \mathit{verb} \\ \textsc{rreq}: \langle 8 \rangle \\ \textsc{ragr}: \langle 9 \rangle \\ \textsc{tagr}: \langle 10 \rangle \\ \textsc{vagr}: \langle 11 \rangle \\ \textsc{semcat}: \langle 12 \rangle \end{bmatrix}$$

$9 \bullet$ `vp` $\rightarrow$ `tc vp`

$$\begin{bmatrix} \textsc{subcat}: \langle 1 \rangle \\ \textsc{person}: \langle 3 \rangle \\ \textsc{agr}: \langle 4 \rangle \\ \textsc{pos}: \mathit{prefield} \\ \textsc{rreq}: \langle 8 \rangle \\ \textsc{ragr}: \langle 9 \rangle \\ \textsc{tagr}: \langle 10 \rangle \\ \textsc{vagr}: \langle 11 \rangle \\ \textsc{semcat}: \langle 12 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{tagr}: \langle 10 \rangle \end{bmatrix} \begin{bmatrix} \textsc{subcat}: \langle 1 \rangle \\ \textsc{person}: \langle 3 \rangle \\ \textsc{agr}: \langle 4 \rangle \\ \textsc{pos}: \mathit{verb} \\ \textsc{rreq}: \langle 8 \rangle \\ \textsc{ragr}: \langle 9 \rangle \\ \textsc{tagr}: \langle 10 \rangle \\ \textsc{vagr}: \langle 11 \rangle \\ \textsc{semcat}: \langle 12 \rangle \end{bmatrix}$$

$10 \bullet$ `vp` $\rightarrow$ `junk vp`

$$\begin{bmatrix} \textsc{subcat}: \langle 1 \rangle \\ \textsc{person}: \langle 3 \rangle \\ \textsc{agr}: \langle 4 \rangle \\ \textsc{pos}: \mathit{verb} \\ \textsc{rreq}: \langle 8 \rangle \\ \textsc{ragr}: \langle 9 \rangle \\ \textsc{tagr}: \langle 10 \rangle \\ \textsc{vagr}: \langle 11 \rangle \\ \textsc{semcat}: \langle 12 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \mathit{init} \end{bmatrix} \begin{bmatrix} \textsc{subcat}: \langle 1 \rangle \\ \textsc{person}: \langle 3 \rangle \\ \textsc{agr}: \langle 4 \rangle \\ \textsc{pos}: \mathit{verb} \\ \textsc{rreq}: \langle 8 \rangle \\ \textsc{ragr}: \langle 9 \rangle \\ \textsc{tagr}: \langle 10 \rangle \\ \textsc{vagr}: \langle 11 \rangle \\ \textsc{semcat}: \langle 12 \rangle \end{bmatrix}$$

$11 \bullet$ `vp` $\rightarrow$ `v vp`

$$\begin{bmatrix} \textsc{subcat}: \mathit{fin} \\ \textsc{person}: \langle 2 \rangle \\ \textsc{agr}: \langle 3 \rangle \\ \textsc{pos}: \mathit{verb} \\ \textsc{rreq}: \langle 7 \rangle \\ \textsc{ragr}: \langle 8 \rangle \\ \textsc{tagr}: \langle 9 \rangle \\ \textsc{vagr}: \langle 10 \rangle \\ \textsc{semcat}: \langle 11 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \mathit{fin} \\ \textsc{person}: \langle 2 \rangle \\ \textsc{agr}: \langle 3 \rangle \\ \textsc{vagr}: \langle 10 \rangle \\ \textsc{semcat}: \langle 11 \rangle \end{bmatrix} \begin{bmatrix} \textsc{subcat}: \mathit{fin} \\ \textsc{person}: \langle 2 \rangle \\ \textsc{agr}: \langle 3 \rangle \\ \textsc{pos}: \mathit{midfield} \\ \textsc{rreq}: \langle 7 \rangle \\ \textsc{ragr}: \langle 8 \rangle \\ \textsc{tagr}: \langle 9 \rangle \\ \textsc{vagr}: \langle 10 \rangle \\ \textsc{semcat}: \langle 11 \rangle \end{bmatrix}$$

---

12 • `vp → v vp`

$$\begin{bmatrix} \text{SUBCAT}: \mathit{aux} \\ \text{PERSON}: \langle 2\rangle \\ \text{AGR}: \langle 3\rangle \\ \text{POS}: \mathit{verb} \\ \text{RREQ}: \langle 7\rangle \\ \text{RAGR}: \langle 8\rangle \\ \text{TAGR}: \langle 9\rangle \\ \text{VAGR}: \langle 10\rangle \\ \text{SEMCAT}: \langle 11\rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{SUBCAT}: \mathit{aux} \\ \text{PERSON}: \langle 2\rangle \\ \text{AGR}: \langle 3\rangle \end{bmatrix} \begin{bmatrix} \text{SUBCAT}: \mathit{aux} \\ \text{PERSON}: \langle 2\rangle \\ \text{AGR}: \langle 3\rangle \\ \text{POS}: \mathit{midfield} \\ \text{RREQ}: \langle 7\rangle \\ \text{RAGR}: \langle 8\rangle \\ \text{TAGR}: \langle 9\rangle \\ \text{VAGR}: \langle 10\rangle \\ \text{SEMCAT}: \langle 11\rangle \end{bmatrix}$$

---

13 • `vp → adv vp`

$$\begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \langle 3\rangle \\ \text{AGR}: \langle 4\rangle \\ \text{POS}: \mathit{midfield} \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{VAGR}: \langle 11\rangle \\ \text{SEMCAT}: \langle 12\rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{SEMCAT}: \mathit{gerne} \end{bmatrix} \begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \langle 3\rangle \\ \text{AGR}: \langle 4\rangle \\ \text{POS}: \mathit{midfield} \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{VAGR}: \langle 11\rangle \\ \text{SEMCAT}: \langle 12\rangle \end{bmatrix}$$

---

14 • `vp → adv vp`

$$\begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \langle 3\rangle \\ \text{AGR}: \langle 4\rangle \\ \text{POS}: \mathit{midfield} \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{VAGR}: \langle 11\rangle \\ \text{SEMCAT}: \langle 12\rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{SEMCAT}: \mathit{auch} \end{bmatrix} \begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \langle 3\rangle \\ \text{AGR}: \langle 4\rangle \\ \text{POS}: \mathit{midfield} \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{VAGR}: \langle 11\rangle \\ \text{SEMCAT}: \langle 12\rangle \end{bmatrix}$$

---

15 • `vp → comp vp`

$$\begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \langle 3\rangle \\ \text{AGR}: \langle 4\rangle \\ \text{POS}: \mathit{midfield} \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{VAGR}: \langle 11\rangle \\ \text{SEMCAT}: \langle 12\rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{SUBCAT}: \mathit{mid} \\ \text{PERSON}: \langle 3\rangle \\ \text{AGR}: \langle 4\rangle \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{VAGR}: \langle 11\rangle \end{bmatrix} \begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \langle 3\rangle \\ \text{AGR}: \langle 4\rangle \\ \text{POS}: \mathit{midfield} \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{VAGR}: \langle 11\rangle \\ \text{SEMCAT}: \langle 12\rangle \end{bmatrix}$$

$16 \bullet \texttt{vp} \rightarrow \texttt{tc vp}$

$$\begin{bmatrix} \textsc{subcat}: \langle 1\rangle \\ \textsc{person}: \langle 3\rangle \\ \textsc{agr}: \langle 4\rangle \\ \textsc{pos}: \mathit{midfield} \\ \textsc{rreq}: \langle 8\rangle \\ \textsc{ragr}: \langle 9\rangle \\ \textsc{tagr}: \langle 10\rangle \\ \textsc{vagr}: \langle 11\rangle \\ \textsc{semcat}: \langle 12\rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \mathit{single} \\ \textsc{tagr}: \langle 10\rangle \end{bmatrix} \begin{bmatrix} \textsc{subcat}: \langle 1\rangle \\ \textsc{person}: \langle 3\rangle \\ \textsc{agr}: \langle 4\rangle \\ \textsc{pos}: \mathit{midfield} \\ \textsc{rreq}: \langle 8\rangle \\ \textsc{ragr}: \langle 9\rangle \\ \textsc{tagr}: \langle 10\rangle \\ \textsc{vagr}: \langle 11\rangle \\ \textsc{semcat}: \langle 12\rangle \end{bmatrix}$$

$17 \bullet \texttt{vp} \rightarrow \texttt{pp vp}$

$$\begin{bmatrix} \textsc{subcat}: \langle 1\rangle \\ \textsc{person}: \langle 3\rangle \\ \textsc{agr}: \langle 4\rangle \\ \textsc{pos}: \mathit{midfield} \\ \textsc{rreq}: \langle 8\rangle \\ \textsc{ragr}: \langle 9\rangle \\ \textsc{tagr}: \langle 10\rangle \\ \textsc{vagr}: \langle 11\rangle \\ \textsc{semcat}: \mathit{bewegen} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{rreq}: \langle 8\rangle \\ \textsc{ragr}: \langle 9\rangle \\ \textsc{semcat}: (\mathit{nach}, \mathit{ort}) \end{bmatrix} \begin{bmatrix} \textsc{subcat}: \langle 1\rangle \\ \textsc{person}: \langle 3\rangle \\ \textsc{agr}: \langle 4\rangle \\ \textsc{pos}: \mathit{midfield} \\ \textsc{rreq}: \langle 8\rangle \\ \textsc{ragr}: \langle 9\rangle \\ \textsc{tagr}: \langle 10\rangle \\ \textsc{vagr}: \langle 11\rangle \\ \textsc{semcat}: \mathit{bewegen} \end{bmatrix}$$

