Aus: Computational Linguistics. An International Handbook on Computer Oriented Language Research and Applications. Edited by I. S. Bátori, W. Lenders, W. Putschke. Berlin: De Gruyter 1989. (Handbücher zur Sprach- und Kommunikationswissenschaft.) S. 348 - 377.

31. Parsing natürlicher Sprachen: Grundlagen

1. Vorbemerkung

Die Konzeption des Artikels. Ziel der folgenden Ausführungen ist es, die Information zu bieten, die man benötigt, um einen Parser zu konstruieren. Es gibt allerdings zahlreiche Möglichkeiten, wie die Parsingkomponente im Rahmen eines sprachverarbeitenden Computersystems gestaltet werden kann. Im vorliegenden Teil (Art. 31) wird versucht, systematisch die Alternativen zusammenzustellen, unter denen eine Auswahl getroffen werden muß, wenn man einen Parser entwirft. Dieser Teil ist als Entscheidungshilfe gedacht. Im folgenden Teil (Art. 32) werden exemplarische Realisierungen von Parsern dargestellt. An ihnen kann man das Zusammenspiel sowie Vor- und Nachteile der verschiedenen Einzelentscheidungen studieren. Es sind natürlich viele weitere Kombinationen der vorhandenen Mittel und vielleicht auch ganz neue Wege denkbar. Die beispielhafte Darstellung von Parsern soll den Leser letztlich dazu anregen, selbst einen neuen Parser zu erfinden. Eine Literaturauswahl befindet sich am Ende von Art. 32.

Die Beschreibungsebene. Der Artikel bewegt sich bewußt nicht auf einer mathematischen Beschreibungsebene. Es soll relativ direkt dargestellt werden, wie sich die Phänomene der natürlichen Sprache mittels bestimmter Programmiertechniken erfassen lassen. Dadurch hoffe ich, das Transferproblem zu umgehen, das zwischen den Kenntnissen des Sprachwissenschaftlers, des Informatikers und des Programmierers besteht. Ich nehme an, daß es für viele Leser leichter ist, aus dem Verständnis konkreter Beispiele heraus generelle Zusammenhänge zu erfassen, als umgekehrt eine abstrakte mathematische Darstellung auf linguistische Fakten und die unmittelbare Programmieraufgabe zu beziehen. Der mathematisch vorgebildete Leser findet in der angegebenen Literatur genügend formale Darstellungen.

2. Grundlagen

2.1. Definitionen

Parsing wird enger oder weiter definiert, je nach den hinzugenommenen Bestimmungsstücken. Man versteht darunter

(i) die Zuordnung einer Strukturbeschreibung zu Zeichenfolgen;

(ii) die Zuordnung einer Strukturbeschreibung zu Sätzen als Einheiten einer besonderen Ebene des Sprachsystems;

(iii) die Zuordnung einer disambiguierten semantischen Repräsentation zu einer Äußerung als Teil des Sprachverstehens.

Die drei Definitionen sind je für die Disziplinen Informatik, Linguistik, Künstliche Intelligenz charakteristisch. Ein Parser ist ein Computerprogramm, das im Sinne von (i) bis (iii) einer natürlichsprachigen Eingabe eine Struktur zuordnet.

Die formale Definition. Bestimmte Objekte bestehen aus Folgen von Elementen. Die Folgen sind aber nicht beliebig, sondern die Elemente haben eine bestimmte Distribution. Die Objekte weisen implizit einen bestimmten Aufbau auf. Dieser Aufbau ist ihre Syntax in einem formalen Verständnis des Begriffs. Im allgemeinsten Sinn von Parsing geht es darum, diesen Aufbau zu erkennen und explizit zu machen (vgl. Foster 1971, 6 f.). Nach dieser Definition spielt es keine Rolle, von welcher Art die Objekte und Elemente sind, ob es sich um Teile eines Programms handelt oder um Einheiten einer natürlichen Sprache wie Silben, Sätze, Texte. In der Tat sind ein Großteil der im folgenden dargestellten Parsingmethoden anwendbar, wo immer es gilt, Strukturen aufzudecken.

Die linguistische Definition. In der Linguistik wird unter Syntax nicht jedweder Aufbau verstanden, sondern vielmehr der einer bestimmten Ebene des Sprachsystems. Die genaue Bestimmung dieser Ebene ist theorieabhängig. Die vorherrschende Ansicht ist die, daß Sätze oder satzwertige Einheiten eine herausragende Rolle im Sprachsystem spielen. Als Syntax einer Sprachbeschreibung wird daher im allgemeinen die Komponente bezeichnet, in welcher der Aufbau der Sätze der betreffenden Sprache beschrieben wird (Kratzer/Pause/v. Stechow 1973, 3). Häufig wird der Begriff Syntax jedoch noch eingeschränkt, indem zwischen Satzsyntax und Satzsemantik unterschieden wird. Während die Syntax den Aufbau der Sätze festlegt, behandelt die Satzsemantik die Frage, welche Beziehungen zwischen den Sätzen bestehen. Als Kriterium der Satzsemantik wird vor allem das Wahrheitswertverhältnis der Sätze herangezogen. Dies darf nicht so mißverstanden werden, als sei die Beschreibung des Aufbaus von Sätzen ohne Berücksichtigung der daran geknüpften Bedeutungsunterschiede möglich. Die Syntax hat vielmehr genau die Einheiten zu liefern, welche für die Bedeutungsbeziehungen konstituierend sind. Ein Parser im linguistischen Sinn ordnet demnach Sätzen und Satzteilen eine semantisch interpretierbare Struktur zu. (Hays 1966a, 73; Hays 1967, 106; Brockhaus 1971, 31; Kilbury 1984, 1 f.)

Die Definition in wissensbasierten Systemen. Natürlichsprachige Computersysteme sollen aktuelle Eingaben verarbeiten. Sie haben es also nicht nur mit Sprache als System, sondern mit Sprache im Gebrauch zu tun. Äußerungen müssen tatsächlich interpretiert, d. h. im pragmatischen Kontext zu einem aktuellen Objektbereich in Beziehung gesetzt werden. Das Modell des Objektbereiches liegt in den Systemen in Form einer Wissensbasis vor. Das Programm soll über dieser Wissensbasis Operationen durchführen, u. a. Information auffinden, Schlußfolgerungen ziehen, Antworten generieren. Dafür ist eine geeignete semantische Repräsentation eine Voraussetzung. Anstelle der Ebenen des Sprachsystems tritt daher in natürlichsprachigen Systemen eine andere Aufteilung in den Vordergrund: (i) eine Analysekomponente, welche natürlichsprachige Eingaben in die semantische Repräsentation überführt und sie dabei disambiguiert, (ii) eine Deduktionskomponente, die semantische Repräsentationen verarbeitet, (iii) eine Generierungskomponente, welche semantische Repräsentationen in natürlichsprachige Ausgaben überführt. Die Unterscheidung von linguistischen Ebenen ist für jede dieser Komponenten ein internes Problem. Wesentlich ist, daß jede Komponente ein zusammenhängendes Design hat, sodaß sie als ganze ihre Aufgabe erfüllt. Die Aufgabe des Parsers geht hier in der Aufgabe der Analysekomponente auf, Texte in eine Wissensrepräsentation zu überführen. (Vgl. Wilks 1983; Christaller 1985.)

2.2. Das Verhältnis von Grammatikformalismus, Grammatik und Parser

Das Problem der Grammatik. In systematischer Mehrdeutigkeit werden die Eigenschaften einer Sprache sowie die Beschreibung dieser Eigenschaften „Grammatik" genannt. Der Begriff ist hier umfassend gemeint. Er soll Satzsyntax und Satzsemantik einschließen, ebenso den Wortschatz. Ein Parser ist genauer zu bestimmen als ein Programm, das einer vorliegenden Zeichenkette relativ zu einer gegebenen Grammatik eine Beschreibung zuweist. Für Kunstsprachen bildet die Grammatik kein Problem. Sie werden ja gerade dadurch geschaffen, daß man für sie ausdrücklich eine Grammatik festlegt. Man hat es zudem in der Hand, Kunstsprachen so zu ändern, daß sie sich einfacher beschreiben und leichter parsen lassen. Im Unterschied dazu müssen die Eigenschaften natürlicher Sprachen erst empirisch ermittelt werden. Auch lassen sich natürliche Sprachen nicht zum Zwecke der leichteren Beschreibung und Verarbeitung manipulieren. Sie sind einfach gegeben. Das Schreiben einer Grammatik bildet beim natürlichsprachigen Parsing daher einen wesentlichen Teil der Aufgabe. Dies macht einen entscheidenden Unterschied zwischen dem Parsing formaler Sprachen, wie es in der Informatik gelehrt wird, und der maschinellen Analyse natürlicher Sprachen aus.

Grammatikformalismus, Grammatik und Parser. Es stellen sich drei Aufgaben. (i) Zunächst muß ein brauchbares Beschreibungsinstrumentarium für Grammatiken natürlicher Sprachen überhaupt geschaffen werden. Die Lösung dieser Aufgabe hängt natürlich von theoretischen Annahmen darüber ab, welcher Art die sprachlichen Phänomene sind, und welches Grammatikmodell diese

Phänomene adäquat abbildet. Solche Annahmen haben zu verschiedenen Grammatikformalismen geführt. So unverantwortlich es wäre, einen Parser zu konstruieren, ohne den Stand der theoretischen Grammatikforschung zu berücksichtigen, so sehr kann die Grammatiktheorie von den Erfahrungen beim Parsen der verschiedenen Modelle profitieren. (ii) Sodann ist für jede einzelne Sprache, die der Parser verarbeiten soll, tatsächlich eine Grammatik in dem gewählten Formalismus zu schreiben. Diese Aufgabe ist eine empirische, die angesichts der Komplexität der einzelnen Sprache noch für lange Zeit das schwierigste Problem darstellen dürfte. Parserentwicklung ist hier an die Ergebnisse der einzelsprachigen Linguistik gebunden. (iii) Schließlich muß der Parser selbst entwickelt werden, d. h. ein mechanisches Verfahren, das auf irgendeine Weise entscheidet, ob eine vorliegende Äußerung unter eine der Beschreibungen der Grammatik fällt, und wenn ja, unter welche.

Grammatik-Parser-Kombination. In einer heute gängigen Terminologie läßt sich der Zusammenhang von Grammatikformalismus, Grammatik und Parser als der von Wissensrepräsentation, Wissen und Wissensverarbeitung charakterisieren (vgl. Christaller 1985, 162). Für jeden dieser drei Problembereiche gelten eigene Kriterien, z. B. Ausdruckstärke für den Formalismus, Korrektheit für die Grammatik, Effizienz für die Verarbeitung. Alle drei sind entscheidend. Eine geeignete Grammatik-Parser-Kombination ist daher die wichtigste Aufgabe, die es beim Entwurf eines maschinellen Analysesystems zu lösen gilt. Rein äußerlich gibt es zwei Möglichkeiten für das Design einer Analysekomponente: (i) Grammatik und Parser sind getrennt. (ii) Grammatik und Parser sind integriert.

Trennung von Grammatik und Parser. Die Grammatik existiert unabhängig vom Programm als eine Datenmenge. Der Parser liest sowohl die natürlichsprachige Eingabe wie auch die Grammatik ein, vergleicht beide und erzeugt eine Ausgabe. Die Grammatik ist hier eine Beschreibung der Objektsprache in deklarativer Form, d. h. es wird dargestellt, wie die grammatischen Strukturen aussehen, nicht wie man sie erzeugt oder erkennt. Der Parser fußt nur auf der Syntax des Grammatikformalismus, nicht auf den Inhalten der Grammatik. Je nach der Form der grammatischen Aussagen wendet das Programm bestimmte Erkennungsstrategien auf die natürlichsprachige Eingabe an. In der Terminologie der Informatik handelt es sich bei dem Parser um einen Interpreter (vgl. Aho/Sethi/Ullman 1986, 3). Das System hat folgende Architektur:

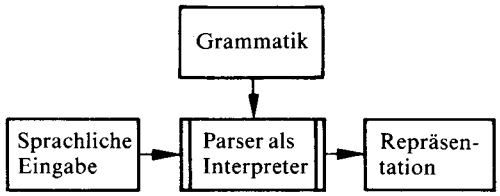

Abb. 31.1: Interpretierender Parser

Integration von Grammatik und Parser. In diesem Fall wird das Wissen über die Sprache schon unter dem Aspekt der Analyse formuliert. Die Grammatik ist hier eine Beschreibung der Objektsprache in prozeduraler Form, d. h. sie besteht aus einer Menge von Anweisungen zum Erkennen von Ausdrücken. Häufig wird eine allgemein zur Verfügung stehende Programmiersprache, wie LISP, zum Schreiben der Grammatik benutzt. (Winograd 1972, Charniak/Riesbeck/McDermott 1979). Der Parser führt die Anweisungen einfach aus. Es gibt keine Trennung von grammatischen Daten und Programm. Die Daten sind das Programm. Das System hat folgende Architektur:

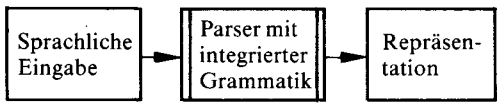

Abb. 31.2: Prozeduraler Parser

Parsergenerator. Die beiden Möglichkeiten lassen sich verbinden. Wie im ersten Fall existiert die Grammatik zunächst unabhängig vom Parser. Sie wird jedoch nicht in dieser deklarativen Form zum Analysieren benutzt, sondern erst in eine prozedurale Form gebracht. Dies geschieht automatisch mit Hilfe einer zusätzlichen Komponente im Sy-

stem: dem Parsergenerator (Aho/Sethi/Ullman 1986, 257 f.). Dieser ist im Sinne der Informatik ein Compiler. Er erzeugt, unter Umständen über einige Zwischenstufen, auf der Grundlage der Syntax des Grammatikformalismus, ein ausführbares Programm, das die Grammatik als unmittelbare Anweisung zum Erkennen der sprachlichen Eingabe enthält. Das System hat folgende Architektur:

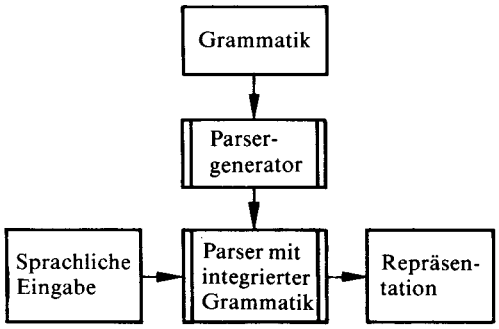

Abb. 31.3: Compilierter Parser

Vorteile der Trennung von Grammatik und Parser. In natürlichsprachigen Analysesystemen sollten die Beschreibung der Sprache und die Verarbeitung im Parser getrennt werden. Es gibt dafür folgende Gründe. (i) Getrennt können sowohl die Grammatik als auch der Parser leichter verbessert werden. Fehlerquellen sowie die zu treffenden Maßnahmen sind bei beiden grundverschieden: Die Grammatik muß auf Übereinstimmung mit den linguistischen Fakten überprüft werden; das Programm verlangt vor allem Korrekturen in der Kodierung und der Ablauflogik. Da sowohl Fehler in der Grammatik wie Programmfehler zum Scheitern der Analyse führen, ist in einem prozeduralen System schon die Lokalisierung des Versagens ein Problem. (ii) Wenn der Parser nur an das Format der grammatischen Beschreibung gebunden ist, nicht aber an deren Inhalt, kann das Programm fertig gestellt werden, ohne daß die Grammtik schon endgültig vorliegt. Änderungen der Grammatik sind beim heutigen Stand der Wissenschaft praktisch unbegrenzt notwendig. (iii) Bei der Trennung von Daten und Programm können verschiedene Sprachen mit demselben Parser verarbeitet werden. Nur die Grammatik als Datensatz muß jeweils ausgetauscht werden. (iv) Im Idealfall kann dieselbe Grammatik auch vom Generator benutzt werden, also von der Komponente, welche die interne Repräsentation in natürliche Sprache zurückübersetzt. (v) Eine vom Programm unbhängige Grammatik kann so geschrieben sein, daß sie auch außerhalb des Computersystems benutzbar ist. Eine in einer Programmiersprache verfaßte Grammatik ist dagegen für Außenstehende nicht lesbar. Da das Problem der vollständigen Beschreibung natürlicher Sprachen noch lange nicht gelöst ist und sich auch nur mit Hilfe des Computers wird lösen lassen, ist der leichte Transfer des Wissens aus der Linguistik in die Computeranwendung und umgekehrt ein entscheidender Gesichtspunkt.