$18 \bullet \texttt{vp} \rightarrow \texttt{pp vp}$

$$\begin{bmatrix} \textsc{subcat}: \langle 1\rangle \\ \textsc{person}: \langle 3\rangle \\ \textsc{agr}: \langle 4\rangle \\ \textsc{pos}: \mathit{midfield} \\ \textsc{rreq}: \langle 8\rangle \\ \textsc{ragr}: \langle 9\rangle \\ \textsc{tagr}: \langle 10\rangle \\ \textsc{vagr}: \langle 11\rangle \\ \textsc{semcat}: \mathit{bewegen} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{rreq}: \langle 8\rangle \\ \textsc{ragr}: \langle 9\rangle \\ \textsc{semcat}: (\mathit{von}, \mathit{ort}) \end{bmatrix} \begin{bmatrix} \textsc{subcat}: \langle 1\rangle \\ \textsc{person}: \langle 3\rangle \\ \textsc{agr}: \langle 4\rangle \\ \textsc{pos}: \mathit{midfield} \\ \textsc{rreq}: \langle 8\rangle \\ \textsc{ragr}: \langle 9\rangle \\ \textsc{tagr}: \langle 10\rangle \\ \textsc{vagr}: \langle 11\rangle \\ \textsc{semcat}: \mathit{bewegen} \end{bmatrix}$$

$19 \bullet \texttt{vp} \rightarrow \texttt{pp vp}$

$$\begin{bmatrix} \textsc{subcat}: \langle 1\rangle \\ \textsc{person}: \langle 3\rangle \\ \textsc{agr}: \langle 4\rangle \\ \textsc{pos}: \mathit{midfield} \\ \textsc{rreq}: \langle 8\rangle \\ \textsc{ragr}: \langle 9\rangle \\ \textsc{tagr}: \langle 10\rangle \\ \textsc{vagr}: \langle 11\rangle \\ \textsc{semcat}: \langle 12\rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{rreq}: \langle 8\rangle \\ \textsc{ragr}: \langle 9\rangle \\ \textsc{semcat}: (\mathit{mit}, \langle\rangle) \end{bmatrix} \begin{bmatrix} \textsc{subcat}: \langle 1\rangle \\ \textsc{person}: \langle 3\rangle \\ \textsc{agr}: \langle 4\rangle \\ \textsc{pos}: \mathit{midfield} \\ \textsc{rreq}: \langle 8\rangle \\ \textsc{ragr}: \langle 9\rangle \\ \textsc{tagr}: \langle 10\rangle \\ \textsc{vagr}: \langle 11\rangle \\ \textsc{semcat}: \langle 12\rangle \end{bmatrix}$$

---

20 • vp → comp

$$\begin{bmatrix} \text{SUBCAT}: \textit{fin} \\ \text{PERSON}: \langle 2 \rangle \\ \text{AGR}: \langle 3 \rangle \\ \text{POS}: \textit{midfield} \\ \text{RREQ}: \langle 7 \rangle \\ \text{RAGR}: \langle 8 \rangle \\ \text{TAGR}: \begin{bmatrix} \text{DAY}: \text{may} \\ \text{TOD}: \text{may} \\ \text{TIME}: \text{may} \end{bmatrix} \\ \text{VAGR}: \langle 12 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{PERSON}: \langle 2 \rangle \\ \text{AGR}: \langle 3 \rangle \\ \text{RREQ}: \langle 7 \rangle \\ \text{RAGR}: \langle 8 \rangle \\ \text{VAGR}: \langle 12 \rangle \end{bmatrix}$$

---

21 • vp → pp

$$\begin{bmatrix} \text{SUBCAT}: \textit{fin} \\ \text{POS}: \textit{midfield} \\ \text{RREQ}: \langle 7 \rangle \\ \text{RAGR}: \langle 8 \rangle \\ \text{TAGR}: \begin{bmatrix} \text{DAY}: \text{may} \\ \text{TOD}: \text{may} \\ \text{TIME}: \text{may} \end{bmatrix} \\ \text{SEMCAT}: \textit{bewegen} \end{bmatrix} \rightarrow \begin{bmatrix} \text{RREQ}: \langle 7 \rangle \\ \text{RAGR}: \langle 8 \rangle \\ \text{SEMCAT}: (\textit{nach}, \textit{ort}) \end{bmatrix}$$

---

22 • vp → sc

$$\begin{bmatrix} \text{SUBCAT}: \textit{fin} \\ \text{POS}: \textit{midfield} \\ \text{TAGR}: \begin{bmatrix} \text{DAY}: \text{may} \\ \text{TOD}: \text{may} \\ \text{TIME}: \text{may} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \text{SUBCAT}: \textit{main} \end{bmatrix}$$

---

23 • vp → sc

$$\begin{bmatrix} \text{SUBCAT}: \textit{fin} \\ \text{POS}: \textit{midfield} \\ \text{TAGR}: \begin{bmatrix} \text{DAY}: \text{may} \\ \text{TOD}: \text{may} \\ \text{TIME}: \text{may} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \text{SUBCAT}: \textit{sub} \end{bmatrix}$$

---

24 • vp → sc

$$\begin{bmatrix} \text{SUBCAT}: \textit{fin} \\ \text{POS}: \textit{midfield} \\ \text{TAGR}: \begin{bmatrix} \text{DAY}: \text{may} \\ \text{TOD}: \text{may} \\ \text{TIME}: \text{may} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \text{SUBCAT}: \textit{zu} \end{bmatrix}$$

---

25 • `vp` → `sc`

$$\left[\begin{array}{ll}\textsc{subcat}: & \mathit{fin} \\ \textsc{pos}: & \mathit{midfield} \\ \textsc{tagr}: & \left[\begin{array}{ll}\textsc{day}: & \text{may} \\ \textsc{tod}: & \text{may} \\ \textsc{time}: & \text{may}\end{array}\right] \\ \textsc{vagr}: & \left[\begin{array}{ll}\textsc{scomp}: & \text{has}\end{array}\right]\end{array}\right] \rightarrow \left[\begin{array}{ll}\textsc{subcat}: & \mathit{scomp}\end{array}\right]$$

---

26 • `vp` → `sc`

$$\left[\begin{array}{ll}\textsc{subcat}: & \mathit{fin} \\ \textsc{pos}: & \mathit{midfield} \\ \textsc{tagr}: & \left[\begin{array}{ll}\textsc{day}: & \text{may} \\ \textsc{tod}: & \text{may} \\ \textsc{time}: & \text{may}\end{array}\right] \\ \textsc{vagr}: & \left[\begin{array}{ll}\textsc{scomp}: & \text{has}\end{array}\right]\end{array}\right] \rightarrow \left[\begin{array}{ll}\textsc{subcat}: & \mathit{wq}\end{array}\right]$$

---

27 • `vp` → `v`

$$\left[\begin{array}{ll}\textsc{subcat}: & \mathit{aux} \\ \textsc{pos}: & \mathit{midfield} \\ \textsc{tagr}: & \left[\begin{array}{ll}\textsc{day}: & \text{may} \\ \textsc{tod}: & \text{may} \\ \textsc{time}: & \text{may}\end{array}\right] \\ \textsc{vagr}: & \langle 12 \rangle \\ \textsc{semcat}: & \langle 13 \rangle\end{array}\right] \rightarrow \left[\begin{array}{ll}\textsc{subcat}: & \mathit{inf} \\ \textsc{vagr}: & \langle 12 \rangle \\ \textsc{semcat}: & \langle 13 \rangle\end{array}\right]$$

---

28 • `vp` → `v` `vp`

$$\left[\begin{array}{ll}\textsc{subcat}: & \mathit{aux} \\ \textsc{pos}: & \mathit{midfield} \\ \textsc{rreq}: & \langle 7 \rangle \\ \textsc{ragr}: & \langle 8 \rangle \\ \textsc{tagr}: & \langle 9 \rangle \\ \textsc{vagr}: & \langle 10 \rangle \\ \textsc{semcat}: & \langle 11 \rangle\end{array}\right] \rightarrow \left[\begin{array}{ll}\textsc{subcat}: & \mathit{inf} \\ \textsc{vagr}: & \langle 10 \rangle \\ \textsc{semcat}: & \langle 11 \rangle\end{array}\right] \left[\begin{array}{ll}\textsc{subcat}: & \mathit{aux} \\ \textsc{pos}: & \mathit{postfield} \\ \textsc{rreq}: & \langle 7 \rangle \\ \textsc{ragr}: & \langle 8 \rangle \\ \textsc{tagr}: & \langle 9 \rangle \\ \textsc{vagr}: & \langle 10 \rangle \\ \textsc{semcat}: & \langle 11 \rangle\end{array}\right]$$

---

29 • `vp` → `v` `vp`

$$\left[\begin{array}{ll}\textsc{subcat}: & \mathit{sc} \\ \textsc{person}: & \langle 2 \rangle \\ \textsc{agr}: & \langle 3 \rangle \\ \textsc{pos}: & \mathit{midfield} \\ \textsc{rreq}: & \langle 7 \rangle \\ \textsc{ragr}: & \langle 8 \rangle \\ \textsc{tagr}: & \langle 9 \rangle \\ \textsc{vagr}: & \langle 10 \rangle \\ \textsc{semcat}: & \langle 11 \rangle\end{array}\right] \rightarrow \left[\begin{array}{ll}\textsc{subcat}: & \mathit{fin} \\ \textsc{person}: & \langle 2 \rangle \\ \textsc{agr}: & \langle 3 \rangle \\ \textsc{vagr}: & \langle 10 \rangle \\ \textsc{semcat}: & \langle 11 \rangle\end{array}\right] \left[\begin{array}{ll}\textsc{subcat}: & \mathit{sc} \\ \textsc{person}: & \langle 2 \rangle \\ \textsc{agr}: & \langle 3 \rangle \\ \textsc{pos}: & \mathit{postfield} \\ \textsc{rreq}: & \langle 7 \rangle \\ \textsc{ragr}: & \langle 8 \rangle \\ \textsc{tagr}: & \langle 9 \rangle \\ \textsc{vagr}: & \langle 10 \rangle \\ \textsc{semcat}: & \langle 11 \rangle\end{array}\right]$$