Simulation psychischer Prozesse. Anders stellt sich die Frage nach dem Verhältnis von Grammatik und Parser wenn die Analyse mit dem menschlichen Sprachverstehen nicht nur ergebnisgleich sondern auch ablaufgleich (Bátori 1981a) sein soll. In diesem Fall werden mit den Prozeduren des Parsers (psycho)linguistische Ansprüche verknüpft. Der Parser hätte damit ebenso wie die Grammatik den Status einer linguistischen Theorie. Beschreibt die Grammatik sprachliche Strukturen, so beschreibt der Parser sprachliche Prozesse. Er müßte so konstruiert sein, daß er eine Erklärung dafür abgeben könnte, wie Menschen offensichtlich relativ mühelos sprachliche Strukturen erkennen. Derartige Ansprüche werden besonders im Rahmen der Künstlichen Intelligenz erhoben, ohne daß jedoch bisher für bestimmte Parsingverfahren empirische Evidenz erreicht werden konnte. (Vgl. Kaplan 1972; Marslen-Wilson 1975; Fodor/Frazier 1980; Cottrell/Small 1984; Krems 1984; Crain/Fodor 1985 sowie die unter 4.6. angegebene Literatur.)

Das Verhältnis von Grammatik und Weltwissen. Wir haben oben zwischen Satzsyntax und Satzsemantik unterschieden. Erstere beschreibt den Aufbau von Sätzen, letztere die logisch-semantischen Beziehungen zwischen Sätzen. Dabei liefert die Satzsyntax die Einheiten, auf die sich die Beschreibung der Satzsemantik stützt. Die beiden Komponenten werden zwar unterschiedliche Regelmengen beinhalten (Aufbauregeln hier, Deduktionsregeln dort), doch sind sie andererseits so aufeinander bezogen, daß man nicht die Daten der einen Komponente beibehalten und die der anderen austauschen könnte. Beide zusammen definieren die Sprache als

System. Als Oberbegriff für beide benutzen wir den Terminus Grammatik. Anders verhält es sich nun mit der Semantik, soweit damit die Beziehung von Äußerungen zum Objektbereich gemeint ist. Zur Auflösung der Referenz ist natürlich ein aktuelles Weltmodell erforderlich. Es ist aber auch bekannt, daß das Weltwissen bei der Disambiguierung grammatischer Mehrdeutigkeiten eine große Rolle spielt. Man hat hier die Wahl zwischen folgenden Architekturen: (i) Das Wissen über Zusammenhänge in der Welt existiert als getrennter Datensatz neben der Grammatik und wird von der Analysekomponente interpretiert. (ii) Das Weltwissen wird in die Grammatik integriert und unmittelbar zum Parsen genutzt. Letztere Option, häufig zusammen mit der prozeduralen Version der Grammatik, wird in der Künstlichen Intelligenz bevorzugt. Für die Tennung von Weltwissen und Grammatik sprechen jedoch die gleichen Gründe wie schon für die Trennung von Grammatik und Parser. Zudem verspricht diese Trennung auch theoretisch mehr Einsichten. Während integrierte Systeme auf einen bestimmten Objektbereich festgelegt sind und daher über das regelhafte Verhältnis zwischen Sprache und Welt eigentlich gar nichts offenbaren, darf man wohl ein Sprachverstehenssystem mit beliebig wechselnden Wissensbeständen als das große Projekt der Zukunft betrachten. (Siehe auch 4.6.)

3. Parameter für die Grammatik

3.1. Phänomene der natürlichen Sprache

Linguistische Rahmenbedingungen. Es hat wenig Sinn, viele Mühe in die Programmierung eines Parsers zu stecken, wenn der Grammatikansatz zur Behandlung der sprachlichen Phänomene nichts taugt. Linguistisches Problembewußtsein ist daher die wichtigste Eigenschaft des Entwicklers eines natürlichsprachigen Computersystems. Auch bei bescheidenen Zielen wird man, angesichts des immer beträchtlichen Aufwands, darauf sehen, daß der grammatische Ansatz erweiterbar ist. Die folgende Zusammenstellung von Eigenschaften der natürlichen Sprache (anhand des Deutschen) soll einen Eindruck davon geben, was es dabei zu berücksichtigen gilt. Die Aufstellung ist ohne jeden Anspruch auf Vollständigkeit.

Abgrenzung der Analyseeinheiten. Schon das Erkennen der Grundelemente ist in der natürlichen Sprache ein Problem. In geschriebenen Texten kann man von Buchstabenfolgen zwischen Leerzeichen oder Sonderzeichen ausgehen. Die so ermittelten Wörter müssen aber oft noch weiter zerlegt werden. Bei der Analyse gesprochener Sprache ist nicht einmal sicher, welcher Laut im linguistischen Sinn einem physikalischen Schallereignis entspricht. Erst recht ist unmöglich, Wortgrenzen unabhängig vom Kontext zu erkennen. Generell ist für natürliche Sprachen der hermeneutische Zirkel charakteristisch: Die Teile leiten sich ab aus dem Ganzen, das Ganze leitet sich ab aus den Teilen. Die Schwierigkeit besteht darin, diesen Zirkel zu durchbrechen und irgendwo zu beginnen.

Analysedimensionen. Jeder Ausdruck in einer Äußerung hat gleichzeitig mehrere Eigenschaften, die er auf verschiedene Weise mit anderen Ausdrücken teilt. Für die Beschreibung stellt sich die Aufgabe, die Ausdrücke mit Hilfe geeigneter Kategorien zu klassifizieren. Die Eigenschaften lassen sich gruppieren, so daß sich gewissermaßen drei Dimensionen ergeben:

(i) die lexikalische Bedeutung,
(ii) die morpho-syntaktische Form,
(iii) die syntagmatische Funktion.

(i) *geben, gibst, gabst* haben z. B. dieselbe lexikalische Bedeutung, nämlich 'geben'. (ii) *gibt, liebt, singt* haben dieselben morpho-syntaktischen Merkmale, nämlich '3. Person' und 'Singular'. (iii) *Jens / der Mann / wer das gesagt hat* erfüllen im Zusammenhang mit *... lügt* dieselbe syntagmatische Funktion, nämlich die des Subjekts. Ein sprachlicher Ausdruck hat immer teil an allen drei Dimensionen. Neben geeigneten Kategorien für jede Dimension muß die Grammatik auch über Mittel verfügen, die Korrespondenz zwischen den Dimensionen darzustellen.

Mehrdeutigkeit. Unterschiedliche Bedeutungs-Form-Funktionskomplexe fallen oft in einem Ausdruck zusammen. Für sich genommen erscheint der Ausdruck als mehrdeutig. Erst der Kontext zeigt, welche Eigenschaften der Ausdruck in der Äußerung tatsächlich hat. In manchen Fällen bleibt die Mehrdeutigkeit auch bestehen, sei es weil der Kontext, in dem sie sich auflösen würde, z. B. das Wissen über den Objektbereich, nicht zur Verfügung steht, sei es, weil die Äußerung mit Ab-

sicht mehrdeutig war. Mehrdeutigkeit kann innerhalb jeder der Dimensionen auftreten: (i) Es gibt lexikalische Homonymien, wie z. B. *Schloß*. (ii) Zahlreich sind morpho-syntaktische Homonymien, wie z. B. das Wort *der*, das Pronomen oder Artikel in verschiedenen Kasus, Numeri und Genera sein kann. (iii) Schließlich gibt es Mehrdeutigkeiten hinsichtlich der syntagmatischen Funktion, z. B. von *zum Glück* im Satz *Dem Junggesellen fehlt zum Glück die Frau*. Oft sind an einer Mehrdeutigkeit alle drei Dimensionen beteiligt, z. B. in *Ich möchte gern rumkugeln*, wo (wenn man von der gesprochenen Form ausgeht) *rumkugeln* entweder die Bedeutung von 'herumkugeln' hat, ein Verb im Infinitiv ist und als Prädikatsteil fungiert, oder die Bedeutung von 'Kugeln mit Rum' hat, ein Substantiv im Akkusativ Plural ist und als Objekt fungiert. Beispiele für komplexe Mehrdeutigkeiten liefert auch die Wortbildung, z. B. im Falle von *Staubecken*, das zerlegt werden kann in die lexikalischen Bedeutungen 'Stau' und 'Becken' und dann morpho-syntaktisch ein Substantiv im Singular oder Plural darstellt oder in 'Staub' und 'Ecken' und dann nur Plural sein kann.

Kongruenz. Die morpho-syntaktischen Merkmale der Elemente einer Äußerung bedingen sich gegenseitig, und zwar in sich kreuzenden Kombinationen und über weite Abstände hinweg. Man vergleiche *der alte Herr, der sich im Wartezimmer befindet* und *die alten Damen, die sich im Wartezimmer befinden*. Kasus, Genus, Numerus müssen übereinstimmen beim Artikel, Adjektiv und Substantiv, Genus und Numerus beim Substantiv und dem Relativpronomen, Person und Numerus beim Substantiv, dem Reflexivpronomen und dem Verb. Die Kongruenz im Deutschen stellt hohe Ansprüche an die Ausdruckskraft des Grammatikformalismus und an die Effizienz der Verarbeitung. Zur Auflösung der Homonymie der Wortformen ist sie aber auch sehr nützlich.

Valenz. Welche Elemente in einer Äußerung miteinander vorkommen können, hängt von diesen selbst ab. Dies gilt nicht nur für das Verhältnis des Verbs und seiner Ergänzungen, das zuerst mit dem Terminus 'Valenz' belegt worden ist, sondern ganz allgemein. Im Umkreis der generativen Grammatik wird diese Abhängigkeit der syntaktischen Strukturen vom lexikalischen Material unter dem Terminus '(strikte) Subkategorisierung' behandelt. Letztlich ist es die lexikalische Bedeutung der Elemente, auf der die syntaktischen Strukturen fußen. Die Bedeutung von *schlafen* impliziert z. B. den Schläfer, die Bedeutung von *lieben* den Liebenden und den Geliebten, die Bedeutung von *geben* den Gebenden, das Gegebene und den Empfänger. Wo diese Verben verwendet werden, treten gleichzeitig Ausdrücke auf, die diese Größen bezeichnen. Da das Verb selbst die inhaltliche Rolle der Ergänzungen schon hinreichend bestimmt, reichen zur Klassifikation der unterschiedlichen Funktionen formale Begriffe, wie Subjekt, Akkusativobjekt und Dativobjekt. Die natürliche Sprache bedient sich der Morphologie und der Wortstellung, um die syntagmatischen Funktionen auseinanderzuhalten. Betrachten wir auch noch ein Substantiv, z. B. *Brücke*. Eine Brücke ist ein Bauwerk, das ein Hindernis überspannt und zwei Ufer miteinander verbindet. Dieses Faktum führt syntaktisch zu den Ergänzungen des Substantivs *Brücke* mit der Präposition *über* und *zwischen*, wobei diese Konstituenten das Überspannte und das Verbundene bezeichnen. Die Valenz schafft die molekulare Verbindung zwischen den drei Dimensionen: Ein Element hat eine bestimmte lexikalische Bedeutung; es verlangt infolgedessen nach bestimmten Ergänzungen und legt gleichzeitig die Funktion wie die morpho-syntaktische Form dieser Ergänzungen fest. Eine hinreichende Beschränkung der Syntax und eine adäquate funktionale Interpretation von Äußerungen ist nur möglich, wenn der Grammatikformalismus diese Selektionsbeziehungen darzustellen vermag.

Idiomatische Wendungen. Feste Syntagmen, wie *auf dem Standpunkt stehen, daß*, und idiomatische Wendungen, wie *alle Brükken hinter sich abbrechen*, haben als ganze eine lexikalische Bedeutung. Syntaktisch bleiben die Bestandteile aber selbständig, z. B. unterliegen sie den normalen Variationen der Wortstellung im Satz. Feste Syntagmen und Idiome können also nicht etwa einfach dem Lexikon entnommen werden. Erwähnt seien hier auch noch Kollokationen, wie *eingefleischter Junggeselle*. Dabei handelt es sich um Ergänzungen im Rahmen der regulären Valenz, nur daß auch die lexikalische Ausfüllung der abhängigen Elemente festgelegt ist.

Hierarchische Struktur. Lexikalische Ele-

mente, deren Bedeutungen Beziehungen stiften, eröffnen jeweils abstrakte Stellen im Satz, und zwar je nach der Wertigkeit der Relation unterschiedlich viele. Die Stellen werden in der aktuellen Äußerung von Ausdrücken eingenommen, deren innerer Aufbau wiederum darauf beruht, daß bestimmte Relatoren neue Stellen eröffnen und so fort. Es ergibt sich eine hierarchische Struktur. An einer Stelle mit bestimmter Funktion können syntaktisch unterschiedliche Konstituenten kommutieren; z. B. kann das Subjekt einer Lesung des Verbs *ärgern,* dessen konkrete Funktion es ist, das Ärgerliche zu bezeichnen, alternativ von einer Nominalphrase, einem *daß*-Satz oder einem *wie*-Satz realisiert werden: *Sein Verhalten ärgert mich. Daß er sich so verhält, ärgert mich. Wie er sich verhält, ärgert mich.* Es ist wichtig, daß der Grammatikformalismus imstande ist, den verschiedenen Konstruktionen dieselbe Funktion zuzuordnen und andererseits die Variation zu beschreiben. Zu erwähnen sind hier auch auf Charles Fillmore zurückgehende Versuche, eine inhaltliche Klassifizierung für die syntagmatischen Funktionen einzuführen: die sogenannten Tiefenkasus, wie 'Agent', 'Patient', 'Instrument' u. a.

Lineare Gliederung. Die Syntax der natürlichen Sprache erschöpft sich nicht in der hierarchischen Struktur des Satzes. Gewissermaßen quer zu ihr, schafft die lineare Gliederung eine zusätzliche Einteilung. Dazu gehört z. B. die Reichweite von Quantifizierungen und der Negation, vgl. *Jens hat zweimal ein Examen gemacht* und *Jens hat ein Examen zweimal gemacht.* Verwickelt sind die Verhältnisse bei der Koordination, wo Ausschnitte aus der hierarchischen Struktur linear miteinander verknüpft werden. *Die Eltern schicken Jens ins Zeltlager und Antje auf den Reiterhof.* Der Satz läßt sich nicht ohne weiteres als eine Hierarchie von Prädikat, Argumenten und Konjunktur repräsentieren, da der Konjunktor zwei Argumente hat, die je zwei der drei Argumente des Prädikats umfassen. Ein Problem ist auch die Reichweite der Koordination nach links und nach rechts. Ein weites Feld ist schließlich der Zusammenhang der linearen Gliederung des Satzes mit der Textstruktur.

Diskontinuierliche Konstituenten. Formale Probleme für bestimmte Grammatikansätze entstehen dadurch, daß die Elemente, die der hierarchischen Struktur nach zusammengehören, in der linearen Abfolge nicht immer zusammen stehen. Dies ist im Deutschen relativ häufig der Fall. Eine Ursache ist die Satzklammer, die z. B. das abtrennbare Präfix, das der lexikalischen Repräsentation eines Verbs zuzurechnen ist, an das andere Ende des Satzes rückt: *Er gab seine Bemühungen nicht auf.* Die freie Wortstellung erlaubt Verschiebungen von Elementen unter dem Einfluß der Thematik des Satzes, so daß diskontinuierliche Konstituenten entstehen, z. B. *Äpfel haben wir heute keine.* Diskontinuierlichkeit verursachen Einschübe in der wörtlichen Rede, z. B. in *„Heute", sagte sie, „arbeite ich nicht".* Unter dem Schlagwort 'unbounded dependencies' wird das Problem der herausgestellten Konstituenten, wie in *What do you think Gudrun said she feeds her cats?* (Kilbury 1984, 3), zur Zeit viel diskutiert (u. a. Gazdar 1981; Pereira 1981).

Fakultative Konstituenten. Über fakultative und obligatorische Konstituenten ist in der Valenztheorie viel diskutiert worden, häufig im Zusammenhang damit, was als Ergänzung (Subjekt, Objekt) und was als Angabe (Adverbialbestimmung) zu einem Verb zu gelten hat (siehe Hellwig 1978a, 122). In Wirklichkeit ist das, was gesagt wird, und das, was nicht gesagt wird, aber keine Frage der Grammatik, sondern der Kommunikationssituation. Es fällt einem nur nicht gleich eine Situation ein, in der ein Satz wie *Jens wohnt* ohne das Wo eine sinnvolle Mitteilung wäre. Für die Grammatik ergeben sich kaum Probleme, wenn man die maximal möglichen Ergänzungen und Angaben festschreibt und dazu vorsieht, daß in einer Äußerung nicht alle realisiert sein müssen. Der Grammatikformalismus sollte allerdings von der Art sein, daß letzteres nicht zu einer Vervielfachung der Regeln führt.