30 • `vp → v`

$$\begin{bmatrix} \textsc{subcat}: \textit{sc} \\ \textsc{person}: \langle 2 \rangle \\ \textsc{agr}: \langle 3 \rangle \\ \textsc{pos}: \textit{midfield} \\ \textsc{tagr}: \begin{bmatrix} \textsc{day}: \text{may} \\ \textsc{tod}: \text{may} \\ \textsc{time}: \text{may} \end{bmatrix} \\ \textsc{vagr}: \langle 12 \rangle \\ \textsc{semcat}: \langle 13 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \textit{fin} \\ \textsc{person}: \langle 2 \rangle \\ \textsc{agr}: \langle 3 \rangle \\ \textsc{vagr}: \langle 12 \rangle \\ \textsc{semcat}: \langle 13 \rangle \end{bmatrix}$$

31 • `vp → v vp`

$$\begin{bmatrix} \textsc{subcat}: \textit{sc} \\ \textsc{person}: \langle 2 \rangle \\ \textsc{agr}: \langle 3 \rangle \\ \textsc{pos}: \textit{midfield} \\ \textsc{rreq}: \langle 7 \rangle \\ \textsc{ragr}: \langle 8 \rangle \\ \textsc{tagr}: \langle 9 \rangle \\ \textsc{vagr}: \langle 10 \rangle \\ \textsc{semcat}: \langle 11 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \textit{infaux} \\ \textsc{person}: \langle 2 \rangle \\ \textsc{agr}: \langle 3 \rangle \\ \textsc{vagr}: \langle 10 \rangle \\ \textsc{semcat}: \langle 11 \rangle \end{bmatrix} \begin{bmatrix} \textsc{subcat}: \textit{sc} \\ \textsc{person}: \langle 2 \rangle \\ \textsc{agr}: \langle 3 \rangle \\ \textsc{pos}: \textit{postfield} \\ \textsc{rreq}: \langle 7 \rangle \\ \textsc{ragr}: \langle 8 \rangle \\ \textsc{tagr}: \langle 9 \rangle \\ \textsc{vagr}: \langle 10 \rangle \\ \textsc{semcat}: \langle 11 \rangle \end{bmatrix}$$

32 • `vp → v`

$$\begin{bmatrix} \textsc{subcat}: \textit{sc} \\ \textsc{person}: \langle 2 \rangle \\ \textsc{agr}: \langle 3 \rangle \\ \textsc{pos}: \textit{midfield} \\ \textsc{tagr}: \begin{bmatrix} \textsc{day}: \text{may} \\ \textsc{tod}: \text{may} \\ \textsc{time}: \text{may} \end{bmatrix} \\ \textsc{vagr}: \langle 12 \rangle \\ \textsc{semcat}: \langle 13 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \textit{infaux} \\ \textsc{person}: \langle 2 \rangle \\ \textsc{agr}: \langle 3 \rangle \\ \textsc{vagr}: \langle 12 \rangle \\ \textsc{semcat}: \langle 13 \rangle \end{bmatrix}$$

33 • `vp → v`

$$\begin{bmatrix} \textsc{subcat}: \textit{zu} \\ \textsc{pos}: \textit{midfield} \\ \textsc{tagr}: \begin{bmatrix} \textsc{day}: \text{may} \\ \textsc{tod}: \text{may} \\ \textsc{time}: \text{may} \end{bmatrix} \\ \textsc{vagr}: \langle 12 \rangle \\ \textsc{semcat}: \langle 13 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \textit{zu} \\ \textsc{vagr}: \langle 12 \rangle \\ \textsc{semcat}: \langle 13 \rangle \end{bmatrix}$$

34 • vp → sc

$$\begin{bmatrix} \textsc{pos}: \textit{postfield} \\ \textsc{tagr}: \begin{bmatrix} \textsc{day}: \text{may} \\ \textsc{tod}: \text{may} \\ \textsc{time}: \text{may} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \textit{main} \end{bmatrix}$$

---

35 • vp → sc

$$\begin{bmatrix} \textsc{pos}: \textit{postfield} \\ \textsc{tagr}: \begin{bmatrix} \textsc{day}: \text{may} \\ \textsc{tod}: \text{may} \\ \textsc{time}: \text{may} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \textit{sub} \end{bmatrix}$$

---

36 • vp → sc

$$\begin{bmatrix} \textsc{pos}: \textit{postfield} \\ \textsc{tagr}: \begin{bmatrix} \textsc{day}: \text{may} \\ \textsc{tod}: \text{may} \\ \textsc{time}: \text{may} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \textit{zu} \end{bmatrix}$$

---

37 • vp → sc

$$\begin{bmatrix} \textsc{pos}: \textit{postfield} \\ \textsc{rreq}: \langle 8 \rangle \\ \textsc{ragr}: \langle 9 \rangle \\ \textsc{tagr}: \begin{bmatrix} \textsc{day}: \text{may} \\ \textsc{tod}: \text{may} \\ \textsc{time}: \text{may} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \textit{rel} \\ \textsc{rreq}: \langle 8 \rangle \\ \textsc{ragr}: \langle 9 \rangle \end{bmatrix}$$

---

38 • vp → sc

$$\begin{bmatrix} \textsc{pos}: \textit{postfield} \\ \textsc{tagr}: \begin{bmatrix} \textsc{day}: \text{may} \\ \textsc{tod}: \text{may} \\ \textsc{time}: \text{may} \end{bmatrix} \\ \textsc{vagr}: \begin{bmatrix} \textsc{scomp}: \text{has} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \textit{scomp} \end{bmatrix}$$

---

39 • vp → sc

$$\begin{bmatrix} \textsc{pos}: \textit{postfield} \\ \textsc{tagr}: \begin{bmatrix} \textsc{day}: \text{may} \\ \textsc{tod}: \text{may} \\ \textsc{time}: \text{may} \end{bmatrix} \\ \textsc{vagr}: \begin{bmatrix} \textsc{scomp}: \text{has} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \textit{wq} \end{bmatrix}$$

---

40 • `vp` → `comp`

$$\left[\begin{array}{l} \textsc{person}: \langle 3\rangle \\ \textsc{agr}: \langle 4\rangle \\ \textsc{pos}: \mathit{postfield} \\ \textsc{tagr}: \left[\begin{array}{l} \textsc{day}: \mathrm{may} \\ \textsc{tod}: \mathrm{may} \\ \textsc{time}: \mathrm{may} \end{array}\right] \\ \textsc{vagr}: \langle 13\rangle \end{array}\right] \rightarrow \left[\begin{array}{l} \textsc{subcat}: \mathit{ext} \\ \textsc{person}: \langle 3\rangle \\ \textsc{agr}: \langle 4\rangle \\ \textsc{vagr}: \langle 13\rangle \end{array}\right]$$

---

41 • `v` → `v v`

$$\left[\begin{array}{l} \textsc{subcat}: \mathit{infaux} \\ \textsc{person}: \langle 2\rangle \\ \textsc{agr}: \langle 3\rangle \\ \textsc{vagr}: \langle 11\rangle \\ \textsc{semcat}: \langle 12\rangle \end{array}\right] \rightarrow \left[\begin{array}{l} \textsc{subcat}: \mathit{inf} \\ \textsc{vagr}: \langle 11\rangle \\ \textsc{semcat}: \langle 12\rangle \end{array}\right] \left[\begin{array}{l} \textsc{subcat}: \mathit{aux} \\ \textsc{person}: \langle 2\rangle \\ \textsc{agr}: \langle 3\rangle \end{array}\right]$$

---

42 • `v` → `part v`

$$\left[\begin{array}{l} \textsc{subcat}: \mathit{zu} \\ \textsc{vagr}: \langle 11\rangle \\ \textsc{semcat}: \langle 12\rangle \end{array}\right] \rightarrow \left[\ \textsc{subcat}: \mathit{zu}\ \right] \left[\begin{array}{l} \textsc{subcat}: \mathit{inf} \\ \textsc{vagr}: \langle 11\rangle \\ \textsc{semcat}: \langle 12\rangle \end{array}\right]$$

---

43 • `v` → `v v`

$$\left[\begin{array}{l} \textsc{subcat}: \mathit{zu} \\ \textsc{vagr}: \langle 11\rangle \\ \textsc{semcat}: \langle 12\rangle \end{array}\right] \rightarrow \left[\begin{array}{l} \textsc{subcat}: \mathit{inf} \\ \textsc{vagr}: \langle 11\rangle \\ \textsc{semcat}: \langle 12\rangle \end{array}\right] \left[\ \textsc{subcat}: \mathit{zuaux}\ \right]$$

---

44 • `v` → `part v`

$$\left[\ \textsc{subcat}: \mathit{zuaux}\ \right] \rightarrow \left[\ \textsc{subcat}: \mathit{zu}\ \right] \left[\ \textsc{subcat}: \mathit{auxinf}\ \right]$$

---

45 • `sc` → `part vp`

$$\left[\ \textsc{subcat}: \mathit{main}\ \right] \rightarrow \left[\ \textsc{subcat}: \mathit{main}\ \right] \left[\begin{array}{l} \textsc{pos}: \mathit{prefield} \\ \textsc{rreq}: \mathit{complete} \\ \textsc{vagr}: \mathit{complete} \end{array}\right]$$

---

46 • `sc` → `part vp`

$$\left[\ \textsc{subcat}: \mathit{sub}\ \right] \rightarrow \left[\ \textsc{subcat}: \mathit{sub}\ \right] \left[\begin{array}{l} \textsc{subcat}: \mathit{sc} \\ \textsc{pos}: \mathit{midfield} \\ \textsc{rreq}: \mathit{complete} \\ \textsc{vagr}: \mathit{complete} \end{array}\right]$$

---

47 • sc → vp

$$\left[\ \textsc{subcat}:\ zu\ \right] \rightarrow \left[\begin{array}{l}\textsc{subcat}:\ zu\\ \textsc{pos}:\ midfield\\ \textsc{rreq}:\ complete\end{array}\right]$$