Ellipsen. Im Unterschied zur Auslassung fakultativer Elemente, bei der Information einfach fehlt, ist die Ellipse eine Erscheinung der Sprachökonomie. Die ausgelassenen Elemente sind aus dem Kontext oder dem Vorwissen prinzipiell rekonstruierbar. Sie müssen bei der syntaktischen Analyse auch tatsächlich rekonstruiert werden, weil davon die Interpretation der vorhandenen Elemente abhängt. Diese Rekonstruktion gehört allerdings zu den schwierigsten Problemen der Analyse natürlicher Sprachen. Ellipsen sind

typisch für die Koordination, z. B. im Satz *Die linke Seite hat nach oben, die rechte nach unten eine Öffnung.* Beim Komparativ muß der Rahmen ergänzt werden, in dem die Konstituente hinter dem *als* zu verstehen ist, vgl. *Das Geld hat ihm mehr genützt als uns. Das Geld hat ihm mehr genützt als die Empfehlung.* Der Mechanismus zur Auflösung von Ellipsen muß sicherlich die Satzgrenzen und häufig sogar den Bereich des Verbalisierten überhaupt überschreiten. Dies gilt besonders in Dialogen. *Jens meint Vanilleeis* ist z. B. eine einwandfreie Antwort auf die Frage *Was sollen wir der Antje mitbringen?* Wie erreichen wir, daß ein Computerprogramm die Antwort so interpretiert, wie den Satz *Jens meint, daß wir der Antje Vanilleeis mitbringen sollen?*

Paraphrasen. Die funktionale Dimension der syntaktischen Struktur zeigt sich im einzelnen Satz darin, daß die Elemente einerseits in unterscheidbaren Beziehungen zueinander stehen und daß andererseits Ausdrücke unterschiedlicher Art gegeneinander ausgetauscht werden können, ohne daß sich die Beziehung zu dem dominierenden Element ändert, wie wir oben am Beispiel des Subjekts zum Verb *ärgern* gezeigt haben. Ein weiterer Gesichtspunkt ist der, daß dasselbe auf verschiedene Weise ausgedrückt werden kann, daß also zwischen bestimmten Konstruktionen Paraphrasebeziehungen bestehen. Statt *Sein Verhalten ärgert mich* kann man auch sagen *Ich ärgere mich über sein Verhalten.* Generellere Paraphrasebeziehungen bestehen zwischen Aktiv- und Passivsatz oder zwischen Sätzen wie den folgenden: *Es regnete heftig. Deshalb geschahen viele Unfälle. — Weil es heftig regnete, geschahen viele Unfälle. — Wegen heftigen Regens geschahen viele Unfälle.* Es ist das Verdienst der Transformationsgrammatik, auf solche satzsemantischen Beziehungen aufmerksam gemacht zu haben.

Logische Form. Wenn man vorhat, in natürlicher Sprache einen Dialog mit einem Frage-Antwort-System zu führen, das nicht nur das Eingegebene wörtlich wieder ausgeben kann, sondern auch Schlüsse zieht, so interessieren nicht nur Paraphrasen, sondern schlechthin alle logischen Relationen zwischen den Aussagen, wie Äquivalenz, Implikation, Disjunktion, Exklusion, Kontravalenz usw. So sollte es möglich sein, aus dem Satz *Ich überredete ihn zu kommen* abzuleiten *Er wird kommen,* aus *Ich entschloß mich zu kommen* abzuleiten *Ich werde kommen* und aus *Ich weigere mich zu kommen* abzuleiten. *Ich werde nicht kommen.* Der Zusammenhang zwischen der Aussage *Fritz hat aufgehört zu rauchen* und der Präsupposition *Fritz hat früher geraucht* sollte berücksichtigt sein, ebenso das Verhältnis zwischen Quantifikation und Gewißheitsgrad in der folgenden Schlußfigur: *Alle/die meisten/manche Studenten sind nach Hause gefahren. Fritz ist Student. Fritz ist sicher/wahrscheinlich/vielleicht nach Hause gefahren.* Die Aufgabe der syntaktischen Beschreibung mündet hier ein in die Beschreibung der logischen Form der natürlichen Sprache.

Textzusammenhang. Eine Reihe von syntaktischen Erscheinungen lassen sich nur satzübergreifend behandeln, z. B. Pronomina und Ellipsen. Außerdem ist die Ermittlung des Textzusammenhanges selbst für viele Vorhaben eine notwendige Voraussetzung. Zur Kohäsion des Textes gehört, daß verschiedene Ausdrücke im Text auf dasselbe Objekt referieren, wie z. B. wenn es zunächst heißt *ein Motorrad,* dann *das Motorrad,* dann *es,* dann *die Maschine* usw. Die Kohärenz eines Textes zu erfassen, heißt herauszufinden, worum es eigentlich geht. Dazu gibt die Syntax des Satzes, insbesondere die Wortstellung, Hinweise. Beispielsweise paßt der Satz *Nachrichtensammlung ist Aufgabe des Verfassungsschutzes* in einen Kontext, in dem es darum geht, wer Nachrichten sammeln darf und wer nicht, während *Aufgabe des Verfassungsschutzes ist Nachrichtensammlung* nicht passen würde. Stichwörter in diesem Zusammenhang sind u. a. Fokus, Thema und Rhema.

Weltwissen. Man kann leicht Beispiele finden, die zeigen, daß manchmal erst Wissen über den Objektbereich ausreicht, um einer Äußerung eine eindeutige Beschreibung zuzuordnen. Ohne dieses Wissen bleibt *Die Tür fiel ins Schloß* infolge der Homonymie von *Schloß* mehrdeutig. Ebensowenig ist der syntaktische Bezug von *im Springer-Verlag* in *Gibt es ein Buch über die Kulturrevolution im Springer-Verlag?* zu entscheiden. Schließlich ist auch die Kohärenz von *Der Wagen springt nicht an. Die Zündkerzen sind verschmutzt.* nur einsichtig, wenn man über das Funktionieren von Motoren etwas weiß. Die Frage ist

nur, wo dieses Wissen im Rahmen des Systems seinen Platz hat. (Vgl. 4.6.)

3.2. Anforderungen an Grammatikformalismus

Grammatikformalismus. Im folgenden machen wir das Beschreibungsmittel zum Gegenstand der Überlegungen: (i) Man kann die grammatischen Erscheinungen einer Sprache in derselben oder in einer anderen natürlichen Sprache beschreiben. (ii) Handelt es sich um eine wissenschaftliche Beschreibung, so ist eine terminologische Normierung die Regel. (iii) Schließlich kann man zur Beschreibung ein formales System benutzen. Ein formales System besteht aus nichts anderem als einer Menge von Symbolen und einer Menge von Vorschriften, nach denen diese Symbole manipuliert werden dürfen, wobei nur die Form der Symbole ausschlaggebend ist. Ein formales System ist ein Modell, wenn für bestimmte seiner Eigenschaften Analogie zu einem Objekt postuliert wird. Da ein Computer nur zur Symbolverarbeitung im formalen Sinn fähig ist, ist die Repräsentation der Grammatik durch ein formales System zum Zwecke des Parsing unumgänglich.

Anforderungen. Es ist schwierig, zu einer natürlichen Sprache ein analoges formales System zu konstruieren, weil viele der Eigenschaften der Sprache noch unklar sind. Probleme entstehen aber auch dadurch, daß der einmal gewählte Grammatikformalismus Eigengesetzlichkeiten enthält, die mit dem, was man gerne abbilden möchte, in Konflikt stehen. Man hat oft den Eindruck, daß das Mittel dem Zweck entgegensteht. Es empfiehlt sich natürlich, von Grammatikmodellen auszugehen, die schon einige Erprobung hinter sich haben. Aber auch sie wird man prüfen müssen. Es gibt im wesentlichen drei Kriterien für einen Grammatikformalismus: (i) die Adäquatheit hinsichtlich der sprachlichen Phänomene, (ii) die Verarbeitbarkeit durch die Maschine, (iii) die Handhabbarkeit durch den Menschen.

Theorie der formalen Sprachen. Einige Maßstäbe für Grammatiken und Sprachen lassen sich aus einer mathematischen Betrachtungsweise gewinnen. Diese hat in der theoretischen Linguistik und Informatik zu einer umfassenden Theorie geführt, von der im folgenden nur einige wichtige Punkte referiert werden. Vgl. Art. 6. Man muß beachten, daß bei dieser Betrachtungsweise viele Eigenschaften einer natürlichen Sprache ausgeklammert bleiben (Chomsky 1963; Hopcroft/Ullman 1969; Gross/Lentin 1971; Aho/Ullman 1972, 83 ff.; Harrison 1978. Mit Bezug zur natürlichen Sprache Kratzer/Pause/von Stechow 1973; Klenk 1980).

Sprachen als Mengen. Ausgangspunkt ist ein Vokabular (oft auch Alphabet genannt), das definiert ist als eine endliche Menge von Symbolen. Durch Aneinanderreihen von Elementen des Vokabulars entstehen Symbolketten. Das freie Monoid über dem Vokabular besteht aus der Menge aller Ketten, die sich aus dem Vokabular bilden lassen, einschließlich der leeren Ketten, die kein Symbol enthält. Jede Teilmenge des freien Monoids über dem Vokabular ist eine (formale) Sprache. Sprachen sind also nach dieser Betrachtungsweise Mengen. Es lassen sich mit ihnen die üblichen Mengenoperationen durchführen.

Typen von Grammatiken und Sprachen. Eine geläufige Aufstellung wichtiger Grammatiktypen und Sprachen ist die sog. Chomsky-Hierarchie (Chomsky 1959). Chomsky unterscheidet

Typ 0: unbeschränkte Ersetzungssysteme,
Typ 1: kontext-sensitive Grammatiken,
Typ 2: kontex-freie Grammatiken,
Typ 3: reguläre Grammatiken.

Es ist üblich, die erzeugten Sprachen nach der jeweiligen Grammatik zu benennen.

Inklusion der Grammatiktypen. Die Grammatiktypen 0 bis 3 sind definiert durch fortschreitende Einschränkungen der Regeln. Dies bedeutet, daß die Regeln des jeweils eingeschränkteren Typs im jeweils weniger eingeschränkteren Typ ebenfalls zugelassen sind. Daraus wiederum folgt, daß die regulären Sprachen echte Teilmengen der kontextfreien, diese echte Teilmengen der kontextsensitiven und diese echte Teilmengen der aufzählbaren Sprachen sind.

Äquivalenz und Normalformen. Man unterscheidet zwischen schwacher und starker Äquivalenz: (i) Zwei Grammatiken sind schwach äquivalent, wenn sie dieselbe Sprache festlegen. (ii) Zwei Grammatiken sind stark äquivalent, wenn sie außerdem dieselben Zerlegungen der Ketten vornehmen, oder m. a. W. den Ketten dieselbe Struktur

zuordnen. Von Bedeutung ist die schwache Äquivalenz für die Konstruktion von sog. Normalform-Grammatiken, die für die maschinelle Abarbeitung von Vorteil sein können.

Adäquatheit. Auf der Suche nach einem adäquaten Grammatiktyp für eine Sprache ist man bestrebt, einen möglichst eingeschränkten Typ zu finden, denn je eingeschränkter die Grammatik ist, desto effizienter ist die Verarbeitung. Allerdings muß man zwischen schwacher und starker Adäquatheit unterscheiden: (i) Eine Grammatik bestimmten Typs ist für eine Sprache schwach adäquat, wenn sie die Menge der Ketten, aus der die Sprache besteht, erzeugen kann. (ii) Eine Grammatik ist stark adäquat, wenn sie die Ketten so strukturiert, wie es aus vorgegebenen (z. B. semantischen) Gründen erforderlich ist. Obwohl es in der Praxis auf die starke Adäquatheit ankommt, ist die schwache Adäquatheit von gewissem Interesse. Wenn ein Grammatiktyp schon nicht schwach adäquat ist, kann er natürlich erst recht nicht stark adäquat sein. Bei der Betrachtung einer solch komplexen Sprache wie einer natürlichen, ist es auch sinnvoll, die Frage nach dem formalen Typ nicht nur global, sondern für Teilmengen der sprachlichen Strukturen getrennt zu stellen. Vielleicht empfiehlt es sich, eine Gesamtgrammatik für die Sprache zu entwerfen, die aus Komponenten unterschiedlichen formalen Typs besteht.

Der formale Typ natürlicher Sprachen. (i) Reguläre Grammatiken sind möglicherweise für Teilbereiche der natürlichen Sprache, wie Morphologie und Wortbildung, stark oder schwach adäquat. Dagegen ist der Satzbau sicher nicht regulär, denn er erlaubt Einbettungen, die reguläre Grammatiken nicht beschreiben können. (ii) Mit kontext-freien Grammatiken läßt sich theoretisch ein Großteil der Sätze einer natürlichen Sprache generieren. Es scheint, daß Kongruenzphänomene, die früher als Gegenbeispiele angeführt worden sind, möglicherweise von kontext-freien Grammatiken gemeistert werden können, wenn das Hilfvokabular nicht aus einfachen sondern aus komplexen Kategorien besteht (siehe 3.3.). (iii) Eher dürfte die Überlagerung von valenzgebundenen Anhängigkeiten und linearer Gliederung, wie sie z. B. die Koordination aufweist, den Bereich des kontext-frei Machbaren übersteigen, so daß natürliche Sprachen im Prinzip wohl zu den kontext-sensitiven gehören. Man muß im übrigen das Studium mathematischer Eigenschaften von Sprachen anhand von Ersetzungssystemen trennen von den Überlegungen, welche Gestalt der Grammatikformalismus haben soll, in dem man eine Grammatik tatsächlich schreibt. Der Aufwand, der bei der unmittelbaren Verwendung eines Ersetzungssystems schon allein zur schwach adäquaten Beschreibung der freien Wortstellung, der Kongruenzen und der Valenz im Deutschen notwendig ist, wäre ungeheuer, von der starken Adäquatheit ganz zu schweigen. (Zur formalen Eigenschaft natürlicher Sprachen siehe Peters/Ritchie 1969; Pullum/Gazdar 1982, Pullum 1984a sowie das Sonderheft *Computational Linguistics* 10 [1984], No. 3—4 mit Beiträgen von Perrault, Postal/Langendoen, Pullum, Berwick.)

Der formale Typ der Grammatiknotation. Man muß sich vergegenwärtigen, daß ein Grammatikformalismus selbst eine Sprache ist. Auch Ersetzungsregeln sind nichts anderes als Ketten über einem bestimmten Vokabular, die eine bestimmte Syntax haben. Die Maßstäbe der Theorie der formalen Sprachen können daher ohne weiteres auch an die Notationen gelegt werden, in denen die verschiedenen Grammatiken geschrieben werden. Diese Betrachtungsweise ist von höchster Relevanz, wenn es darum geht, einen Grammatikformalismus für einen Parser zu entwickeln. Um die grammatischen Beschreibungen auf die Eingabesprache anwenden zu können, muß der Parser diese Beschreibungen erst einmal selbst parsen. Da der Grammatikformalismus eine Kunstsprache ist, liegt es an uns, ihn so zu konstruieren, daß er alle wünschenswerten Eigenschaften hat. Wir werden natürlich möglichst zur beschränktesten Form greifen, und darauf sehen, daß die Grammatiknotation selbst eine reguläre Sprache ist, für die es die effizientesten Erkennungsalgorithmen gibt (vgl. Art. 32).

Weitere Zielvorgaben für Grammatikformalismen. (i) Der Formalismus sollte einen solchen Grad der Generalisierung besitzen, daß beliebige Einzelsprachen darin beschrieben werden können. Auf diese Weise ist gewährleistet, daß für alle Grammatiken derselbe Parser verwendet werden kann. (ii) Gleichzeitig sollte der Formalismus so aus-

druckstark sein, daß darin die Einschränkungen, die für die jeweilige Objektsprache gelten, genau und direkt formulierbar sind. (iii) Von Vorteil wäre eine möglichst große Homogenität der Notation, so daß die Verarbeitung uniform erfolgen kann. (iv) Die Beschreibung der einzelnen sprachlichen Fakten sollte so modular sein, daß Hinzufügungen und Änderungen keine unerwünschten Nebenwirkungen auf den Rest der Grammatik haben.

Menschliche Handhabbarkeit. Der bei weitem größte Aufwand bei der Entwicklung eines einsatzfähigen natürlichsprachigen Systems ist die Beschreibung vieler Hunderte von syntaktischen Erscheinungen und vieler Tausende von lexikalischen Elementen. Den größten Teil dieser Arbeit hat der Linguist zu erledigen. Der Formalismus muß daher in erster Linie dessen Bedürfnissen gerecht werden: (i) Die Notation sollte flexibel, überschaubar, natürlich sein. (ii) Sie sollte leicht editierbar sein, damit Daten ohne viel Aufwand eingegeben und geändert werden können. (iii) Sofern die Anforderungen der Maschine und die des menschlichen Bearbeiters an die Repräsentation sich nicht decken, müssen interne und externe Ebenen der Beschreibung entworfen werden, die sich automatisch ineinander überführen lassen. (iv) Entwicklungssysteme sollten Programme zur Grammatik- und Lexikonpflege mit komfortablen Prüf-, Such-, Sortier-, und Displayfunktionen enthalten. (v) Der Parser muß so konstruiert sein, daß er bei Bedarf informative Protokolle des Analyseablaufs liefert, um Fehler in der Grammatik schnell erkennen und beseitigen zu können.