---

48 • sc → part vp

$$\left[\ \textsc{subcat}:\ zu\ \right] \rightarrow \left[\ \textsc{subcat}:\ um\ \right] \left[\begin{array}{l}\textsc{subcat}:\ zu\\ \textsc{pos}:\ midfield\\ \textsc{rreq}:\ complete\end{array}\right]$$

---

49 • sc → vp

$$\left[\begin{array}{l}\textsc{subcat}:\ rel\\ \textsc{rreq}:\ \left[\ \textsc{req}:\ \text{has}\ \right]\\ \textsc{ragr}:\ \langle 9\rangle\end{array}\right] \rightarrow \left[\begin{array}{l}\textsc{subcat}:\ rel\\ \textsc{pos}:\ begin\\ \textsc{ragr}:\ \langle 9\rangle\\ \textsc{vagr}:\ complete\end{array}\right]$$

---

50 • sc → part vp

$$\left[\ \textsc{subcat}:\ scomp\ \right] \rightarrow \left[\ \textsc{subcat}:\ scomp\ \right] \left[\begin{array}{l}\textsc{subcat}:\ sc\\ \textsc{pos}:\ midfield\\ \textsc{rreq}:\ complete\\ \textsc{vagr}:\ complete\end{array}\right]$$

---

51 • sc → vp

$$\left[\begin{array}{l}\textsc{subcat}:\ wq\\ \textsc{pos}:\ begin\end{array}\right] \rightarrow \left[\begin{array}{l}\textsc{subcat}:\ wq\\ \textsc{rreq}:\ complete\\ \textsc{vagr}:\ complete\end{array}\right]$$

---

52 • sc → vp

$$\left[\begin{array}{l}\textsc{subcat}:\ cond\\ \textsc{pos}:\ begin\end{array}\right] \rightarrow \left[\begin{array}{l}\textsc{pos}:\ verb\\ \textsc{rreq}:\ complete\\ \textsc{vagr}:\ complete\end{array}\right]$$

---

53 • vp → rc vp

$$\left[\begin{array}{l}\textsc{subcat}:\ \langle 1\rangle\\ \textsc{pos}:\ begin\\ \textsc{ragr}:\ \langle 9\rangle\\ \textsc{vagr}:\ \langle 11\rangle\\ \textsc{semcat}:\ \langle 12\rangle\end{array}\right] \rightarrow \left[\begin{array}{l}\textsc{subcat}:\ \langle 1\rangle\\ \textsc{person}:\ \langle 14\rangle\\ \textsc{agr}:\ \langle 15\rangle\\ \textsc{ragr}:\ \langle 9\rangle\\ \textsc{vagr}:\ \langle 11\rangle\end{array}\right] \left[\begin{array}{l}\textsc{subcat}:\ sc\\ \textsc{person}:\ \langle 14\rangle\\ \textsc{agr}:\ \langle 15\rangle\\ \textsc{pos}:\ midfield\\ \textsc{vagr}:\ \langle 11\rangle\\ \textsc{semcat}:\ \langle 12\rangle\end{array}\right]$$

54 • `rc` → `pro`

$$\begin{bmatrix} \textsc{subcat}: \textit{rel} \\ \textsc{person}: \textit{third} \\ \textsc{agr}: \langle 2 \rangle \\ \textsc{ragr}: \langle 2 \rangle \\ \textsc{vagr}: \begin{bmatrix} \textsc{subj}: \text{has} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \textit{rel} \\ \textsc{case}: \textit{nom} \\ \textsc{agr}: \langle 2 \rangle \end{bmatrix}$$

55 • `rc` → `pro`

$$\begin{bmatrix} \textsc{subcat}: \textit{rel} \\ \textsc{ragr}: \langle 9 \rangle \\ \textsc{vagr}: \begin{bmatrix} \textsc{obj}1: \text{has} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \textit{rel} \\ \textsc{case}: \textit{dat} \\ \textsc{agr}: \langle 9 \rangle \end{bmatrix}$$

56 • `rc` → `pro`

$$\begin{bmatrix} \textsc{subcat}: \textit{rel} \\ \textsc{ragr}: \langle 9 \rangle \\ \textsc{vagr}: \begin{bmatrix} \textsc{obj}2: \text{has} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \textit{rel} \\ \textsc{case}: \textit{acc} \\ \textsc{agr}: \langle 9 \rangle \end{bmatrix}$$

57 • `rc` → `pro`

$$\begin{bmatrix} \textsc{subcat}: \textit{rel} \\ \textsc{ragr}: \langle 9 \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \textit{rel} \\ \textsc{case}: \textit{none} \\ \textsc{agr}: \langle 9 \rangle \end{bmatrix}$$

58 • `pro` → `p pro`

$$\begin{bmatrix} \textsc{subcat}: \textit{rel} \\ \textsc{case}: \textit{none} \\ \textsc{agr}: \langle 2 \rangle \\ \textsc{pos}: \textit{begin} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{case}: \langle 12 \rangle \\ \textsc{det}: \textit{no} \end{bmatrix} \begin{bmatrix} \textsc{subcat}: \textit{rel} \\ \textsc{case}: \langle 12 \rangle \\ \textsc{agr}: \langle 2 \rangle \end{bmatrix}$$

59 • `rc` → `pro`

$$\begin{bmatrix} \textsc{subcat}: \textit{wq} \\ \textsc{person}: \textit{third} \\ \textsc{agr}: \langle 2 \rangle \\ \textsc{vagr}: \begin{bmatrix} \textsc{subj}: \text{has} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \textit{qu}1 \\ \textsc{case}: \textit{nom} \\ \textsc{agr}: \langle 2 \rangle \end{bmatrix}$$

60 • `rc` → `pro`

$$\begin{bmatrix} \textsc{subcat}: \textit{wq} \\ \textsc{vagr}: \begin{bmatrix} \textsc{obj}1: \text{has} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \textit{qu}1 \\ \textsc{case}: \textit{dat} \end{bmatrix}$$

61 • `rc → pro`

$$\begin{bmatrix} \textsc{subcat}: & wq \\ \textsc{vagr}: & \begin{bmatrix} \textsc{obj2}: & \text{has} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: & qu1 \\ \textsc{case}: & acc \end{bmatrix}$$

---

62 • `rc → pro`

$$\begin{bmatrix} \textsc{subcat}: & wq \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: & qu1 \\ \textsc{case}: & none \end{bmatrix}$$

---

63 • `pro → p pro`

$$\begin{bmatrix} \textsc{subcat}: & qu1 \\ \textsc{case}: & none \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{case}: & \langle 13 \rangle \\ \textsc{det}: & no \end{bmatrix} \begin{bmatrix} \textsc{subcat}: & qu1 \\ \textsc{case}: & \langle 13 \rangle \end{bmatrix}$$

---

64 • `comp → pp`

$$\begin{bmatrix} \textsc{subcat}: & mid \\ \textsc{rreq}: & \langle 8 \rangle \\ \textsc{ragr}: & \langle 9 \rangle \\ \textsc{vagr}: & \begin{bmatrix} \textsc{ppobj}: & \text{has}\langle 16 \rangle \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{rreq}: & \langle 8 \rangle \\ \textsc{ragr}: & \langle 9 \rangle \\ \textsc{semcat}: & \langle 16 \rangle \end{bmatrix}$$

---

65 • `comp → np`

$$\begin{bmatrix} \textsc{subcat}: & mid \\ \textsc{person}: & \langle 2 \rangle \\ \textsc{agr}: & \langle 3 \rangle \\ \textsc{rreq}: & \langle 8 \rangle \\ \textsc{ragr}: & \langle 9 \rangle \\ \textsc{vagr}: & \begin{bmatrix} \textsc{subj}: & \text{has}\langle 11 \rangle \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: & nom \\ \textsc{person}: & \langle 2 \rangle \\ \textsc{agr}: & \langle 3 \rangle \\ \textsc{rreq}: & \langle 8 \rangle \\ \textsc{ragr}: & \langle 9 \rangle \\ \textsc{semcat}: & \langle 11 \rangle \end{bmatrix}$$

---

66 • `comp → np`

$$\begin{bmatrix} \textsc{subcat}: & mid \\ \textsc{rreq}: & \langle 8 \rangle \\ \textsc{ragr}: & \langle 9 \rangle \\ \textsc{vagr}: & \begin{bmatrix} \textsc{obj1}: & \text{has}\langle 12 \rangle \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: & dat \\ \textsc{rreq}: & \langle 8 \rangle \\ \textsc{ragr}: & \langle 9 \rangle \\ \textsc{semcat}: & \langle 12 \rangle \end{bmatrix}$$

---

67 • `comp → np`

$$\begin{bmatrix} \textsc{subcat}: & mid \\ \textsc{rreq}: & \langle 8 \rangle \\ \textsc{ragr}: & \langle 9 \rangle \\ \textsc{vagr}: & \begin{bmatrix} \textsc{obj2}: & \text{has}\langle 13 \rangle \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: & acc \\ \textsc{rreq}: & \langle 8 \rangle \\ \textsc{ragr}: & \langle 9 \rangle \\ \textsc{semcat}: & \langle 13 \rangle \end{bmatrix}$$

---

68 • `comp` → `sc`

$$\left[\begin{array}{l} \textsc{subcat}: \textit{ext} \\ \textsc{rreq}: \langle 8\rangle \\ \textsc{ragr}: \langle 9\rangle \\ \textsc{vagr}: \left[\ \textsc{scomp}: \text{has}\langle 15\rangle\ \right] \end{array}\right] \rightarrow \left[\begin{array}{l} \textsc{subcat}: \textit{scomp} \\ \textsc{rreq}: \langle 8\rangle \\ \textsc{ragr}: \langle 9\rangle \\ \textsc{semcat}: \langle 15\rangle \end{array}\right]$$