3.3. Bestimmungsstücke für Grammatiken

Grundsatzentscheidungen. In diesem Kapitel tragen wir Parameter zusammen, welche die Ausprägung von Grammatiken für natürliche Sprachen bestimmen. Es sind vor allem drei Fragen, die relativ unabhängig voneinander beantwortet werden können: (i) Wie soll die Strukturrepräsentation aussehen, die aktuellen Äußerungen zugeschrieben wird? (ii) Mittels welchen Verfahrens wird diese Struktur für alle potentiellen Fälle definiert und aktuellen Äußerungen zugeschrieben? (iii) Mit Hilfe welcher Kategorien lassen sich genau die Ausdrücke eingrenzen, die zu einer syntaktischen Einheit gehören? Die Entscheidungen für die eine oder andere Alternative hängen von Erwägungen ab, die in Auseinandersetzung mit den verschiedenen Grammatiktheorien erfolgen müssen. Es gibt allerdings gegenwärtig eine Reihe von Konvergenzen zwischen Grammatikmodellen verschiedener Herkunft. Der Trend geht zur funktionalen Orientierung bei der Strukturrepräsentation, zur lexikalischen Definition von Strukturen und zur Verwendung von komplexen Kategorien bei der Eingrenzung der wohlgeformgen Ausdrücke. (Siehe unten unter dem Stichwort Unifikationsgrammatiken.)

Strukturrepräsentation. Eine Struktur ist definiert durch eine Menge von Elementen und eine Menge von Relationen über diesen Elementen. Es ist üblich, Strukturen als Graphen darzustellen, wobei die Elemente den Knoten und die Relationen den Kanten entsprechen. Eine besondere Rolle spielen Baumgraphen. Sie repräsentieren streng hierarchische Strukturen. Hierarchien sind intellektuell einprägsam. Hierarchisch organisierte Daten sind im Computer leicht zu handhaben. Schließlich bieten syntagmatische Erscheinungen wie Valenz und Rektion auch tatsächlich ein Bild von Über- und Unterordnung. Aus diesen Gründen sind Baumgraphen (auch kurz 'Bäume' genannt) die vorherrschende Form der Strukturrepräsentation in der Syntax. Obwohl es kaum Alternativen gibt, sollte man auf der Hut sein, daß einem das 'Denken in Bäumen' nicht für manche sprachlichen Erscheinungen den Blick verstellt.

Konstituenz- oder Dependenzrelation. Die erste grundsätzliche Entscheidung, von der die Ausprägung einer Grammatik abhängt, betrifft die Relation, die zur primären Strukturierung der syntaktischen Einheiten verwendet wird. Entsprechend unterscheiden sich die zur Strukturdarstellung verwendeten Baumgraphen. Man hat die Wahl zwischen Konstituenz- und Dependenzrelation. (i) Die Elemente der Konstituentenstruktur, welche durch die Knoten im Baum repräsentiert werden, sind mehr oder weniger umfassende Segmente in der linearen Dimension der Rede; die Relation, welche durch die Kanten im Baum bezeichnet wird, ist die zwischen Ganzem und Teilen. Die Segmente resultieren daraus, daß die darin enthaltenen Ausdrücke intern in syntagmatischer Beziehung stehen und daß sie nach außen hin als Ganzes

eine syntagmatische Funktion erfüllen. Die Beziehung zwischen den Konstituenten kommt in der Hierarchie von Ganzem und Teilen allerdings nur indirekt zum Ausdruck (Wells 1947; Chomsky 1957). (ii) Die Elemente der Dependenzstruktur, welche durch die Knoten im Baum repräsentiert werden, sind die kleinsten Einheiten, die in syntagmatische Beziehungen treten; die Relation, welche durch die Kanten im Baum bezeichnet wird, ist die der syntagmatischen Beziehung selbst. Wie wir oben gesehen haben, resultieren die syntagmatischen Beziehungen aus den lexikalischen Bedeutungen der Elemente und haben eine funktionale und eine formale Seite. (Tesnière 1959; Gaifman 1965; Hays 1966b; Kunze 1975; Hellwig 1978a. Zur Abwägung zwischen Konstituenz und Dependenz siehe Hellwig 1978a, 127 ff.; Hudson 1980; Hellwig 1986.)

Komplementarität von Konstituenz und Dependenz. (i) Ist die Konstituenz zur Grundlage der Baumstruktur gemacht worden, muß die Dependenz zwischen den Konstituenten auf andere Weise dargestellt werden, z. B. durch eine Markierung der dependentiell dominierenden und den unmittelbaren Konstituenten (dazu zählt u. a. die 'x-bar'-Notation, vgl. Jackendoff 1977; Gazdar/Klein/Pullum et al. 1985, 50; Dependenzrelationen werden außerdem erfaßt durch 'gaps' und 'traces' bei Extrapositionen in der GB-Theorie, u. a. in Chomsky 1982a und 1982b, und die 'slash'-Kategorien in der GPSG, siehe Gazdar/Klein/Pullum et al. 1985, 137 ff.). (ii) Ist die Dependenz zur Grundlage gemacht worden, so ist eine implizite Konstituentenstruktur bereits mitgegeben. Jedem Knoten im Dependenzbaum entspricht nämlich ein (kleinstes) Segment in der Kette, in der Regel ein Wort. Nun bildet aber im Dependenzbaum jeder Knoten mit allen von ihm abhängigen Knoten einen Teilbaum. Jedem Teilbaum wiederum kann unmittelbar ein Segment aus der Kette zugeordnet werden, das aus den Segmenten (bzw. Wörtern) zu den Knoten besteht, die der Teilbaum enthält. Die Zerlegung des Dependenzbaumes in Teilbäume ergibt so gleichzeitig eine Gliederung der Kette in Konstituenten (Hellwig 1986, 196).

Form und Funktion. Die notwendige Korrespondenz zwischen den lexikalischen Bedeutungen, den morpho-syntaktischen Formen und den syntagmatischen Funktionen sprachlicher Einheiten kann auf zwei Weisen hergestellt werden. (i) Die formalen und die funktionalen Zusammenhänge werden als je eigene Systeme mit unterschiedlichen Strukturen dargestellt. Die Korrespondenz zwischen den Strukturen der verschiedenen Systeme wird durch Transformationen oder Abbildungen geregelt. Dieser Weg wird meist eingeschlagen, wenn die Konstituenz zur Formalisierung der Formseite benutzt wird, da bei dieser Gliederung die funktionalen Beziehungen zwischen den unmittelbaren Konstituenten nur schwer dargestellt werden können. Beispiele für die Verteilung der grammatischen Information auf verschiedene Komponenten, wenn auch in recht unterschiedlicher Weise, sind die Oberflächen- und Tiefenstruktur sowie verschiedene Subsysteme in den Weiterentwicklungen der Transformationsgrammatik (u. a. in Chomsky 1982b), die Systeme für verschiedene Sprachfunktionen in der Systemischen Grammatik (Halliday 1967), die Strata in der Stratifikationsgrammatik (Lamb 1966), die c-Strukturen und die f-Strukturen in der Lexical Functional Grammar (Kaplan/Bresnan 1982). Manche Grammatiken, z. B. die Generalized Phrase Structure Grammar (Gazdar/Klein/Pullum et al. 1985), klammern den funktionalen Aspekt auch aus der Syntax aus und weisen ihn der Semantikkomponente zu. Dabei wird häufig angenommen, daß die semantische Repräsentation, z. B. eine prädikatenlogische Formel, mithilfe von Regeln erzeugt wird, die jeweils parallel zu den Regeln angewandt werden, mit denen die syntaktische Struktur erzeugt worden ist (vgl. 4.7. und Winograd 1983, Kapitel 6). (ii) Die formalen und funktionalen Zusammenhänge werden in einem einzigen System dargestellt, dessen Elemente Form-Funktions-Einheiten sind. Die Knoten in den Strukturbäumen tragen sowohl funktionale wie formale Etiketten. Eine derartige Integration von Form- und Funktionsseite der Syntax ist möglich, wenn man die Dependenzrelation zur primären Strukturierung benutzt. Beispiele dafür sind die Lexicase-Grammar (Starosta/Nomura 1986), die Word Grammar (Hudson 1984) und die Dependentielle Unifikationsgrammatik (Hellwig 1986).

Lineare Abfolge. Die in der Theorie der formalen Sprachen betrachteten Ersetzungssysteme ergeben eine Konstituentenstruktur. Außerdem ist dort durch die Operation der

Verkettung die lineare Abfolge geregelt. Die Phrasenstrukturgrammatik stellt den Versuch dar, diese Art von Grammatik unmittelbar auf die natürliche Sprache zu übertragen. Die Phrasenstrukturbäume sind projektiv, d. h. die Reihenfolge der Knoten entspricht der Reihenfolge der entsprechenden Segmente in der linearen Dimension. Unter funktionalen Aspekten ergeben sich in der natürlichen Sprache jedoch Konstituenten, die aus nichtbenachbarten Teilen bestehen (diskontinuierliche Konstituenten). Phrasenstrukturgrammatiken sind daher für natürliche Sprachen nicht stark adäquat. Infolgedessen wird in manchen Grammatiken die Projektivität der Konstituentenstrukturbäume aufgegeben und die lineare Abfolge der Konstituenten mithilfe eines zusätzlichen Regelwerkes beschrieben, so z. B. in der GPSG. Damit bietet aber die Konstituenz als Basisrelation der Phrasenstrukturbäume verglichen mit der Konstituenz, die in Dependenzbäumen implizit mitrepräsentiert ist, keine zusätzliche Information mehr (vgl. Chomsky 1965, 124; Gazdar/Klein/Pullum et al. 1985, 44 ff.). Dependenzbäume sind freilich ebenfalls ungeeignet, um die Abfolge der Elemente durch Projektion in die Ebene zu repräsentieren. Da hier aber jedem Knoten ein unabhängig vom Baumaufbau feststehendes Segment entspricht, besteht die einfache Möglichkeit, die relative Folge der Segmente durch Positionsmerkmale an den Knoten explizit zu notieren (Hays 1966b, 117 f.; Hellwig 1986).

Strukturdefinition. Die zweite grundsätzliche Entscheidung, von der die Ausprägung einer Grammatik abhängt, betrifft die Art und Weise, wie die Struktur potentieller Einheiten definiert wird. Dies kann ja nicht durch eine Liste aller möglichen Strukturbäume geschehen, denn deren Zahl ist unendlich. Es gibt im wesentlichen zwei Vorgehensweisen: (i) Man versucht abstrakte Muster für syntaktische Einheiten zu erzeugen, in die sich das lexikalische Material einpassen läßt. (ii) Man beschreibt die Kombinationsfähigkeit der einzelnen Elemente des Vokabulars, aus denen sich komplexe Ausdrücke zusammensetzen lassen. Die erste Vorgehensweise entspricht der satzbezogenen Sicht der Sprache in der generativen Grammatik, die zweite der wortbezogenen Sicht in der traditionellen Sprachwissenschaft.

Regelgrammatiken. Die beiden genannten Alternativen erlauben eine grobe Zweiteilung in Regelgrammatiken und lexikalisierte Grammatiken (Hellwig 1978a, 64 ff.). Erstere enthalten als Kern eine Komponente, in welcher der Aufbau von Strukturbäumen generell beschrieben wird. Es kann sich um eine Menge von Ersetzungsregeln handeln, wie sie aus der Theorie der formalen Sprachen bekannt sind (vgl. Art. 32.1. und 2.), oder aus Mustern, wie sie z. B. Übergangsnetzwerken zugrundeliegen. Auch eine Menge von Prozeduren ist eine Möglichkeit (vgl. Art. 32.3.). Als zweite Komponente ist ein Lexikon vorhanden (bzw. eine Menge von lexikalischen Regeln, das zunächst nur die Verbindung schafft zwischen den von den Regeln erzeugten präterminalen Kategorien des Baumes und den terminalen Elementen. Bei dieser Zuordnung muß freilich die Valenz der terminalen Elemente berücksichtigt werden. Da auch die funktionale Interpretation der syntaktischen Strukturen von den beteiligten lexikalischen Einheiten abhängt, ist für die gegenwärtige Entwicklung der Grammatikformalismen eine zunehmende Verschiebung der Information von der Regelkomponente ins Lexikon charakteristisch. Die Entscheidung zwischen Konstituenz oder Dependenz als Strukturprinzip präjudiziert die Wahl zwischen Regelgrammatik und lexikalisierter Grammatik nicht, oder wenigstens nicht unbedingt. Die erste Formalisierung einer Dependenzgrammatik von Hays und Gaifman fußt z. B. auf Ersetzungsregeln (vgl. Gaifman 1965; Kratzer/Pause/von Stechow 1974, 137 ff.; aber auch die Kritik in Hellwig 1978a, 93).

Lexikalisierte Grammatiken. Der Aufbau einer Baumrepräsentation kann auch dadurch festgelegt sein, daß die Kategorien der terminalen Elemente Strukturinformationen enthalten. Im Gegensatz zu den Kategorien in der Regelgrammatik, sind hier die Kategorien syntaktisch transparent (siehe dazu Hausser 1986, 57 ff.), d. h. man sieht ihnen an, in welche syntaktischen Teilstrukturen die betreffenden Elemente passen. Neben dem Lexikon ist nur noch eine mehr oder weniger uniforme Operation notwendig, um aus den Teilstrukturen die Gesamtstruktur zusammenzusetzen. Auch mit dem lexikalistischen Ansatz sind sowohl Konstituenz wie Dependenz vereinbar, obwohl hier, umgekehrt wie bei der Regelgrammatik, die Dependenzstruktur die natürlichere ist. Die lexikalistische Variante der Konstituentenstruk-

turgrammatiken ist die Kategorialgrammatik. Die Kategorien, die den Lexikonelementen zugeordnet sind, haben die Struktur von Kellerspeichern. Indem fortgesetzt jeweils die obersten Symbole zweier Konstituenten gegeneinander gekürzt und entfernt werden, erreicht man die Wurzel des Strukturbaumes (vgl. Kratzer/Pause/v. Stechow 1974, 202 ff. und Art. 32.4.1.). Lexikalische Kategorien, die Dependenzstrukturen transparent machen, sind die Valenzangaben, wie sie ansatzweise schon in den Valenzwörterbüchern enthalten sind. Am naheliegendsten ist es, diese Angaben als Beschreibungen von Leerstellen zu präzisieren. Der Aufbau von Dependenzbäumen kann daher einheitlich erfolgen, indem Elemente und Teilbäume in die von anderen Elementen eröffneten Leerstellen insertiert werden (siehe Hellwig 1978b und Art. 32.4.2., vgl. außerdem McCord 1980; Hoekstra/van der Hulst/Moortgat 1980; Gross 1984; Starosta/Nomura 1986). Schließlich gibt es auch eine prozedurale Variante des lexikalistischen Ansatzes: das Word Expert Parsing. Hier ist allerdings die mangelnde Trennung zwischen Grammatik und Parser bedenklich (Rieger, Chuck 1976; Small/Rieger 1982).

Restriktionen. Eine der wesentlichen Aufgaben beim Schreiben einer Grammatik ist es, die Besetzung jeder syntagmatischen Stelle so genau einzuschränken, daß nur wohlgeformte Ketten akzeptiert werden. Zum einen ist dies aus theoretischen Gründen wünschenswert. Zum anderen führt jedes falsche Zwischenergebnis zu weiteren falschen Schritten beim Parsing, was selbst, wenn das Endergebnis in Ordnung ist, den Aufwand unnötig erhöht. Nun sind ja die Beschränkungen vielfältig, die es zu berücksichtigen gilt: zunächst vor allem die Kongruenz, dann die Wortstellung, schließlich semantische Einschränkungen. All diese Aufgaben zusammen sind nur mithilfe komplexer Kategorien und dem Mechanismus der Unifikation elegant zu lösen.

Komplexe Kategorien. Im Deutschen müssen Subjekt und finites Verb in Numerus und Person übereinstimmen. Da es zwei Numeri und drei Personen gibt, wären sechs Phrasenstrukturregeln nötig, wenn man diesen Sachverhalt mit einfachen Kategorien ausdrücken wollte. Die Regeln für die Nominalphrase müßten nach den vier Kasus differenziert werden. Weitere Kongruenzanforderungen würden für eine explosionsartige Vervielfachung der Regeln sorgen. Dies alles läßt sich vermeiden, wenn man komplexe Kategorien verwendet. Die Anfangsregel einer Phrasenstrukturgrammatik könnte etwa wie folgt aussehen:

S → NP Kasus[Nominativ] Numerus Person
+ VP Numerus Person

Eine Kategorie besteht aus einer Menge von Merkmalsspezifikationen. Jede Merkmalspezifikation besteht prinzipiell aus einem Parameter (d. i. der Merkmalstyp) und einer Menge von Werten (d. s. die Merkmale). Die Aufteilung in Parameter und Wert (in engl. Terminologie häufig 'attribute' und 'value') ist entscheidend. Sie macht es nämlich möglich, Generalisierungen mithilfe der Parameter allein auszudrücken. Die obige Regel soll besagen, daß Numerus und Person beliebige Werte haben können (was durch das Fortlassen einer Wertangabe symbolisiert ist), aber daß die Werte in der NP und der VP dieselben sein müssen. Die Parameter fungieren also wie Variablen, die in einer Formel gleich instanziiert werden müssen. Für die Notation gibt es mehrere Möglichkeiten der Vereinfachung. 'S', 'NP', 'VP' in obiger Regel sind z. B. Werte eines Parameters 'syntagmatische Konstituente', der aber fortgelassen werden kann, weil er durch die Stelle, an denen die Symbole in der Formel auftreten, zu erschließen ist. Das ändert nichts am Grundsatz, daß Kategorien Mengen von Parameter-Werte-Paaren sind.

Ein Beispiel. Ein Beispiel möge die Verwendung komplexer Kategorien mit Parametern und variablen Werten genauer zeigen. Innerhalb der Nominalphrase müssen Kasus, Numerus und Genus übereinstimmen und die Flexionsart von Artikel und Adjektiv zueinander passen: *ein hoher Preis, der hohe Preis, des hohen Preises* usw. Es sind $4 \times 2 \times 3 \times 2 = 48$ Merkmalskombinationen möglich. Beziehen wir außerdem das Faktum mit ein, daß, wenn eine Nominalphrase ein Adjektiv im Komparativ enthält, eine Präpositionalphrase mit *als* im Satz möglich ist, vgl. *Er bezahlte einen höheren Preis als ich* im Gegensatz zu * *Er bezahlte einen Preis als ich.* Die Erscheinungen in diesem Beispiel werden durch folgende Regeln erfaßt. (Es wären natürlich mehr Regeln nötig, um alle möglichen Fälle zu beschreiben.)

NP Numerus Kasus Komparativ Person[3] →
Artikel Numerus Kasus Genus Flexion
+ Adjektiv Numerus Kasus Genus Flexion Komparativ
+ Substantiv Numerus Kasus Genus

Satz Komparativ →
NP Kasus[Nominativ] Numerus Person
+ Verb Numerus Person
+ NP Kasus[Akkusativ] Komparativ

Satz Komparativ[—] →
Satz Komparativ[+]
+ PP Präposition[als]

Die Merkmale der terminalen Konstituenten werden im Lexikon festgelegt, z. B. u. a.:

einen = Artikel Numerus[Singular] Kasus[Akkusativ] Genus[Maskulinum] Flexion[typ 1]

höheren = Adjektiv Numerus[Singular] Kasus[Genitiv, Dativ, Akkusativ] Genus[Maskulinum] Flexion[typ 1] Komparativ[+]

Preis = Substantiv Numerus[Singular] Kasus[Nominativ, Dativ, Akkusativ] Genus[Maskulinum]

Man sieht hier übrigens, welche Vorteile die freie Kombinierbarkeit von Merkmalen für die lexikalische Beschreibung hat. Wird die obige NP-Regel angewendet, so daß diese terminalen Elemente die unmittelbaren Konstituenten bilden, so übertragen sich die Werte der Parameter aus dem Lexikon auf die entsprechenden Parameter in der Regel. Da alle freien Parameter in der Regel aber gleich instanziiert werden müssen, wird die Kongruenz der Merkmale zwischen den unmittelbaren Konstituenten erzwungen. Außerdem führt dies dazu, daß die Parameter Numerus, Kasus und Komparativ auch in der Kategorie NP die gleichen Werte annehmen, wie in den Kategorien der unmittelbaren Konstituenten. Im Strukturbaum entspricht dies einem Transport der Merkmale von einem Knoten zum anderen. Im Fall des Parameters Komparativ wird das Merkmal [+] oder das Merkmal [—] (letzteres ist im Lexikon der Grundstufe des Adjektives zugeordnet) durch die zweite der obigen Regeln in die Kategorie von 'Satz' übernommen. Falls ein Satz das Merkmal [+] hat, läßt er sich nach der dritten Regel mit einer Präpositionalphrase mit *als* zu einer Konstituente 'Satz' zusammenfassen, die nun das Merkmal [—] erhält, um eine erneute Anwendung der Regel zu verhindern.

Unifikation. Regeln, wie die obigen, ähneln Gesetzen in einem Logikkalkül. Ein Strukturbaum, der mithilfe solcher Regeln aufgebaut ist, entspricht einer komplexen Gleichung mit einer Reihe von Unbekannten. Das Lexikon sorgt für die Einsetzung von Konstanten bei der Auflösung einer solchen Gleichung. Es gibt aber jeweils viele verschiedene Lösungen. In Anlehnung an das Theorembeweisen in der Logik wird die Operation, mit der die verschiedenen Parameter in den Bäumen und Teilbäumen, wenn möglich, in Übereinstimmung gebracht werden, Unifikation genannt. Jede Instanziierung eines Parameters mit einem Wert setzt sich möglicherweise durch den ganzen Baum fort und kann für andere Parameter Folgen haben. Die Kategorien sind nicht statisch, sondern, je nach Kontext und Stand der Ableitung, veränderlich. Bei der Aufstellung der Regeln braucht man sich nicht darum zu kümmern, welches Merkmal vorhanden ist, sondern nur darum, in welchem Zusammenhang die Merkmale stehen, wenn sie vorkommen sollten. Das hat den Vorteil, daß man dort eine Überspezifikation vermeiden kann, wo bestimmte Merkmale keine Rolle spielen oder noch nicht eindeutig feststehen. Der Mechanismus ist auch nicht auf die morphosyntaktischen Merkmale beschränkt. Über die Vernüpfung entsprechender Parameter in den Regeln ist z. B. eine Unifikation der syntaktischen Struktur mit einer semantischen Repräsentation denkbar, so daß sich beide während des Parsings fortgesetzt gegenseitig einschränken.

Unifikationsgrammatiken. Komplexe Kategorien lassen sich mit vielen Grammatiktypen verbinden. Für Sprachen mit komplizierter Morphologie, wie das Deutsche, wird man schlechthin nicht auf komplexe Kategorien verzichten können. Wir sind hier ausführlich auf sie eingegangen, weil im Art. 32 z. T. Grammatiken mit einfachen Kategorien vorgeführt werden, um die Darstellung übersichtlich zu halten. Die meisten Verfahren würden auch mit komplexen Kategorien funktionieren. (In Chomsky 1965 werden komplexe Kategorien für das Lexikon eingeführt. In Chomsky 1970 wird erwogen, daß alle Kategorien komplex sein könnten und dadurch Transformationen überflüssig werden. Hays 1966b, 114 ff. beschreibt eine Implementierung komplexer Kategorien als Bitstrings, über denen sehr effizient Operationen ausgeführt werden können. Umfang-

reiche Grammatiken in komplexer Notation findet man schon in Brockhaus 1971 und Kratzer/Pause/von Stechow 1974. Zu Parameter-Werte-Grammatiken in jüngster Zeit siehe Karttunen 1984; Shieber 1986. Es bestehen Berührungspunkte zu den in der Informatik bekannten Attributgrammatiken, siehe dazu Knuth 1968; Wijngaarden 1969; Wijngaarden 1974; Pagan 1981). Der Mechanismus zur Verarbeitung komplexer Kategorien ist die Unifikation. Mit Hilfe der Unifikation sind Grammatiken handhabbar, die es früher nicht waren. So galt z. B. die Abhängigkeit zwischen dem Komparativ eines Adjektivs in einer NP und einer PP mit *als* im Satz, wie oben vorgeführt, als ein Beispiel, das sich von einer kontext-freien Grammatik, jedenfalls mit einigermaßen vertretbarem Aufwand, nicht definieren ließ. Die obigen Regeln beweisen das Gegenteil. Der Typ einer Parameter-Werte-Grammatik im formalen Sinn hängt letztlich von der Art der Attribute ab, welche als Parameter in den Kategorien zugelassen sind. Es steht an sich nichts im Wege, Parameter einzuführen, deren terminale Werte nicht kontext-frei zu ermitteln sind. Z. B. könnten die relativen Positionen der Wörter im Text solche Werte sein (vgl. Hays 1966b, 117 f.; Hellwig 1978a, 131 f.). Die Wortstellungsparameter, die derartige Werte annehmen können, könnten trotzdem mit demselben Unifikationsalgorithmus bearbeitet werden, wie alle übrigen. Insofern eröffnen Unifikationsgrammatiken auch formal neue Perspektiven. Es gibt Versuche, die Techniken des Theorembeweisens direkt auf Grammatiken anzuwenden. So wird die Logik-Programmiersprache PROLOG zur Implementation von Parsern für Unifikationsgrammatiken verwendet. Die Regeln der Grammatik spielen dabei die Rolle von logischen Gesetzen. Die Analyse eines Satzes gilt als Deduktionsproblem. Gerade der Unifikationsmechanismus ermöglicht es aber auch, völlig auf Regeln zu verzichten, wie es ja die Option der lexikalisierten Grammatiken ist. Komplexe Strukturen entstehen einfach dadurch, daß fortgesetzt die ursprünglich vom Lexikon gelieferten Parameter-Werte-Mengen je zweier Ausdrücke miteinander unifiziert werden, und zwar so, daß der Parameter Subkategorisierung (d. i. die Valenzangabe) in der einen Kategorie mit der Kategorie des anderen Ausdruck in Übereinstimmung gebracht wird. Dies funktioniert besonders reibungslos, wenn die Dependenzrelation zur Grundlage der Repräsentation gemacht ist. (Die Übernahme komplexer Symbole und des Unifikationsmechanismus hat zu einer Konvergenz von ursprünglich recht verschiedenen Grammatikansätzen geführt, so daß man heute von der Gruppe der Unifikationsgrammatiken spricht. Dazu zählen u. a. die Government and Binding (GB)-Theorie (Chomsky 1981a; Chomsky 1982a und 1982b), die Generalized Phrase Structure Grammar (GPSG) (Gazdar/Pullum 1984; Gazdar/Klein/Pullum et al. 1985; Evans 1985), die Lexical Functional Grammar (LFG) (Kaplan/Bresnan 1982; Horn 1983; Frey/Reyle 1983; Eisele 1985), die Definite Clause Grammar (DCG) (Colmerauer 1978; Pereira/Warren 1980; Pereira/Warren 1983), die Functional Unification Grammar (FUG) (Kay 1985a und 1985b), die Categorial Unification Grammar (CUG) (Uszkoreit 1986), die Dependency Unification Grammar (DUG) (Hellwig 1978a; Hellwig 1980; Hellwig 1986). Den Versuch, ein allgemeines Format für Unifikationsgrammatiken bereitzustellen, stellt das Program PATR-II dar (Shieber 1986). Über ein Grammatiklabor in PROLOG berichtet Gust (1984).

Obsolete Transformationsgrammatik. Die neuere Entwicklung der Grammatiktheorie hat den transformationalen Ansatz der Syntaxbeschreibung überholt, soweit er durch das Bestreben motiviert war, die Deffizienzen in der Darstellungskraft der Phrasenstrukturgrammatik zu kompensieren. Das zweite Anliegen der generativen Transformationsgrammatik, nämlich die Beziehungen zwischen Sätzen zu beschreiben, läßt sich generell der Deduktionskomponente des natürlichsprachigen Systems zurechnen. Was die Analyse betrifft, so kommt es nur darauf an, daß sie Strukturen liefert, die eine hinreichende Grundlage für Deduktionen darstellen. Versuche, Parser für die generative Transformationsgrammatik zu konstruieren, sind heute nur noch von historischem Interesse. Das Verfahren bestand aus vier Phasen: einer kontextfreien Grammatik für die Analyse der Oberfläche, die aber zu viel akzeptierte; der Anwendung der Transformationsregeln in umgekehrter Richtung; einem Parser der die transformierten, theoretisch der Basiskomponente der Grammatik zugehörigen Ergebnisse der umgekehrten Transformationen analysierte; einem Tester, der die so gewonnenen Basisstrukturen wiederum vorwärts transformierte, um zu ermitteln, ob sich die ursprüngliche Eingabe tatsächlich

generieren ließ. (Vgl. Petrick 1965; Zwicky/Friedman/Hall et al. 1965; Petrick 1973; Grishman 1976 und die historische Darstellung King 1983.)

Fehlerhafte Äußerungen. Ein Computersystem hat es mit Sprache im Gebrauch zu tun. Äußerungen sind im Sinne des Sprachsystems aber keineswegs immer fehlerfrei. In praktischen Anwendungen ist es daher erwünscht, daß der Parser auch nicht-wohlgeformte Eingaben akzeptiert. Dies führt zu einem Dilemma, da die Grammatik per definitionem genau die wohlgeformten Ausdrücke einer Sprache beschreiben muß. Wird von diesem Grundsatz abgewichen, läßt sich die Grammatik einer Sprache nicht mehr bestimmen. Wird daran festgehalten, ist das Parsing fehlerhafter Äußerungen nicht ohne weiteres möglich. (i) Man kann daran denken, die Grammatik 'liberal' zu gestalten, d. h. die Anforderungen an die Eigenschaften der wohlgeformten Ausdrücke zu verringern. Das führt aber leicht dazu, daß nun richtige Eingaben falsch oder zumindest eindeutige Eingaben als mehrdeutig beschrieben werden. (ii) Eine andere Möglichkeit ist es, die Reichweite der Grammatik um die erwarteten fehlerhaften Strukturen zu erweitern, diese aber als fehlerhaft zu markieren. (iii) Die beste Lösung wäre es sicherlich, daß der Parser Fehler durch den Vergleich mit einer korrekten Grammatik entdeckt, ohne die Eingabe als unanalysierbar zurückzuweisen. Das ist allerdings eine einschneidende Rahmenbedingung, die mit vielen der bekannten Parsingalgorithmen nicht verträglich ist. Bei den beiden letzten Lösungen wären als Ausgabe des Parsing eine Fehlerdiagnose sowie unter Umständen eine Fehlerkorrektur möglich. Dies wären sehr willkommene Leistungen. (Hayes/Mouradian 1981; Kwasny/Sondheimer 1981; Charniak 1983 sowie das Sonderheft des *American Journal of Computational Linguistics* 9 (1983).)

4. Parameter für den Parser

4.1. Technische Voraussetzungen

Datenstrukturen. Wenn man einen Parser konstruieren will, muß man über Grundkenntnisse der Informatik verfügen. Diese werden hier vorausgesetzt. Damit jedoch der Leser, der keine Erfahrung mit Computern hat, sich vom Funktionieren eines Parsers eine Vorstellung machen kann, werden im folgenden einige Datenstrukturen erklärt, die im Art. 32 eine Rolle spielen werden.

Bereich. Ein Bereich (engl. array) ist ein Speicher, den man sich als Folge von aufsteigend numerierten Zeilen vorstellen kann. (In Art. 32 stehen die Nummern in Klammern vor den Zeilen der Bereiche.) Jeder Bereich hat einen Namen, durch den er für das Programm identifizierbar ist. Der Zugriff auf die einzelnen Zeilen kann dynamisch geschehen, indem man eine Variable einführt, die Zeilennummern bezeichnet und deren Wert man berechnet. Eine Laufvariable nimmt nacheinander alle Werte, von einem Anfangs- bis zu einem Endwert, an, und ermöglicht es so, einen Bereich in systematischer Weise zu durchsuchen. Mathematisch gesehen, ist ein Bereich eine geordnete Menge von Elementen. Die Anzahl der Elemente bzw. der beschriebenen Zeilen eines Bereichs ist in der Regel veränderbar. Sie wird durch eine gesonderte Variable festgehalten, die erhöht wird, wenn eine neue Zeile geschrieben wird. In einem mehrdimensionalen Bereich bestehen Elemente wiederum aus Bereichen. Ein zweidimensionaler Bereich ist eine Matrix. Mehrere Bereiche mit der gleichen Anzahl von Zeilen können als Tabelle gedacht werden, deren Zeilen in mehrere Spalten aufgeteilt sind. Die Spalten werden durch ihren Namen identifiziert.