---

69 • `comp` → `vcomp`

$$\left[\begin{array}{l} \textsc{subcat}: \textit{mid} \\ \textsc{vagr}: \left[\ \textsc{vcomp}: \text{has}\langle 14\rangle\ \right] \end{array}\right] \rightarrow \left[\ \textsc{subcat}: \langle 14\rangle\ \right]$$

---

70 • `pp` → `adv pp`

$$\left[\begin{array}{l} \textsc{rreq}: \langle 9\rangle \\ \textsc{ragr}: \langle 10\rangle \\ \textsc{semcat}: \langle 13\rangle \end{array}\right] \rightarrow \left[\ \textsc{semcat}: \textit{auch}\ \right] \left[\begin{array}{l} \textsc{rreq}: \langle 9\rangle \\ \textsc{ragr}: \langle 10\rangle \\ \textsc{semcat}: \langle 13\rangle \end{array}\right]$$

---

71 • `pp` → `p pn`

$$\left[\ \textsc{semcat}: (\langle 13\rangle, \langle 14\rangle)\ \right] \rightarrow \left[\ \textsc{semcat}: \langle 13\rangle\ \right] \left[\ \textsc{semcat}: \langle 14\rangle\ \right]$$

---

72 • `pp` → `p np`

$$\left[\begin{array}{l} \textsc{rreq}: \langle 9\rangle \\ \textsc{ragr}: \langle 10\rangle \\ \textsc{semcat}: (\langle 13\rangle, \langle 14\rangle) \end{array}\right] \rightarrow \left[\begin{array}{l} \textsc{case}: \langle 16\rangle \\ \textsc{semcat}: \langle 13\rangle \end{array}\right] \left[\begin{array}{l} \textsc{subcat}: \langle 16\rangle \\ \textsc{rreq}: \langle 9\rangle \\ \textsc{ragr}: \langle 10\rangle \\ \textsc{semcat}: \langle 14\rangle \end{array}\right]$$

---

73 • `pp` → `p v`

$$\left[\ \textsc{semcat}: (\langle 13\rangle, \langle 14\rangle)\ \right] \rightarrow \left[\ \textsc{semcat}: \langle 13\rangle\ \right] \left[\begin{array}{l} \textsc{subcat}: \textit{inf} \\ \textsc{semcat}: \langle 14\rangle \end{array}\right]$$

---

74 • `np` → `adv np`

$$\left[\begin{array}{l} \textsc{subcat}: \langle 1\rangle \\ \textsc{person}: \langle 3\rangle \\ \textsc{agr}: \langle 4\rangle \\ \textsc{rreq}: \langle 9\rangle \\ \textsc{ragr}: \langle 10\rangle \\ \textsc{tagr}: \langle 11\rangle \\ \textsc{semcat}: \langle 13\rangle \end{array}\right] \rightarrow \left[\ \textsc{semcat}: \textit{auch}\ \right] \left[\begin{array}{l} \textsc{subcat}: \langle 1\rangle \\ \textsc{person}: \langle 3\rangle \\ \textsc{agr}: \langle 4\rangle \\ \textsc{rreq}: \langle 9\rangle \\ \textsc{ragr}: \langle 10\rangle \\ \textsc{tagr}: \langle 11\rangle \\ \textsc{semcat}: \langle 13\rangle \end{array}\right]$$

---

75 • `np → np sc`

$$\begin{bmatrix} \textsc{subcat}: \langle 1\rangle \\ \textsc{person}: \langle 3\rangle \\ \textsc{agr}: \langle 4\rangle \\ \textsc{tagr}: \langle 11\rangle \\ \textsc{semcat}: \langle 13\rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \langle 1\rangle \\ \textsc{person}: \langle 3\rangle \\ \textsc{agr}: \langle 4\rangle \\ \textsc{tagr}: \langle 11\rangle \\ \textsc{semcat}: \langle 13\rangle \end{bmatrix} \begin{bmatrix} \textsc{subcat}: \mathit{rel} \\ \textsc{ragr}: \langle 4\rangle \end{bmatrix}$$

---

76 • `np → pro`

$$\begin{bmatrix} \textsc{subcat}: \langle 1\rangle \\ \textsc{person}: \langle 3\rangle \\ \textsc{agr}: \langle 4\rangle \\ \textsc{tagr}: \begin{bmatrix} \textsc{day}: \text{may} \\ \textsc{tod}: \text{may} \\ \textsc{time}: \text{may} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{subcat}: \mathit{person} \\ \textsc{case}: \langle 1\rangle \\ \textsc{person}: \langle 3\rangle \\ \textsc{agr}: \langle 4\rangle \end{bmatrix}$$

---

77 • `np → n`

$$\begin{bmatrix} \textsc{subcat}: \langle 1\rangle \\ \textsc{person}: \mathit{third} \\ \textsc{agr}: \begin{bmatrix} \textsc{number}: \mathit{sing} \\ \textsc{gender}: \langle 3\rangle \end{bmatrix} \\ \textsc{tagr}: \begin{bmatrix} \textsc{day}: \text{may} \\ \textsc{tod}: \text{may} \\ \textsc{time}: \text{may} \end{bmatrix} \\ \textsc{semcat}: \langle 14\rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{case}: \langle 1\rangle \\ \textsc{agr}: \begin{bmatrix} \textsc{number}: \mathit{sing} \\ \textsc{gender}: \langle 3\rangle \end{bmatrix} \\ \textsc{det}: \mathit{no} \\ \textsc{semcat}: \langle 14\rangle \end{bmatrix}$$

---

78 • `np → n`

$$\begin{bmatrix} \textsc{subcat}: \langle 1\rangle \\ \textsc{person}: \mathit{third} \\ \textsc{agr}: \begin{bmatrix} \textsc{number}: \mathit{plur} \\ \textsc{gender}: \langle 3\rangle \end{bmatrix} \\ \textsc{tagr}: \begin{bmatrix} \textsc{day}: \text{may} \\ \textsc{tod}: \text{may} \\ \textsc{time}: \text{may} \end{bmatrix} \\ \textsc{semcat}: \langle 14\rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{case}: \langle 1\rangle \\ \textsc{agr}: \begin{bmatrix} \textsc{number}: \mathit{plur} \\ \textsc{gender}: \langle 3\rangle \end{bmatrix} \\ \textsc{semcat}: \langle 14\rangle \end{bmatrix}$$

---

79 • `np → det n`

$$\begin{bmatrix} \textsc{subcat}: \langle 1\rangle \\ \textsc{person}: \mathit{third} \\ \textsc{agr}: \langle 3\rangle \\ \textsc{tagr}: \begin{bmatrix} \textsc{day}: \text{may} \\ \textsc{tod}: \text{may} \\ \textsc{time}: \text{may} \end{bmatrix} \\ \textsc{semcat}: \langle 14\rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{case}: \langle 1\rangle \\ \textsc{agr}: \langle 3\rangle \\ \textsc{def}: \langle 20\rangle \end{bmatrix} \begin{bmatrix} \textsc{case}: \langle 1\rangle \\ \textsc{agr}: \langle 3\rangle \\ \textsc{def}: \langle 20\rangle \\ \textsc{semcat}: \langle 14\rangle \end{bmatrix}$$

$$80 \bullet \texttt{np} \rightarrow \texttt{n}$$

$$\left[\begin{array}{ll}
\textsc{subcat}: & \langle 1 \rangle \\
\textsc{person}: & \mathit{third} \\
\textsc{agr}: & \left[\begin{array}{ll} \textsc{number}: & \mathit{sing} \\ \textsc{gender}: & \langle 3 \rangle \end{array}\right] \\
\textsc{rreq}: & \left[\begin{array}{ll} \textsc{req}: & \text{need} \end{array}\right] \\
\textsc{ragr}: & \left[\begin{array}{ll} \textsc{number}: & \mathit{sing} \\ \textsc{gender}: & \langle 3 \rangle \end{array}\right] \\
\textsc{tagr}: & \left[\begin{array}{ll} \textsc{day}: & \text{may} \\ \textsc{tod}: & \text{may} \\ \textsc{time}: & \text{may} \end{array}\right] \\
\textsc{semcat}: & \langle 13 \rangle
\end{array}\right]
\rightarrow
\left[\begin{array}{ll}
\textsc{case}: & \langle 1 \rangle \\
\textsc{agr}: & \left[\begin{array}{ll} \textsc{number}: & \mathit{sing} \\ \textsc{gender}: & \langle 3 \rangle \end{array}\right] \\
\textsc{det}: & \mathit{no} \\
\textsc{semcat}: & \langle 13 \rangle
\end{array}\right]$$

$$81 \bullet \texttt{np} \rightarrow \texttt{n}$$

$$\left[\begin{array}{ll}
\textsc{subcat}: & \langle 1 \rangle \\
\textsc{person}: & \mathit{third} \\
\textsc{agr}: & \left[\begin{array}{ll} \textsc{number}: & \mathit{plur} \\ \textsc{gender}: & \langle 3 \rangle \end{array}\right] \\
\textsc{rreq}: & \left[\begin{array}{ll} \textsc{req}: & \text{need} \end{array}\right] \\
\textsc{ragr}: & \left[\begin{array}{ll} \textsc{number}: & \mathit{plur} \\ \textsc{gender}: & \langle 3 \rangle \end{array}\right] \\
\textsc{tagr}: & \left[\begin{array}{ll} \textsc{day}: & \text{may} \\ \textsc{tod}: & \text{may} \\ \textsc{time}: & \text{may} \end{array}\right] \\
\textsc{semcat}: & \langle 13 \rangle
\end{array}\right]
\rightarrow
\left[\begin{array}{ll}
\textsc{case}: & \langle 1 \rangle \\
\textsc{agr}: & \left[\begin{array}{ll} \textsc{number}: & \mathit{plur} \\ \textsc{gender}: & \langle 3 \rangle \end{array}\right] \\
\textsc{semcat}: & \langle 13 \rangle
\end{array}\right]$$