Stapelspeicher. Ein Stapelspeicher (auch Kellerspeicher, eng. push-down-store, stack) ist ein Speicher, in dem immer nur das zuletzt eingefügte Element zugänglich ist, das man sich als oben auf dem Stapel befindlich vorstellen kann. Um an ein früheres, und somit tiefer liegendes, Element heranzukommen, müssen erst die später abgespeicherten Elemente wieder entfernt werden. Solche Speicher werden oft als Mittel zur Ablaufkontrolle eines Algorithmus benutzt. Beispielsweise möchte man oft Informationen über bestimmte Aufgaben oder bestimmte Alternativen aufbewahren, während man zunächst andere Aufgaben erledigt. Später kehrt man zu den verbleibenden Aufgaben oder den anderen Alternativen zurück, indem man die Information darüber vom Stapelspeicher holt. Daß dies in systematischer Weise geschieht, ist durch das Prinzip der festen Ein-Ausgabe-Folge garantiert. Implementiert man einen Stapelspeicher als Bereich, so ist immer nur die zuletzt eingefügte Zeile zu-

gänglich, d. i. in der Regel die Zeile, deren Nummer mit der Anzahl der Elemente im Bereich übereinstimmt. ('Oben' ist in diesem Speicher also nicht die erste, sondern die letzte Zeile.) Früher abgespeicherte Elemente können dadurch wieder zugänglich werden, daß nacheinander die letzten Zeilen vom Stapel entfernt werden, oder daß die Variable für die Anzahl der beschriebenen Zeilen in einem Schritt auf eine frühere Zeile zurückgesetzt wird. Dadurch sind alle Elemente in den nachfolgenden Zeilen verloren. (Sie werden bei der Speicherung neuer Elemente überschrieben.)

Liste. 'Liste' ist ein terminus technicus für eine Datenstruktur, die einem Graphen mit Knoten und Kanten entspricht. Die Knoten im Graphen werden durch die Elemente der Liste repräsentiert, die Kanten durch sog. Zeiger. Jedes Element der Liste belegt einen bestimmten Speicherplatz und hat eine sog. Adresse, über die es identifizierbar ist. (Man kann Listen als Bereiche implementieren, wobei die Adressen der Listenelemente die Nummern der Zeilen sind, in denen die Elemente gespeichert werden.) Ein Listenelement enthält die Daten, welche der Beschriftung des Knoten im Graphen entsprechen. Zusätzlich enthält es die Adressen derjenigen Elemente, mit denen der entsprechende Knoten im Graphen durch Kanten verbunden ist. Dies sind die Zeiger.

Baumförmige Listen. Baumgraphen werden dazu benutzt, um hierarchische Strukturen darzustellen. In der Regel ist ein solcher Graph das Ziel des Parsings. In einem Baumgraphen sind alle Knoten miteinander durch Kanten verbunden. Siehe Abb. 31.4. Zu jedem Knoten führt höchstens eine Kante. Es gibt genau einen Knoten, zu dem keine Kante führt (die sog. Wurzel des Baumes).

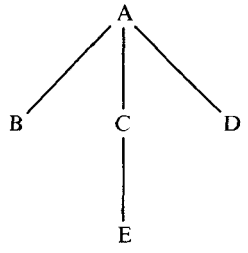

Abb. 31.4: Baumgraph

Beliebig verzweigende Baumgraphen kann man in binär verzweigende umformen und durch Listen mit nur zwei Zeigern darstellen. (Das ist wünschenswert, weil sonst der für die Zeiger benötigte Speicherplatz nicht abgeschätzt werden kann.) Siehe Abb. 31.5. Der eine Zeiger (Z_1) repräsentiert die Relation zwischen einem übergeordneten Knoten und dem ersten der untergeordneten Knoten. Der andere Zeiger (Z_2) stellt die Beziehung zwischen einem Knoten und dem nächsten gleichgeordneten Knoten her.

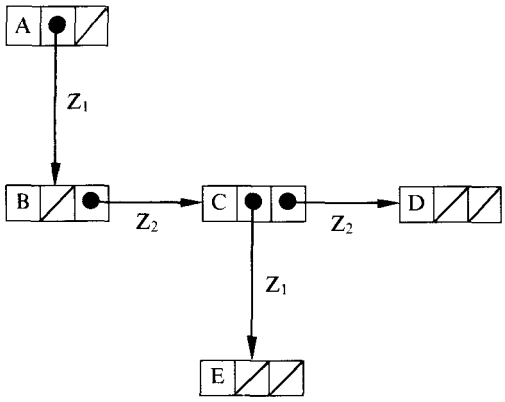

Abb. 31.5: Liste

Baumgraphen lassen sich eindeutig in Klammerausdrücke überführen, welche eine bequeme Form für die Ein- und Ausgabe darstellen. Dabei werden die einem Knoten des Graphen untergeordneten Knoten je von einem Klammerpaar umgeben, wobei diese Klammern auch all jene Knoten umfassen, die selbst wiederum von dem untergeordneten Knoten dominiert werden. Nebengeordnete Knoten werden oft auch nur von einem Klammerpaar umgeben. Die Abbildungen 31.4 und 31.5 ergeben die folgenden Klammerausdrücke

A (B) (C (E)) (D)

A (B C (E) D)

Abb. 31.6: Klammerausdrücke

Netzförmige Listen. Außer Baumgraphen sind für das Parsing netzförmige Graphen von Interesse. Für sie gilt ebenfalls die Bedingung, daß alle Knoten durch Kanten mitein-

ander verbunden sind. In der Regel gibt es auch nur einen Knoten zu dem keine Kante führt (der Eingang in das Netz). Jedoch kann zu jedem Knoten mehr als eine Kante führen und es sind auch Schleifen erlaubt, d. h. Kanten, die von einem Knoten wegführen und wieder in diesen einmünden. Netzgraphen werden in Form von Listen gespeichert, deren Zeiger entsprechend gerichtet sind.

Anmerkung zur Verwendung von Zeigern. Über Zeiger kann auf beliebige Speicherplätze Bezug genommen werden. Treten bestimmte Daten an verschiedenen Stellen auf, brauchen sie nicht kopiert zu werden, sondern es genügt, an den betreffenden Stellen einen Zeiger zu speichern, der auf den Speicherplatz gerichtet ist, an dem sich die Daten dauerhaft befinden. Zeiger können also oft die Daten selbst vertreten. Bei der Implementierung von Parsern wird man von dieser Möglichkeit weitestgehend Gebrauch machen.

Anmerkung zur Kodierung. Die Speicherung und Verarbeitung von beliebigen Zeichenketten im Computer ist aufwendig. Es empfiehlt sich daher, Daten im Computer numerisch oder als Bitketten zu repräsentieren, die nur bei der Eingabe und Ausgabe in für den Menschen lesbare Symbole übersetzt werden. Komplexe Kategorien können z. B. mit Hilfe (frei definierbarer) Schablonen auf Bitketten abgebildet werden. Für jeden Parameter wird durch die Schablone festgelegt, welcher Ausschnitt in der Bitkette ihm entsprechen soll. Die Merkmalswerte werden durch die Bits (1 bzw. 0) in diesen Ausschnitten repräsentiert. Die Unifikation von Kategorien kann dann als Boolesche Operation über Bitketten sehr effizient programmiert werden.

4.2. Bestimmungsstücke des Parsers

Aufgaben. Es gibt zwei bzw. drei Aufgaben, die ein Parser zu lösen hat. (i) Der Parser muß die von der Grammatik definierten sprachlichen Einheiten erkennen. Eine Eingabekette, die nicht zur gegebenen Sprache gehört, muß er zurückweisen (d. h. er muß eine Ausgabe erzeugen, welche die Zurückweisung meldet). Unter Umständen verlangen wir, daß der Parser im Falle der Zurückweisung eine Fehlerdiagnose ausgibt. (ii) Der Parser muß zu jeder Eingabekette, die er erkennt, die Strukturbeschreibung(en) gemäß der Grammatik erzeugen und ausgeben. (iii) Für den Fall, daß wir eine Repräsentation für die Abspeicherung der Eingabe in einer Wissensbasis wünschen, soll das Analyseprogramm die Strukturbeschreibung(en) in eine solche Repräsentation überführen. Die drei Aufgaben lassen sich trennen, so daß man einen Erkenner, einen Parser (im engeren Sinne) und einen Transduktor unterscheidet.

Analysetiefe. An das Analyseergebnis des Parsers können mehr oder weniger hohe Anforderungen gestellt werden. (i) In einem Extremfall wird lediglich verlangt, daß der Parser einen Text überfliegt (eng. skimming) und bestimmte linear abgrenzbare Einheiten entdeckt, wobei wir u. U. zulassen, daß nicht identifizierbare Teilstücke ignoriert werden (partielles Parsing). Als Muster für die gesuchten Einheiten dienen reguläre Ausdrücke. Da eine solche lineare Durchmusterung der Eingabe in Compilern für Programmiersprachen dazu verwendet wird, die Einheiten für die eigentliche syntaktische Analyse bereitzustellen, wird sie auch 'lexikalische Analyse' genannt (vgl. Aho/Sethi/Ullman 1986, 83 ff.). Für Zwecke des information retrieval ist ein solches Überfliegen von Texten u. U. schon ausreichend. (Siehe z. B. Rostek 1979 und Art. 32.3.1.1.) (ii) Im anderen Extremfall wird eine vollständige hierarchische Strukturierung der Eingabe relativ zu einer gegebenen Grammatik verlangt. Dem Folgenden legen wir diese Anforderung an den Parser zugrunde.

Parserlogik. Die Datenbestände eines Parsers sind die folgenden: (i) die zu analysierende Eingabekette, (ii) die Grammatik (einschließlich des Lexikons), (iii) die Zwischenresultate, (iv) die Strukturausgabe. Jeder dieser Datenbestände muß während des Parsingprozesses systematisch abgearbeitet werden. Die möglichen Ordnungsprinzipien für die Abarbeitung bilden eine Menge von Parservariablen. Für die Eingabe zählen dazu das Durchlaufen der Elemente in der linearen Abfolge, für die Grammatik ein bestimmter Zugriff auf Regeln oder ein Weg durch ein Netz, für die Zwischenresultate das sequentielle oder parallele Zusammenfügen von systematisch ausgewählten Teilstrukturen. Die grundsätzliche Logik eines Parsers beruht auf einer bestimmten Schachtelung der Parservariablen aller vier Bereiche. (Hays 1966a, 117 ff.; Barr/Feigenbaum 1981, 256 ff.; Slocum 1981; Winograd 1983, 363 ff.; Karttu-

nen/Zwicky 1985.) Im folgenden werden die variablen Bestimmungsstücke für die Eingabe, die Grammatik, die Zwischenergebnisse und die Strukturausgabe zusammengestellt.

4.3. Abarbeitung der Eingabe und der Grammatik

Analyseeinheiten. Die natürlichsprachigen Zeichenketten werden in den Computer eingelesen. Es sind folgende Unterteilungen des Eingabestroms möglich: (i) Die größten syntaktischen Einheiten, die jeweils durch einen Aufruf des Parsers analysiert werden sollen, sind äußerlich abgegrenzt; die Eingabe eines Textes erfolgt z. B. satzweise. (ii) Das Einlesen geschieht kontinuierlich, bis das Ende der Eingabedatei erreicht ist. Die Ermittlung der Einheiten, denen eine autonome Struktur zugeordnet werden soll, gehört mit zur Aufgabe des Parsers.

Segmentierung. Für lexikalische Einheiten (d. s. zugleich die größten morphologischen Einheiten und die kleinsten syntaktischen Einheiten) der Eingabe gibt es folgende Alternativen: (i) Die den lexikalischen Einheiten der Sprache entsprechenden Segmente sind in der Eingabe äußerlich abgegrenzt. (ii) Die lexikalischen Einheiten müssen erst im Laufe der Analyse erkannt werden. (iii) Eine dritte Möglichkeit ist die Regel: die kleinsten äußerlich abgegrenzten Segmente sind Wörter, die aber zum Teil noch weiter zerlegt werden müssen, um zu lexikalischen Grundelementen zu kommen.

Lexikonphase. In dieser Phase werden den kleinsten Segmenten in der Eingabe grammatische Informationen zugeordnet. In der Regel wird diese Aufgabe gelöst, indem die Zeichenketten in der Eingabe mit denen in einer Datei verglichen werden. Die Datei stellt das Lexikon dar und enthält zu jeder Zeichenkette die gewünschte grammatische Information. Die Zuordnung kann auf folgende Weisen geschehen: (i) Bei einem Wortformenlexikon entsprechen die Wörter in der Eingabe unmittelbar den Einträgen im Lexikon. (ii) Bei einem Stammformen- oder Grundformenlexikon werden die Wörter in der Eingabe einer morphologischen Analyse unterzogen, wobei die im Lexikon enthaltenen Wortstämme oder Grundformen und eine Menge morphologischer Regeln benutzt werden (vgl. Kay 1977). (iii) Man arbeitet mit einem reduzierten Lexikon, das nur morphologisch unanalysierbare Wörter sowie Homographen, die zu mehreren Lesungen führen, enthält. Alle übrigen Wortformen werden allein mit Hilfe der morphologischen Regeln analysiert und interpretiert. (iv) Das Lexikon hat die Form eines finiten Übergangsnetzwerks, bestehend aus Stämmen, Endungen und Derivationselementen. Die Segmente in der Eingabe werden mit Pfaden durch dieses Netzwerk verglichen. Dabei lassen sich Komposita leicht in ihre Bestandteile zerlegen. (Vgl. Hellwig 1980b, 299 ff.) Die Lexikonphase kann während des Parsingprozesses an folgenden Stellen auftreten: (i) Sie läuft vor Beginn der Analyse ab und erledigt die Aufgabe für die gesamte Eingabe. (ii) Es kommt während der Analyse jedesmal zu einer Lexikonphase, wenn der Erkenner zu einem neuen Wort vorrückt.

Das Ergebnis der Lexikonphase hat folgende Struktur: (i) Es gibt eine oder mehrere Folgen von Segmenten (mehrere Folgen, weil mehrere Zerlegungen möglich sind, wie z. B. bei *Stau-becken* versus *Staub-ecken*). (ii) Die Segmente sind über ihre Position in der Folge identifizierbar. (iii) Jedem Segment sind eine oder mehrere lexikalische Einheiten zugeordnet (mehrere, wenn das Segment mehrdeutig ist). (iv) Zu jeder lexikalischen Einheit liegt eine Beschreibung vor. Sie umfaßt ein Lexem, eine morpho-syntaktische Kategorie (in der Regel eine Wortart und eine Menge grammatischer Merkmale; ich benutze dafür den Terminus 'lexikalische Kategorie'), sowie ggf. weitere Angaben. Wenn im folgenden von 'Eingabe' gesprochen wird, ist dadurch das Ergebnis der Lexikonphase gemeint.

Abarbeitung der Eingabe. Eines der Organisationsprinzipien des Parsers ist die Abarbeitung der Elemente in der Eingabe. Es gibt folgende, z. T. kombinierbare Möglichkeiten: (i) Die Eingabe wird von links nach rechts (d. i. von Position 1 bis Position n) durchlaufen. (ii) Die Eingabe wird von rechts nach links (d. i. von Position n bis Position 1) durchlaufen. (iii) Es gibt während des gesamten Parsingprozesses nur einen einzigen Durchgang durch die Eingabe (one-pass parsing). In diesem Fall ist das Vorrücken von Position zu Position die primäre Parservariable. Satzgrenzen können überschritten und ganze Texte kontinuierlich durchlaufen werden. (iv) Es gibt wiederholte Durchgänge durch die Eingabe, weil andere

Parservariablen übergeordnet sind. (v) Die Abarbeitung der Eingabe erfolgt nicht fortlaufend, sondern z. B. von bestimmten Stellen aus nach links und rechts (Insel-Parsing). (vi) Die Reihenfolge der Elemente in der Eingabe spielt für die Logik des Parsers keine Rolle.

Abarbeitung der Grammatik. Von genereller Bedeutung für die Parserlogik ist das Verhältnis zwischen der Abarbeitung der Eingabe und der Abarbeitung der Grammatik. Es gibt folgende Möglichkeiten der Schachtelung. (i) Die Kontrolle liegt bei der Eingabe. Der Parser rückt von Element zu Element vor und sucht für die Folge von Elementen jeweils in der Grammatk nach passenden Beschreibungen. (ii) Die Kontrolle liegt bei der Grammatik. Der Parser durchläuft alle Beschreibungen und prüft für jede, ob sie auf irgendwelche Ausschnitte in der Eingabe paßt. Die Abarbeitung der Grammatik hängt im einzelnen natürlich von der Form ab, welche die Grammatik hat. Prinzipiell kann man unterscheiden: (i) die Konstruktion einer Strukturbeschreibung durch Anwendung von Ersetzungsregeln, (ii) das Erkennen der Struktur der Eingabe durch Vergleich mit Mustern und Netzwerken, (iii) den Aufbau einer Struktur durch Einsetzen von Teilstrukturen in Leerstellen (slot-filler-approach), (iv) die Ausführung von Prozeduren bei in den Parser integrierten Grammatiken.