$$82 \bullet \texttt{np} \rightarrow \texttt{det}\ \texttt{n}$$

$$\left[\begin{array}{ll}
\textsc{subcat}: & \langle 1 \rangle \\
\textsc{person}: & \mathit{third} \\
\textsc{agr}: & \langle 3 \rangle \\
\textsc{rreq}: & \left[\begin{array}{ll} \textsc{req}: & \text{need} \end{array}\right] \\
\textsc{ragr}: & \langle 3 \rangle \\
\textsc{tagr}: & \left[\begin{array}{ll} \textsc{day}: & \text{may} \\ \textsc{tod}: & \text{may} \\ \textsc{time}: & \text{may} \end{array}\right] \\
\textsc{semcat}: & \langle 13 \rangle
\end{array}\right]
\rightarrow
\left[\begin{array}{ll}
\textsc{case}: & \langle 1 \rangle \\
\textsc{agr}: & \langle 3 \rangle \\
\textsc{def}: & \langle 19 \rangle
\end{array}\right]
\left[\begin{array}{ll}
\textsc{case}: & \langle 1 \rangle \\
\textsc{agr}: & \langle 3 \rangle \\
\textsc{def}: & \langle 19 \rangle \\
\textsc{semcat}: & \langle 13 \rangle
\end{array}\right]$$

$$83 \bullet \texttt{n} \rightarrow \texttt{adj}\ \texttt{n}$$

$$\left[\begin{array}{ll}
\textsc{case}: & \langle 2 \rangle \\
\textsc{agr}: & \langle 4 \rangle \\
\textsc{det}: & \langle 7 \rangle \\
\textsc{def}: & \langle 8 \rangle \\
\textsc{semcat}: & \langle 13 \rangle
\end{array}\right]
\rightarrow
\left[\begin{array}{ll}
\textsc{case}: & \langle 2 \rangle \\
\textsc{agr}: & \langle 4 \rangle \\
\textsc{def}: & \langle 8 \rangle
\end{array}\right]
\left[\begin{array}{ll}
\textsc{case}: & \langle 2 \rangle \\
\textsc{agr}: & \langle 4 \rangle \\
\textsc{det}: & \langle 7 \rangle \\
\textsc{def}: & \langle 8 \rangle \\
\textsc{semcat}: & \langle 13 \rangle
\end{array}\right]$$

---

84 • `np → np pp`

$$\begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \mathit{third} \\ \text{AGR}: \langle 3\rangle \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{SEMCAT}: \mathit{zug} \end{bmatrix} \rightarrow \begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \mathit{third} \\ \text{AGR}: \langle 3\rangle \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{SEMCAT}: \mathit{zug} \end{bmatrix} \begin{bmatrix} \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{SEMCAT}: (\mathit{von}, \mathit{ort}) \end{bmatrix}$$

---

85 • `np → np pp`

$$\begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \mathit{third} \\ \text{AGR}: \langle 3\rangle \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{SEMCAT}: \mathit{zug} \end{bmatrix} \rightarrow \begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \mathit{third} \\ \text{AGR}: \langle 3\rangle \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{SEMCAT}: \mathit{zug} \end{bmatrix} \begin{bmatrix} \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{SEMCAT}: (\mathit{nach}, \mathit{ort}) \end{bmatrix}$$

---

86 • `np → np tc`

$$\begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \mathit{third} \\ \text{AGR}: \langle 3\rangle \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{SEMCAT}: \langle 12\rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{SUBCAT}: \langle 1\rangle \\ \text{PERSON}: \mathit{third} \\ \text{AGR}: \langle 3\rangle \\ \text{RREQ}: \langle 8\rangle \\ \text{RAGR}: \langle 9\rangle \\ \text{TAGR}: \langle 10\rangle \\ \text{SEMCAT}: \langle 12\rangle \end{bmatrix} \begin{bmatrix} \text{TAGR}: \langle 10\rangle \end{bmatrix}$$

---

87 • `pro → pro n`

$$\begin{bmatrix} \text{SUBCAT}: \mathit{qu1} \\ \text{CASE}: \langle 1\rangle \\ \text{AGR}: \langle 3\rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{SUBCAT}: \mathit{qu2} \\ \text{CASE}: \langle 1\rangle \\ \text{AGR}: \langle 3\rangle \end{bmatrix} \begin{bmatrix} \text{CASE}: \langle 1\rangle \\ \text{AGR}: \langle 3\rangle \\ \text{DEF}: \mathit{indef} \end{bmatrix}$$

---

88 • `junk → adj n`

$$\begin{bmatrix} \text{SUBCAT}: \mathit{init} \end{bmatrix} \rightarrow \begin{bmatrix} \text{CASE}: \mathit{acc} \\ \text{AGR}: \langle 15\rangle \\ \text{SEMCAT}: \mathit{qual} \end{bmatrix} \begin{bmatrix} \text{CASE}: \mathit{acc} \\ \text{AGR}: \langle 15\rangle \\ \text{SEMCAT}: \mathit{tageszeit} \end{bmatrix}$$

---

89 • `junk → v n`

$$\begin{bmatrix} \text{SUBCAT}: \mathit{init} \end{bmatrix} \rightarrow \begin{bmatrix} \text{PERSON}: \mathit{imp} \\ \text{SEMCAT}: \mathit{gruessen} \end{bmatrix} \begin{bmatrix} \text{CASE}: \mathit{acc} \\ \text{AGR}: \begin{bmatrix} \text{NUMBER}: \mathit{sing} \end{bmatrix} \\ \text{SEMCAT}: \mathit{gott} \end{bmatrix}$$

---

90 • `tc` → `tp`

$$\begin{bmatrix} \textsc{subcat}: \textit{single} \\ \textsc{tagr}: \begin{bmatrix} \textsc{time}: \text{has}\langle 12\rangle \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{semcat}: (\langle 12\rangle, \textit{zeit}) \end{bmatrix}$$

---

91 • `tc` → `tc tp`

$$\begin{bmatrix} \textsc{subcat}: \textit{multiple} \\ \textsc{tagr}: \begin{bmatrix} \textsc{time}: \text{has}\langle 12\rangle \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{tagr}: \begin{bmatrix} \textsc{time}: \text{has}\langle 12\rangle \end{bmatrix} \end{bmatrix} \begin{bmatrix} \textsc{semcat}: (\langle 12\rangle, \textit{zeit}) \end{bmatrix}$$

---

92 • `tc` → `tp`

$$\begin{bmatrix} \textsc{subcat}: \textit{single} \\ \textsc{tagr}: \begin{bmatrix} \textsc{tod}: \text{has}\langle 11\rangle \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{semcat}: (\langle 11\rangle, \textit{tageszeit}) \end{bmatrix}$$

---

93 • `tc` → `tc tp`

$$\begin{bmatrix} \textsc{subcat}: \textit{multiple} \\ \textsc{tagr}: \begin{bmatrix} \textsc{tod}: \text{has}\langle 11\rangle \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{tagr}: \begin{bmatrix} \textsc{tod}: \text{has}\langle 11\rangle \end{bmatrix} \end{bmatrix} \begin{bmatrix} \textsc{semcat}: (\langle 11\rangle, \textit{tageszeit}) \end{bmatrix}$$

---

94 • `tc` → `tp`

$$\begin{bmatrix} \textsc{subcat}: \textit{single} \\ \textsc{tagr}: \begin{bmatrix} \textsc{day}: \text{has}\langle 10\rangle \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{semcat}: (\langle 10\rangle, \textit{tag}) \end{bmatrix}$$

---

95 • `tc` → `tc tp`

$$\begin{bmatrix} \textsc{subcat}: \textit{multiple} \\ \textsc{tagr}: \begin{bmatrix} \textsc{day}: \text{has}\langle 10\rangle \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{tagr}: \begin{bmatrix} \textsc{day}: \text{has}\langle 10\rangle \end{bmatrix} \end{bmatrix} \begin{bmatrix} \textsc{semcat}: (\langle 10\rangle, \textit{tag}) \end{bmatrix}$$

---

96 • `tp` → `p t`

$$\begin{bmatrix} \textsc{semcat}: (\langle 13\rangle, \textit{zeit}) \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{semcat}: \langle 13\rangle \end{bmatrix} \begin{bmatrix} \textsc{semcat}: \textit{zeit} \end{bmatrix}$$

---

97 • `tp` → `t`

$$\begin{bmatrix} \textsc{semcat}: (\textit{none}, \textit{zeit}) \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{semcat}: \textit{zeit} \end{bmatrix}$$

---

98 • `t` → `adj`

$$\begin{bmatrix} \textsc{subcat}: \textit{low} \\ \textsc{semcat}: \textit{zeit} \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{case}: \textit{none} \\ \textsc{semcat}: \textit{zahl} \end{bmatrix}$$

$99 \bullet \texttt{t} \rightarrow \texttt{adj n}$

$$\begin{bmatrix} \text{SUBCAT}: \textit{high} \\ \text{SEMCAT}: \textit{zeit} \end{bmatrix} \rightarrow \begin{bmatrix} \text{CASE}: \textit{nom} \\ \text{AGR}: \begin{bmatrix} \text{GENDER}: \textit{m} \end{bmatrix} \\ \text{SEMCAT}: \textit{zahl} \end{bmatrix} \begin{bmatrix} \text{CASE}: \textit{nom} \\ \text{AGR}: \begin{bmatrix} \text{NUMBER}: \textit{sing} \end{bmatrix} \\ \text{SEMCAT}: \textit{uhr} \end{bmatrix}$$

$100 \bullet \texttt{t} \rightarrow \texttt{t adj}$

$$\begin{bmatrix} \text{SUBCAT}: \textit{high} \\ \text{SEMCAT}: \textit{zeit} \end{bmatrix} \rightarrow \begin{bmatrix} \text{SUBCAT}: \textit{high} \\ \text{SEMCAT}: \textit{zeit} \end{bmatrix} \begin{bmatrix} \text{CASE}: \textit{none} \\ \text{SEMCAT}: \textit{zahl} \end{bmatrix}$$