Ersetzungsregeln. Der Grundgedanke von Ersetzungssystemen ist die Ableitung eines Ergebnisses aus einem Axiom mittels einer Folge von Operationen über Zwischenergebnissen. Gegeben sei eine Ersetzungsregel (oder 'Produktion') der Form

$$X \rightarrow Y_1 Y_2 \ldots Y_n.$$

Wir nennen X die Kategorie der Produktion und $Y_1, Y_2, \ldots, Y_n$ die (Kategorien der) unmittelbaren Konstituenten der Produktion. Ersetzungsregeln können auf folgende Weisen abgearbeitet werden. (i) Zu einer als Zwischenergebnis vorliegenden nicht-terminalen Kategorie werden Produktionen gesucht, deren Kategorie mit der vorliegenden übereinstimmt. Die Kategorien der unmittelbaren Konstituenten dieser Produktionen werden als neue Zwischenergebnisse erzeugt. Die Regel wird also von links nach rechts angewendet. Eine solche Anwendung heiße 'Expansion'. (ii) Zu einer als Zwischenergebnis vorliegenden Kategorie werden Produktionen gesucht, in denen eine der Kategorien der unmittelbaren Konstituenten mit der vorliegenden übereinstimmt. Als neue Zwischenergebnisse werden die übrigen unmittelbaren Konstituenten festgehalten sowie die Kategorie der Produktion erzeugt. Die Regel wird also von rechts nach links angewendet. Eine solche Anwendung heiße 'Reduktion'. Der Zugriff auf eine Produktion von den unmittelbaren Konstituenten her wird normalerweise weiter eingeschränkt, und zwar: (i) Die vorliegende Kategorie muß mit der ersten der unmittelbaren Konstituenten in der Produktion übereinstimmen. Diese Kategorie wird 'linker Aufhänger' ('left handle') genannt. (ii) Die vorliegende Kategorie muß mit der letzten der unmittelbaren Konstituenten in der Produktion übereinstimmen. Diese Kategorie wird 'rechter Aufhänger' genannt. Die übrigen unmittelbaren Konstituenten bilden den 'Rest' ('remainder'). Die Anwendung einer Regel kann davon abhängig gemacht werden, daß alle Kategorien im Rest der Produktion ebenfalls bereits vorliegen. Es sei noch angemerkt, daß die geforderte Übereinstimmung der Kategorien bei Grammatiken mit komplexen Symbolen nicht Identität, sondern Unifizierbarkeit bedeutet. Zu den Vorkehrungen, die getroffen werden müssen, wenn mehrere Produktionen auf eine Kategorie anwendbar sind, siehe unten. (Die Abarbeitung von Ersetzungsregeln ist das übliche Verfahren des auf der Theorie formaler Sprachen basierenden Parsings in der Informatik. Siehe Aho/Ullman 1972; Mayer 1978. Vgl. 32.2. und 3.).

Muster und Übergangsnetzwerke. Die Grammatik kann die Form von Mustern haben, die in der linearen Abfolge der Eingabe Entsprechungen finden. Muster für verschiedene Ausschnitte können zu einem Netzwerk kombiniert werden. Der Grundgedanke der Abarbeitung von Mustern und Netzwerken ist der des prüfenden Fortschreitens vom Anfang einer Kette bis zu deren Ende. Eine Grammatik in Form eines einfachen oder rekursiven Übergangsnetzwerkes ist ein Graph, dessen Kanten mit Kategorien beschriftet sind. Im Prinzip wird parallel mit dem Vorrücken in der Eingabe ein Pfad durch das Netz gesucht, sodaß die Kategorien an den begangenen Kanten mit den Kategorien der Einheiten in der Eingabe übereinstimmen. Wiederholt auftretende Teilstrukturen lassen sich abdecken, indem als Bedingung für das Passieren von Kanten das Durchlaufen eines

separaten Unternetzwerkes für die entsprechende Teilstruktur gefordert wird, wobei dieses Netzwerk auch mit dem gerade in Abarbeitung befindlichen oder einem früher begonnenen Netz identisch sein kann (rekursive Netzwerke). Zur Kontrolle der Schachtelung von Netzen verwendet man Stapelspeicher. Zur Abarbeitung alternativer Kanten siehe unten. In erweiterten Übergangsnetzwerken sind an den Kanten zusätzlich Prozeduren angegeben, die beim Durchlaufen des Netzes aufgerufen werden. (Zur Einführung in die Erkennung sprachlicher Muster siehe Winograd 1983, 35 ff; zu mehr oder weniger oberflächennahen Mustern Weizenbaum 1966; Rostek 1979; Wilks 1983; Brodda 1983; zu erweiterten Übergangsnetzwerken Bobrow/Fraser 1969; Woods 1970 (deutsch 1976); Bates 1978; Pinkal 1980; Johnson 1983b; Winograd 1983, 195 ff. und Art. 27.)

Slot-filler-Prinzip. Der Grundgedanke des Slot-filler-Verfahrens ist der, daß der komplexe Strukturbaum, der das Ergebnis der Analyse darstellt, dadurch erzeugt werden kann, daß Elemente oder Teilbäume rekursiv in andere Teilbäume eingepaßt werden. Letztere enthalten Leerstellen (slots), d. s. Variablen, deren Relation zu den übrigen Elementen des Teilbaumes schon feststeht, für die aber erst noch konkrete Elemente (filler) gefunden werden müssen. Die kategorialen Anforderungen an diese Elemente werden in den Leerstellen spezifiziert. Auf diese Weise läßt sich die Kombinationsfähigkeit der syntaktischen Einheiten beschreiben. Für die Abarbeitung von Strukturbeschreibungen mit Hilfe von Leerstellen gibt es zwei Möglichkeiten. (i) Zu allen Teilbäumen mit Leerstellen werden unter den übrigen Teilbäumen passende Besetzungen gesucht. (ii) Jeder Teilbaum sucht unter den übrigen Teilbäumen nach einer Leerstelle, für deren Besetzung er geeignet ist. Da das Slot-filler-Prinzip nichts anderes als eine Formalisierung der Kombinationsfähigkeit der sprachlichen Einheiten ist, eignet es sich besonders für dependenzbasierte und lexikalisierte Grammatiken. (Glaserfeld/Pisani 1970; Lindsay 1971; Hellwig 1978b; Hellwig 1980b; McCord 1980; Nelimarkka/Jäppinen/Lehtola 1984; Starosta/Nomura 1986.)

Prozeduren. In den bisher besprochenen Fällen erfolgt die Ausarbeitung der Grammatik auf uniforme Weise. Der Austausch einer Grammatik gegen eine andere ändert nichts am Ablauf. Die Initiative für die jeweils nächsten Schritte liegt beim Parser. Dagegen besteht der Grundgedanke einer in Form von Prozeduren geschriebenen Grammatik darin, daß die konkreten, einzelsprachigen Erscheinungen selbst den Ablauf des Parsers steuern sollen. Die Kontrolle über die vorzunehmenden Schritte liegt hier also bei der Grammatik, und zwar nicht nur formal, sondern inhaltlich. Der Algorithmus wird dadurch heterogen und ist nicht auf andere Sprachen übertragbar. Zu den ältesten Parsern seiner Art gehört das Saarbrücker Analysesystem, in dem mit jeweils spezifischen Methoden Wortklassen, Homographen, Subgruppen, nominale, verbale und sonstige Gruppen, Subsätze und Sätze des Deutschen ermittelt wurden. (Vgl. Eggers/Dietrich/Klein et al. 1969; Dietrich/Klein 1974, 92 ff.) Ein extremes Beispiel heterogener Parser aus der Gegenwart ist das Word Expert Parsing. Ihm liegt die Idee des sog. objektorientierten Programmierens zugrunde, bei dem die Module des Programms selbst aktiv an der Bestimmung der nächsten Schritte teilnehmen (vgl. Stoyan/Wedekind 1983). Im Fall des Word Expert Parsings entspricht jedem Wort der Eingabesprache ein eigenes Modul. (Small/Rieger 1982; Small 1983; Cottrell/Small 1984; Hahn 1984; Phillips 1984; Reddig 1984; Eimermacher 1986; Mehl 1986.)

Steigerung der Effizienz. Um die Abarbeitung der Grammatik effizienter zu machen, kann man mehrere Maßnahmen ergreifen. Die folgenden Maßnahmen sind für Grammatiken in Regelform formuliert. Sie gelten aber analog auch für die anderen Repräsentationsformen. (i) Man sorgt für einen möglichst schnellen Zugriff auf die Regeln. (ii) Man wählt ein möglichst einfaches Regelformat, so daß der Vergleich von Regeln und Zwischenergebnissen unkompliziert ist. (iii) Man versucht möglichst solche Regeln erst gar nicht anzuwenden, die sich später als Sackgassen herausstellen. (iv) Man versucht, dieselbe Aufgabe möglichst nicht mehr als einmal lösen zu müssen.

Zugriff auf die Regeln. Ein Mittel zum schnellen Zugriff sind Verzeichnisse, in denen zu jeder Kategorie die Regeln aufgeführt sind, in denen diese Kategorie vorkommt, z. B. als Kategorie der Produktion, als linker oder als rechter Aufhänger. (Sind die Kategorien intern numerisch kodiert, so kann man diese Numerierung zugleich als Adressen in

den Verzeichnissen verwenden. Die Verzeichnisse wiederum enthalten nichts anderes als Zeiger auf die Regeln. Damit kann der Zugriff auf die Gramatik so gut wie direkt erfolgen.)

Normalformgrammatik. Um ein möglichst einfaches Regelformat zu erhalten, macht man sich die Äquivalenzbeziehungen zwischen Grammatiken zunutze. So lassen sich z. B. alle kontext-freien Grammatiken automatisch in eine schwach äquivalente sog. Chomsky-Normalform überführen, deren Ersetzungsregeln die Form

$$A \rightarrow B\,C$$

oder $A \rightarrow a$

haben, wobei A, B, C einzelne nicht-terminale Symbole sind und a ein einzelnes terminales Symbol ist. Es kommen in dieser Grammatik also nur binäre Zerlegungen nicht-terminaler Ketten und lexikalische Einsetzungen vor, die sich leicht abarbeiten lassen. (Umformungs-Algorithmus siehe Aho/Ullman 1972, 151 ff.)

Prädiktive Analyse. Sackgassen kann man teilweise mit Hilfe einer prädiktiven Grammatik (auch Greibach-Normalform) vermeiden. In eine solche lassen sich ebenfalls alle kontext-freien Grammatiken schwach äquivalent übersetzen. Die Umformung ist reversibel. Die Ersetzungsregeln haben hier die Form

$$A \rightarrow a\,\alpha,$$

wobei A ein einzelnes nicht-terminales Symbol, und α eine Kette aus nicht-terminalen Symbolen ist. In dieser Grammatik ist also der linke Aufhänger immer eine lexikalische Kategorie. Eine Produktion wird nur angewendet, wenn diese Kategorie mit der Kategorie des nächsten Elements in der Eingabe übereinstimmt. Statt die Grammatik selbst umzuformen, läßt sich auch eine Matrix erzeugen, der man entnehmen kann, ob und welche Regel von einer gegebenen Kategorie zu einer gegebenen lexikalischen Kategorie als linker Aufhänger führt. (Ist dies genau eine, liegt eine sog. LR(k) — 'left-to-right and right most derivation' — Grammatik mit $k = 1$ vor, für die eine deterministische Analyse mit linearem Zeitbedarf möglich ist; vgl. Mayer 1978, 264 ff.) Eher simple prädiktive Vorkehrungen sind die Bedingung, daß eine Produktion keine lexikalische Kategorie enthalten darf, die in der Eingabe gar nicht vorkommt, und die Bedingung, daß die Produktion nicht mehr unmittelbare Konstituenten enthalten darf, als bis zum Ende der Eingabe noch Elemente vorhanden sind (shaper test). (Umformungs-Algorithmen siehe Aho/Ullman 1972, 153 ff. Zur prädiktiven Analyse siehe Kuno/Oettinger 1963; Greibach 1964; Kuno 1965b; Kuno 1976; unter psychologischen Gesichtspunkten DeJong 1979a.)

Faktorisierung der Regeln. Dieselbe Aufgabe wird u. U. mehrmals zu lösen sein, wenn alternative Regeln teilweise dieselben Konstituenten enthalten, z. B. die folgenden:

$$A \rightarrow B$$
$$A \rightarrow B\,C.$$

Wurde zunächst die erste Regel gewählt und war die Abarbeitung von B auch erfolgreich, stellt sich aber schließlich doch heraus, daß die Regel nicht paßt (etwa, weil eine Konstituente C in der Eingabe folgt), so wird nach nunmehriger Benutzung der zweiten Regel die Konstituente B wiederum geprüft, was wohlmöglich mit der Anwendung zahlreicher anderer Regeln verbunden ist. Der Ausweg ist die Umformung der Grammatik in eine faktorisierte Normalform. Für diese ist festgelegt, daß alternative Produktionen nicht mit derselben unmittelbaren Konstituente beginnen dürfen. Für die obigen beiden Regeln ergibt dies:

$$A \rightarrow B\,X$$
$$X \rightarrow C$$
$$X \rightarrow \varepsilon$$

'ε' steht für die leere Kette. Die letzte Regel bewirkt also die Tilgung der Konstituente X. Tilgungen sind allerdings nicht mit allen Parsingtechniken verträglich. Recht gut läßt sich hingegen eine Grammatik als Übergangsnetzwerk faktorisieren.

4.4. Erzeugung und Verwaltung von Ergebnissen

Analysestrategien. Aufgabe des Parsers ist es, eine Strukturrepräsentation zu erzeugen. Diese Aufgabe kann nicht auf einmal gelöst werden, sondern nur über eine Reihe von Schritten, die jeweils Teilergebnisse erbringen. Jeder Schritt muß vom bis dahin Bekannten ausgehen; das Vorgehen muß aber vor allem zielorientiert sein, so daß mit Sicherheit die Information hinzugewonnen wird, die zum Endergebnis führt. Normalerweise ist die erwünschte Strukturrepräsentation ein Baumgraph. Gehen wir davon aus, daß es sich um einen Phrasenstrukturbaum handeln soll. Damit ist in der Regel die Wurzel des Baumes, der gesucht wird, bekannt:

sie ist mit dem Startsymbol bzw. dem Axiom der Grammatik identisch. Weiter sind die (möglichen) terminalen Konstituenten des Baumes bekannt, soweit sie während der Lexikonphase ermittelt wurden. Die Konstituenten, die zwischen Wurzel und terminalen Konstituenten liegen, sind dagegen unbekannt. Wir haben also folgende Ausgangslage:

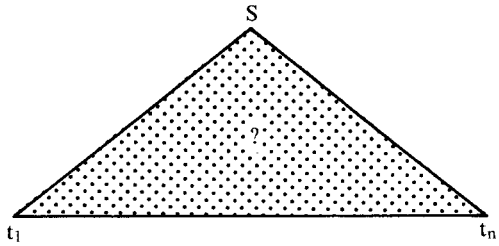

Abb. 31.7: Gesuchter Strukturbaum

'S' ist das Startsymbol, 't_1 — t_n' stellen die in der Lexikonphase ermittelten Kategorien dar. Der gepünktelte Bereich bildet den Teil des Baumes, den es schrittweise zu rekonstruieren gilt: Das einzige, worauf sich der Parser dabei stützen kann, sind die Regeln der Grammatik. Die Regeln definieren Relationen zwischen syntaktischen Einheiten; das heißt aber: sie legen mögliche Ausschnitte von Strukturbäumen fest. Mit ihrer Hilfe können daher von den Eckpunkten der bekannten Kategorien aus, die Kategorien im fraglichen Bereich erschlossen und mit den bekannten verknüpft werden. Die folgenden Strategien unterscheiden sich dadurch, in welcher Reihenfolge dabei vom Bekannten zum noch Unbekannten vorgegangen wird. (Siehe zum folgenden ausführlich Mayer 1978, 79 ff.; De Roeck 1983.)

Top-down Analyse. Beim Vorgehen von oben nach unten (top-down) wird an die Wurzel des Baumes angeknüpft und durch fortgesetzte Expansion der Kategorien versucht, schließlich die Folge der lexikalischen Kategorien zu erreichen. Die Produktionen werden also von links nach rechts benutzt. Ein Zwischenstadium einer top-down Analyse (nachdem eine Produktion S → NP VP benutzt worden ist) illustriert der folgende Baum:

Der schraffierte Bereich ist der in diesem Stadium bearbeitete Teil des Baumes, der gepünktelte Bereich ist der noch zu bearbeitende.