$101 \bullet \texttt{t} \rightarrow \texttt{t t}$

$$\begin{bmatrix} \text{SUBCAT}: \textit{high} \\ \text{SEMCAT}: \textit{zeit} \end{bmatrix} \rightarrow \begin{bmatrix} \text{SUBCAT}: \textit{high} \\ \text{SEMCAT}: \textit{zeitrel} \end{bmatrix} \begin{bmatrix} \text{SUBCAT}: \textit{high} \\ \text{SEMCAT}: \textit{zeit} \end{bmatrix}$$

$102 \bullet \texttt{t} \rightarrow \texttt{t t}$

$$\begin{bmatrix} \text{SUBCAT}: \textit{high} \\ \text{SEMCAT}: \textit{zeit} \end{bmatrix} \rightarrow \begin{bmatrix} \text{SUBCAT}: \textit{low} \\ \text{SEMCAT}: \textit{zeitrel} \end{bmatrix} \begin{bmatrix} \text{SUBCAT}: \textit{low} \\ \text{SEMCAT}: \textit{zeit} \end{bmatrix}$$

$103 \bullet \texttt{t} \rightarrow \texttt{adj p}$

$$\begin{bmatrix} \text{SUBCAT}: \textit{low} \\ \text{SEMCAT}: \textit{zeitrel} \end{bmatrix} \rightarrow \begin{bmatrix} \text{CASE}: \textit{none} \\ \text{SEMCAT}: \textit{bruchteil} \end{bmatrix} \begin{bmatrix} \text{SEMCAT}: \textit{vor} \end{bmatrix}$$

$104 \bullet \texttt{t} \rightarrow \texttt{adj}$

$$\begin{bmatrix} \text{SUBCAT}: \textit{low} \\ \text{SEMCAT}: \textit{zeitrel} \end{bmatrix} \rightarrow \begin{bmatrix} \text{CASE}: \textit{none} \\ \text{SEMCAT}: \textit{bruchteil} \end{bmatrix}$$

$105 \bullet \texttt{t} \rightarrow \texttt{adj p}$

$$\begin{bmatrix} \text{SUBCAT}: \textit{low} \\ \text{SEMCAT}: \textit{zeitrel} \end{bmatrix} \rightarrow \begin{bmatrix} \text{CASE}: \textit{none} \\ \text{SEMCAT}: \textit{bruchteil} \end{bmatrix} \begin{bmatrix} \text{SEMCAT}: \textit{nach} \end{bmatrix}$$

$106 \bullet \texttt{t} \rightarrow \texttt{adj p}$

$$\begin{bmatrix} \text{SUBCAT}: \textit{high} \\ \text{SEMCAT}: \textit{zeitrel} \end{bmatrix} \rightarrow \begin{bmatrix} \text{CASE}: \textit{none} \\ \text{SEMCAT}: \textit{zahl} \end{bmatrix} \begin{bmatrix} \text{SEMCAT}: \textit{vor} \end{bmatrix}$$

107 • `t → adj p`

$$\begin{bmatrix} \text{SUBCAT}: \textit{high} \\ \text{SEMCAT}: \textit{zeitrel} \end{bmatrix} \rightarrow \begin{bmatrix} \text{CASE}: \textit{none} \\ \text{SEMCAT}: \textit{zahl} \end{bmatrix} \begin{bmatrix} \text{SEMCAT}: \textit{nach} \end{bmatrix}$$

---

108 • `t → np p`

$$\begin{bmatrix} \text{SEMCAT}: \textit{zeitrel} \end{bmatrix} \rightarrow \begin{bmatrix} \text{CASE}: \textit{nom} \\ \text{DEF}: \textit{indef} \\ \text{SEMCAT}: \textit{zeiteinheit} \end{bmatrix} \begin{bmatrix} \text{SEMCAT}: \textit{vor} \end{bmatrix}$$

---

109 • `t → np p`

$$\begin{bmatrix} \text{SEMCAT}: \textit{zeitrel} \end{bmatrix} \rightarrow \begin{bmatrix} \text{CASE}: \textit{nom} \\ \text{DEF}: \textit{indef} \\ \text{SEMCAT}: \textit{zeiteinheit} \end{bmatrix} \begin{bmatrix} \text{SEMCAT}: \textit{nach} \end{bmatrix}$$

---

110 • `tp → n`

$$\begin{bmatrix} \text{SEMCAT}: (\textit{none}, \textit{tageszeit}) \end{bmatrix} \rightarrow \begin{bmatrix} \text{CASE}: \textit{nom} \\ \text{AGR}: \begin{bmatrix} \text{NUMBER}: \textit{sing} \end{bmatrix} \\ \text{SEMCAT}: \textit{tageszeit} \end{bmatrix}$$

---

111 • `tp → adv`

$$\begin{bmatrix} \text{SEMCAT}: (\textit{none}, \textit{tageszeit}) \end{bmatrix} \rightarrow \begin{bmatrix} \text{SEMCAT}: \textit{tageszeit} \end{bmatrix}$$

---

112 • `tp → p n`

$$\begin{bmatrix} \text{SEMCAT}: (\textit{an}-\textit{dat}, \textit{tageszeit}) \end{bmatrix} \rightarrow \begin{bmatrix} \text{CASE}: \langle 14 \rangle \\ \text{AGR}: \langle 16 \rangle \\ \text{DET}: \langle 19 \rangle \\ \text{SEMCAT}: \textit{an}-\textit{dat} \end{bmatrix} \begin{bmatrix} \text{CASE}: \langle 14 \rangle \\ \text{AGR}: \langle 16 \rangle \\ \text{DET}: \langle 19 \rangle \\ \text{SEMCAT}: \textit{tageszeit} \end{bmatrix}$$

---

113 • `tp → p n`

$$\begin{bmatrix} \text{SEMCAT}: (\textit{in}-\textit{dat}, \textit{tageszeit}) \end{bmatrix} \rightarrow \begin{bmatrix} \text{CASE}: \langle 14 \rangle \\ \text{AGR}: \langle 16 \rangle \\ \text{DET}: \langle 19 \rangle \\ \text{SEMCAT}: \textit{an}-\textit{dat} \end{bmatrix} \begin{bmatrix} \text{CASE}: \langle 14 \rangle \\ \text{AGR}: \langle 16 \rangle \\ \text{DET}: \langle 19 \rangle \\ \text{SEMCAT}: \textit{tageszeit} \end{bmatrix}$$

---

114 • `tp → n`

$$\begin{bmatrix} \text{SEMCAT}: (\textit{none}, \textit{tag}) \end{bmatrix} \rightarrow \begin{bmatrix} \text{CASE}: \textit{nom} \\ \text{AGR}: \begin{bmatrix} \text{NUMBER}: \textit{sing} \end{bmatrix} \\ \text{SEMCAT}: \textit{tag} \end{bmatrix}$$

$$115 \bullet \texttt{tp} \rightarrow \texttt{adv}$$

$$\left[\ \textsc{semcat}:\ (none, tag)\ \right] \rightarrow \left[\ \textsc{semcat}:\ tag\ \right]$$

$$116 \bullet \texttt{tp} \rightarrow \texttt{p n}$$

$$\left[\ \textsc{semcat}:\ (an - dat, tag)\ \right] \rightarrow \begin{bmatrix} \textsc{case}:\ \langle 14 \rangle \\ \textsc{agr}:\ \langle 16 \rangle \\ \textsc{det}:\ \langle 19 \rangle \\ \textsc{semcat}:\ an - dat \end{bmatrix} \begin{bmatrix} \textsc{case}:\ \langle 14 \rangle \\ \textsc{agr}:\ \langle 16 \rangle \\ \textsc{det}:\ \langle 19 \rangle \\ \textsc{semcat}:\ tag \end{bmatrix}$$

Anhang D

Verwendetes Korpus

Im folgenden sind alle Sätze des experimentellen Testkorpus aufgelist. Vor jeden Satz ist vorangestellt ob er bereits von der Grundgrammatik geparst werden konnte (P) nach dem Lernverfahren erfolgreich geparst werden konnte (L) oder nicht gelernt wurde (-). Bei den mit einem Stern (*) gekennzeichneten Sätzen dauerte das Lernverfahren mehr als 100 Stunden und wurde daher abgebrochen. Ebenfalls ist angegeben ob der Satz als grammatisch wohlgeformt (G) oder nicht (-) erachtet wurde.

Tabelle D.1: Ergebnisübersicht

Ergebnis	Anzahl Sätze
PG ursprünglich geparste Sätze	40
LG gelernte Sätze	45
UG nicht gelernte Sätze	23
-- ungrammatikalische Sätze	29
*G zu langsam	3
insgesamt	140

```
PG 499 ich moechte gerne nach paris
PG 500 kann man auch ohne umsteigen fahren
PG 501 guten tag wann geht morgen vormittag ein zug nach frankfurt
PG 502 wann geht der naechste zug nach mannheim
PG 503 kann ich am samstag abend nach zweiundzwanzig uhr noch von frankfurt
       nach muenchen kommen
PG 504 ich moechte am ersten feiertag nach koblenz fahren
PG 505 geht heute noch ein zug nach hannover
PG 506 ich will am montag um acht uhr nach stuttgart
PG 507 darf ich eine icfahrt zwischendurch unterbrechen
PG 508 ich moechte morgen abend nach koeln fahren
PG 509 wann geht heute der letzte zug von ulm nach muenchen
LG 510 ich moechte ab ulm mit dem zug nach aschaffenburg fahren
PG 511 kann ich heute noch mit einem intercity nach hamburg fahren
PG 512 wann kann ich morgen frueh nach muenchen fahren
```

```
LG 513 ich muss ueber hannover nach hamburg fahren
PG 514 wie kann ich von ulm nach duesseldorf kommen
PG 515 kann man heute noch von ulm nach koblenz kommen
LG 516 koennten sie mir eine verbindung nach emmerich grenzbahnhof geben
PG 517 kann man direkt von ulm nach frankfurt fahren
LG 518 koennen sie mir sagen wann ich spaetestens in muenchen losfahren muss
       wenn ich noch vor zehn uhr in augsburg sein will
-- 519 jetzt am samstag nach ulm wann fahren da morgens zuege
PG 520 ich brauche den naechsten zug nach muenchen
PG 521 ich will morgen abend nach frankfurt
LG 522 ich moechte heute in vierzehn tagen nach rom fahren
PG 523 wann kann ich heute noch nach ulm fahren
PG 524 wann geht morgen frueh ein zug nach frankfurt
PG 525 ich muss nach muenchen
UG 526 guten morgen ich moechte heute nach neun von muenchen nach hamburg
LG 529 ich brauche fuer uebernaechsten montag nachmittag eine zugverbindung
       von badenbaden nach oldenburg
LG 530 ich moechte von muenchen ueber nuernberg nach hamburg fahren
LG 532 ich brauche heute eine verbindung ab koeln
LG 533 ich muesste ueber duesseldorf nach hamburg fahren
PG 534 ich benoetige eine zugverbindung von muenchen nach aachen
LG 536 ich suche zuege muenchen nuernberg etwa um mittag
LG 537 ich brauche eine icverbindung von heidelberg nach bebra naechste woche
PG 538 ja ich moechte gerne nach hamburg fahren
-- 539 bitte eine verbindung von muenchen nach frankfurt
LG 540 guten morgen ich moechte heute nach badenbaden von ulm ueber wuerzburg
       fahren
LG 541 ich moechte in vierzehn tagen von muenchen ueber hannover nach hamburg
       fahren
LG 542 ich brauche informationen ueber zugverbindungen nach saarbruecken
PG 543 wann faehrt samstags der letzte zug nach koeln los
PG 544 ich moechte mit dem ersten zug von muenchen nach nuernberg fahren
LG 545 ich moechte heute mit umsteigen in bremen von muenchen nach bremerhaven
       fahren
UG 546 ich brauche diese woche noch einen intercity von muenchen nach hamburg
       ueber nuernberg
PG 547 gibt es heute abend noch einen zug nach wuerzburg
LG 549 morgen muss ich um sechs in mannheim sein
LG 551 guten tag ich braeuchte eine verbindung nach mannheim fuer morgen
UG 554 ich muss nach dortmund aber mit einem kurzaufenthalt in koeln
UG 555 gibt es einen nachtzug der am dreiundzwanzigsten zwoelften morgens in
       ulm ankommt
UG 556 wie lange faehrt man von hier nach osnabrueck ungefaehr
UG 557 nennen sie mir die guenstigste zugverbindung nach kiel
-- 560 um spaetestens um sechs uhr in hamburg zu sein wann muss ich da hier
       losfahren
LG 561 welcher zug faehrt nicht vor neun uhr nach dortmund ueber koeln
PG 562 fahren keine zuege vor siebzehn uhr von koeln nach dortmund
```

LG 565 ich haette gerne eine auskunft ueber die abfahrtszeit der zuege die ab zirka achtzehn uhr von nuernberg nach augsburg fahren
LG 566 ich moechte an einem wochentag nach hamburg fahren
LG 567 gruess gott koennten sie mir bitte sagen ob um zirka fuenf uhr morgens ein zug von muenchen nach ulm geht
PG 568 komme ich spaetabends noch von hamburg nach muenster
LG 571 gruess gott koennten sie mir bitte die abfahrtszeiten der zuege nennen die von ulm nach frankfurt fahren
PG 572 welchen zug muss ich nehmen um gegen zehn uhr in wuerzburg zu sein
UG 574 wie lange dauert bitte eine fahrt von muenchen nach hamburg
LG 575 wann faehrt neujahr so gegen mittag ein zug nach ulm
LG 576 ich moechte gern wissen wann ich hier spaetestens losfahren muss wenn ich zirka gegen drei uhr nachmittags in ulm sein muss
-- 577 ich brauche fuer morgen eine fahrkarte nach koeln bitte suchen sie mir eine guenstige verbindung
LG 578 koennen sie mir eine verbindung augsburg kiel geben bei der ich morgens losfahren kann
PG 581 faehrt immer noch um acht uhr vierundzwanzig ein intercity nach mannheim
LG 582 ich muss nach hamburg aber ich moechte nicht in der nacht ankommen
PG 583 welche moeglichkeiten habe ich samstags vormittags nach augsburg zu fahren
-- 585 nach regensburg dienstag morgen gegen acht uhr wann fahren da zuege
*G 587 koennen sie mir sagen wann ich hier wegfahren muss wenn ich spaetestens morgen frueh um halb acht in hamburg sein will
LG 588 ich benoetige auskunft ueber ankommende zuege
LG 589 koennen sie mir auch auskunft geben ueber zuege die nach hamburg fahren
-- 590 von muenchen nach koeln ueber augsburg ankunft in koeln gegen drei uhr nachmittags
UG 593 suche einen zug von hier nach ulm der vor siebzehn uhr dort ankommt
-- 595 nach dortmund bitte ich moechte dienstag morgen um neun uhr da sein
UG 597 ich moechte so nach bremen fahren dass ich gegen siebzehn uhr ankomme
PG 598 ich wollte fragen ob der zug von paris nach muenchen einen schlafwagen dabei hat
LG 600 verkehrt der acht uhr dreiundzwanzig zug nach frankfurt auch an heiligabend
PG 601 ich muss am rosenmontag nach koeln
-- 602 wie bitte
PG 699 ich moechte gerne nach paris
PG 700 kann man auch ohne umsteigen fahren
LG 701 ich haette gerne eine zugverbindung fuer morgen von regensburg nach dortmund ueber koeln
-- 704 morgens gegen zehn jedenfalls nicht nach nicht vor neun
UG 707 na das geht
PG 708 geht von etterzhausen nach hindelang ueberhaupt eine bahnverbindung
-- 710 mir fehlt noch die verbindung fuer samstag frueh
-- 711 eine zugverbindung bitte nach oldenburg wobei der zug spaetestens um neun uhr ankommen sollte

```
LG 712 geht es nicht eher
-- 713 aber sicher
UG 714 ja dann nehme ich den um zehn uhr vierundzwanzig
LG 715 ich braeuchte eine zugverbindung von regensburg nach hamburg bei der
       man moeglichst wenig umsteigen muss
LG 716 wieviel stunden faehrt man denn da
-- 717 ja dann so am vormittag am spaeteren vormittag vielleicht
*G 719 also ich moechte eine zugverbindung von regensburg nach dortmund ueber
       koeln mit mindestens zwei stunden aufenthalt in koeln
-- 720 neun uhr
LG 721 kann ich nach dortmund ueber koeln fahren
-- 722 ja gut
-- 724 ah ja am vormittag so um acht uhr vielleicht
UG 725 man braucht dann von nuernberg nach hamburg nicht mehr umzusteigen nein
LG 728 ja ich braeuchte einen liegewagen von regensburg nach hamburg
-- 729 moeglichst die fahrt ueber nacht
UG 730 naja okay das passt schon
UG 732 ja ungefaehr eine woche moecht ich an der ostsee bleiben
-- 739 danke
UG 743 dann nehm ich lieber den spaeteren
-- 747 naja etwa vier stunden
-- 749 ja
-- 750 ja gut prima danke
LG 751 koennten sie mir bitte zuege von regensburg nach frankfurt heute abend
       sagen
-- 755 von beiden zuegen
-- 756 gut danke
LG 757 ich braeuchte eine zugverbindung von regensburg nach hamburg moeglichst
       ohne umsteigen
-- 758 ab zehn uhr vormittags
LG 759 koennten sie mir bitte sagen welche zuege heute abend von regensburg
       nach frankfurt gehen
PG 761 gibt es noch andere moeglichkeiten
UG 762 also ich brauche eine fahrkarte nach mannheim die wenn moeglich auch
       fuer die ganze naechste woche gilt
LG 763 gelten diese fahrkarten fuer mehrere tage
LG 766 ich brauche fuer den zweiundzwanzigsten dezember einen zug nach
       oldenburg gegen abend
LG 767 ich moechte aber ungefaehr um zweiundzwanzig uhr in oldenburg sein
UG 768 wann geht bitte ein zug nach hamburg mit moeglichst wenig ja bahnhoefen
UG 769 wann geht bitte naechste woche ein zug von regensburg nach hamburg am
       vormittag
UG 770 auf welchem gleis geht der zug weg von regensburg aus
PG 773 gibt es auch eine verbindung gegen zwanzig uhr
LG 774 ich brauche eine zugverbindung regensburg dortmund mit einem
       zwischenstopp in koeln
-- 775 morgens um acht
-- 777 ungefaehr eine stunde
```

```
*G 778 wann geht bitte morgen ein zug von regensburg nach oldenburg so dass
       ich in oldenburg um neun uhr morgens ankomme
LG 779 ja wann geht der zug bitte ab von regensburg
PG 780 hat der zug schlafwagenabteile
UG 781 ich braeuchte eine verbindung fuer dienstag von hier nach kiel an der
       ostsee
UG 782 ich wuerde gerne gegen zehn uhr vormittags fahren
-- 784 nein
UG 786 das kann ich nicht genau sagen
LG 788 ich braeuchte die abfahrtszeiten aller zuege heute von regensburg nach
       frankfurt
UG 789 ich braeuchte die abfahrtszeiten aller zuege nach frankfurt ab
       achtzehn uhr mit angabe des bahngleises
-- 791 ich brauche zugverbindung nuernberg regensburg nach zwanzig uhr samstag
LG 797 wann faehrt bitte ein zug von regensburg nach dortmund am montag am
       montag morgen
PG 801 ich muss am rosenmontag nach koeln
-- 802 wie bitte
```