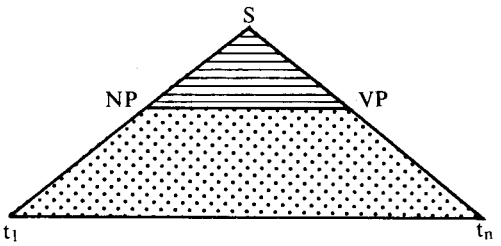

Abb. 31.8: Top-down Analyse

Bottom-up Analyse. Beim Vorgehen von unten nach oben (bottom-up) bilden die lexikalischen Kategorien die Ausgangspunkte. Es werden Produktionen gesucht, nach denen sich Folgen von ihnen zu einer übergeordneten Kategorie zusammenfassen lassen. Die Produktionen werden also von rechts nach links benutzt. Der Prozeß der Reduktion wird fortgesetzt, bis schließlich die Wurzel des Baumes erreicht ist und das Innere des Baumes vollständig ausgefüllt ist. Ein Zwischenstadium einer bottom-up Analyse (nachdem eine Produktion NP → Det N auf die terminalen Einheiten Det und N angewendet worden ist) illustriert der folgende Baum:

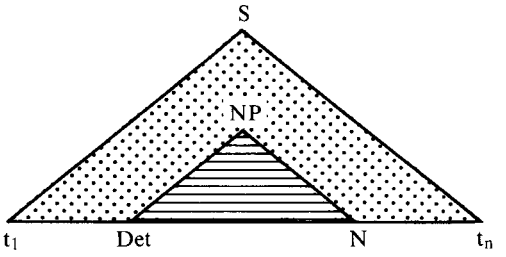

Abb. 31.9: Bottom-up Analyse

Reihenfolge der Ableitungsschritte. Auf dem Weg von der Wurzel des Baumes zu den Endelementen oder umgekehrt werden fortgesetzt Produktionen angewendet. Für die Reihenfolge dieser Anwendungen gibt es folgende Alternativen: (i) Es wird immer die am weitesten links (leftmost) stehende, noch nicht bearbeitete Kategorie als nächste expandiert bzw. reduziert. Analog kann man festlegen, daß immer die nächste am weitesten rechts stehende (rightmost) Kategorie bearbeitet wird. (ii) Die Kategorien werden in der Reihenfolge bearbeitet, wie sie erzeugt worden sind oder ursprünglich vorlagen. Die erste Alternative führt zu einer Analyse 'Tiefe zuerst', die zweite zu einer Analyse 'Breite zuerst'.

Tiefe zuerst. Das Prinzip der Linksableitung (und analog der Rechtsableitung) führt dazu, daß bei der top-down Analyse auf der linken (rechten) Seite des Baumes zunächst immer tiefere Stufen in der Hierarchie (depth-first) betreten werden, bis schließlich die Ebene der terminalen Symbole erreicht ist. Eine solche Organisation ist sinnvoll, weil die terminalen Kategorien ja bekannt sind und ihre Prüfung daher die bisherige Ableitung frühzeitig verifizieren oder falsifizieren kann. Im Falle der Falsifikation braucht der jeweilige Rest der von den Produktionen erzeugten Kategorien gar nicht weiter bearbeitet werden. Im Falle der Verifikation ist damit der Linkskontext für die weiteren Expansionen bekannt. Der folgende Baum illustriert einen Zustand bei einer top-down leftmost depth-first Analyse:

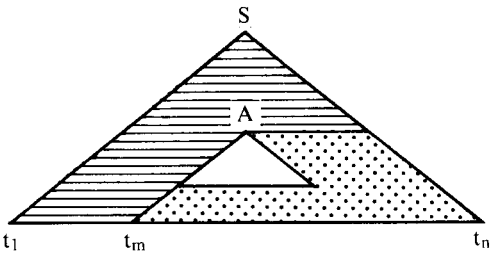

Abb. 31.10: Tiefe zuerst

Der Linkskontext zu der Konstituente A ist, infolge der Abarbeitung des Baumes von den terminalen Kategorien t_1 bis t_m, verifiziert. Insofern ist sicher, daß sich die Expansion von A als nächster Schritt lohnt. Eine bottom-up Variante nach dem Prinzip Tiefe-zuerst ist das sog. left-corner Parsing. Dabei wird, ausgehend vom ersten terminalen Element in der Eingabe, jeweils nach einer Kategorie gesucht, deren erste unmittelbare Konstituente die gerade erkannte Kategorie ist, d. h. für welche top-down Analyse die aktuelle, bottom-up gewonnene Kategorie Ziel der Linksableitung sein könnte. Dies sind die Kategorien solcher Produktionen, deren linker Aufhänger die aktuelle Kategorie ist. Werden solche Produktionen gefunden, so wird jeweils die aktuelle Kategorie auf die Kategorie der Produktion hin reduziert (wobei u. U. ein Rest bleibt), bis die Wurzel des Baumes erreicht ist (vgl. Mayer 1978, 87; Aho/Ullman 1972, 310; Kimball 1975).

Breite zuerst. Die Abarbeitung von Kategorien in der Reihenfolge ihres Entstehens, d. h. die früher erzeugten zuerst, die später erzeugten später, führt dazu, daß der Strukturbaum immer auf ganzer Breite (breadth-first) ausgefüllt wird. Es wird von Hierarchieebene zu Hierarchieebene fortgeschritten, wobei immer zuerst alles getan wird, was sich auf einer Ebene tun läßt. Dies ist ein sinnvolles Organisationsprinzip für die bottom-up Analyse, denn zwei unmittelbare Konstituenten, die zu einer übergeordneten zusammengefaßt werden sollen, müssen ja auf derselben Ebene stehen und in Bezug auf alle tieferen Ebenen komplett sein. Ein Stadium einer solchen Breite-zuerst Analyse von unten nach oben illustriert der folgende Baum:

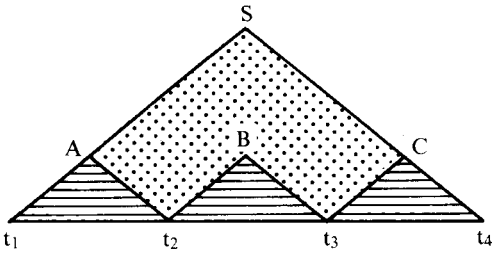

Abb. 31.11: Breite zuerst

Datengesteuert versus erwartungsgesteuert. Der Vorteil der bottom-up Analyse liegt darin, daß an die sprachlichen Einheiten angeknüpft wird, die tatsächlich in der Eingabe vorhanden sind. Deren Merkmale, einschließlich ihrer syntaktischen Verbindbarkeit (Valenz, Subkategorisierung) sind aufgrund der vorgeschalteten Lexikonphase bekannt. Infolgedessen können die Analyseschritte auf genau die Regeln beschränkt werden, die mit dem vorhandenen Material kompatibel sind. Dagegen besteht bei der top-down Analyse die Gefahr, daß viele Ableitungen erzeugt werden, die mit der terminalen Kette überhaupt nicht zusammenpassen. In mancher Hinsicht ist eine top-down Analyse der bottom-up Analyse aber auch überlegen. Z. B. ermittelt letztere zum Satz (ohne Großschreibung) *ich lege die rolle auf den tisch* als Zwischenergebnis u. a. einen Imperativsatz *rolle auf den tisch.* Beim top-down Parsing mit Linkskontext würde eine solche Konstituente, die keinen Platz im Ganzen hat, nicht abgeleitet. Die Analyse von oben her trifft den Sachverhalt besser, daß der menschliche Sprachbenutzer beim Lesen der Wörter Erwartungen aufbaut, die darauf beruhen, daß er schon eine Vorstellung vom

Ganzen hat. Aus der Erkenntnis heraus, daß beide Verfahren für sich genommen unzureichend sind, sind Parsingverfahren von Interesse, die auf irgendeine Weise bottom-up und top-down Informationen verbinden (z. B. die oben erwähnte prädiktive Analyse oder die im folgenden skizzierten Chart-Parser).

Regierend versus abhängig. Die verschiedenen Strategien zur Erschließung der Baumstruktur gelten auch für Dependenzgraphen, nur bedeuten sie hier linguistisch etwas anderes als bei Phrasenstrukturgraphen. Während bei letzteren eine top-down Analyse einem Vorgehen von den umfassenderen zu den kleineren Einheiten, und eine bottom-up Analyse ein Vorgehen von den kleineren zu den umfassenderen darstellt, bedeutet im Dependenzgraphen die Analyse von oben nach unten, daß man von den regierenden Elementen zu den abhängigen übergeht, und die Analyse von unten nach oben, daß zu den abhängigen Elementen die regierenden gesucht werden. Das syntaktische Verhältnis zwischen regierenden und abhängigen Elementen kann man sich so denken, daß erstere Leerstellen (slots) eröffnen, zu denen letztere die Ausfüller (filler) sind. Bei einer top-down Strategie sind demnach Leerstellen die Ausgangspunkte, für die jeweils Teilbäume gesucht werden, die sie ausfüllen können. Dies kann man als erwartungsgesteuertes Vorgehen bezeichnen. Da es viel mehr mögliche Ergänzungen geben wird, als tatsächlich Ausfüller im Satz vorhanden sind, wird bei diesem Verfahren vieles vergeblich gesucht. Bei der bottom-up Strategie liegt die Initiative dagegen bei den Ausfüllern, die gezielt nach passenden Leerstellen suchen. Dies hat den Anschein einer eher datengesteuerten Kontrolle. Nun ist allerdings die Strukturerschließung eines Dependenzbaumes überhaupt datengesteuert, da ja alle Einheiten, seien sie oben oder unten im Baum, lexikalische sind.

Verwaltung der Ergebnisse. Die Ergebnisse können auf folgende Weise verwaltet werden: (i) Es ist immer nur das aktuelle Gesamtergebnis gespeichert, nämlich der im Entstehen begriffene Strukturbaum bzw. die ihn betreffende Information. Diese eine Datenstruktur wird im Laufe des Parsings zum Endergebnis ausgebaut. Jede während eines Parserschrittes gewonnene Information geht in sie ein. Es kann passieren, daß Teile der aufgebauten Struktur revidiert werden müssen und daß dann beim Neuaufbau schon einmal geleistete Arbeit wiederholt werden muß. (ii) Alle Zwischenergebnisse bzw. Teilbäume werden in einer Tabelle gespeichert (well-formed substring table). Es kann jederzeit auf sie zurückgegriffen werden. Ein in einem Stadium erstelltes Zwischenergebnis kann in anderem Zusammenhang wieder verwendet werden. Bevor eine Teilaufgabe vom Parser angegangen wird, wird in der Tabelle nachgesehen, ob es evtl. schon eine Lösung gibt, die fertig übernommen werden kann.

Chart ist der heute oft gebrauchte Terminus für eine zentrale Datenstruktur, in der alle Teilergebnisse des Parsers zusammengetragen werden. Als visuelle Vorstellung einer Chart wird oft ein Graph aus Knoten (vertices) und Bögen (edges) angeführt. Dabei markieren die Knoten die Grenzen zwischen den Konstituenten (sozusagen die Zwischenräume). Die Bögen überbrücken die Spanne zwischen dem Anfang und dem Ende der im Laufe des Parsing ermittelten Konstituenten. Die Knoten sind numeriert, die Bögen mit den Kategorien der Konstituenten beschriftet. In dieser Gestalt ist eine Chart gewissermaßen ein Übergangsnetzwerk für Ergebnisse. Die endgültige Chart zum mehrdeutigen Satz *ich möchte gern rumkugeln* kann man sich z. B. so denken:

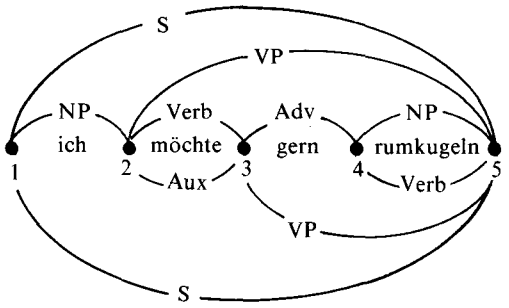

Abb. 31.12: Chart

Die Chart ermöglicht es, daß die nichtmehrdeutigen Konstituenten NP: *ich* und Adv: *gern* in beide Sätze eingebaut werden können, aber nur einmal analysiert werden müssen. Statt der Kategorien an den Bögen enthält eine verallgemeinerte Chart Listen oder komplexe Merkmalsmengen, die die Struktur der entsprechenden Konstituenten beschreiben. Eine Chart ist somit letztlich eine Abbildung alternativer und sich überlappender Strukturbäume auf die Eingabekette. (Kaplan 1973; Kay 1973; Dostert/

Thompson 1976; Kay 1977; Colmerauer 1978; Bear/Karttunen 1979; Brecht 1979; Charniak 1983; Church 1983; Thompson/Ritchie 1984; Varile 1983; Luckhardt 1985; Görz/Beckstein 1986.)

Agenda. Es gibt Parser, deren Logik auf der uniformen Abarbeitung von Zwischenergebnis-Tabellen beruht. (Die bekanntesten sind der Cocke-Algorithmus und der Earley-Algorithmus; siehe 32.2.) Andererseits eröffnet eine Datenstruktur, in der die verschiedenen Ergebnisse zusammengetragen werden, gerade die Chance, verschiedene Strategien zu kombinieren und heterogene Kontrollstrukturen zu erproben, insbesondere eine Verbindung von top-down und bottom-up Analyse sowie ein Vorrücken in der Eingabe sowohl nach rechts wie nach links. Diese Flexibilität ist das Motiv für manche Chart-Parser. Die Aktionen, die auszuführen sind, werden zunächst in einem Agenda genannten Speicher eingetragen (Kay 1977). Solche Eintragungen können aus vielen Quellen stammen, u. a. auch aus Abarbeitungen der Chart. Von der Agenda werden die Aktionen dann abgerufen, wobei die Auswahl und Reihenfolge ebenfalls heterarchisch geregelt sein kann.

4.5. Verfahren bei Alternativen

Ursachen für Alternativen. Alternativen auf verschiedenen Ebenen sind für die natürliche Sprache geradezu konstitutiv. Sie sind die Voraussetzung dafür, daß mit beschränkten Mitteln unbeschränkt viel ausgesagt werden kann. Trotzdem macht es dem menschlichen Hörer/Leser offensichtlich keine Mühe, Äußerungen im linearen Fortschreiten des Hörens/Lesens eindeutige Lesungen zuzuordnen. Alternativen kommen ihm dabei kaum zu Bewußtsein. Diesen Prozeß nachzubilden, bleibt eine Aufgabe für zukünftige Parser. (Vgl. dazu Marcus, M. P. 1980 und die über seinen 'deterministischen' Parser geführte Diskussion, Charniak 1983; Sabah/Rady 1983; Sampson 1983; Thompson/Ritchie 1984; Thiel 1986). Ein Automat, für den in jedem Zustand feststeht, welcher Schritt als nächstes erfolgen muß, heißt deterministisch. Nicht-deterministische Zustände eines Parsers für natürliche Sprachen können folgende Ursachen haben: (i) Die grammatische Beschreibung ist zu grob. (ii) Es ist zu wenig Kontext oder zu wenig Information aus anderen Bereichen (Semantik, Pragmatik) verfügbar. Dies liegt bei Zwischenergebnissen in der Natur der Sache. (iii) Es liegt wirkliche, z. B. gewollte, Mehrdeutigkeit vor. Aus letzterer Möglichkeit folgt, daß ein Parser für natürliche Sprachen im Prinzip nicht vollständig deterministisch sein kann. Im folgenden sind Maßnahmen zusammengestellt, die zur Lösung der Probleme getroffen werden können.

Deterministische Abarbeitung. Ein Computerprogramm ist immer deterministisch. Nicht-deterministische Zustände können jedoch deterministisch abgearbeitet werden. Es gibt dazu zwei Möglichkeiten: (i) Die Alternativen werden nacheinander abgearbeitet. Eine Alternative wird so weit wie möglich verfolgt. Anschließend wird der Parser in den alten Zustand zurückversetzt (backtracking) und die nächste Alternative wird verfolgt. Dies wird fortgesetzt, bis keine Alternative mehr unbearbeitet ist. (ii) Alle Alternativen werden gleichzeitig abgearbeitet (Parallelverarbeitung). Das heißt, daß die Kontrolle verzweigt und der Automat Mengen von Zuständen hat, bzw. daß eine Menge von Automaten an demselben Problem arbeiten. Soweit dazu nicht eine Hardware mit Parallelprozessoren zur Verfügung steht, werden die parallelen Pfade in Wirklichkeit nacheinander beschritten.

Backtracking. Beim Abarbeiten von Alternativen mit Rücksetzen sind zwei Aufgaben zu lösen: (i) über die aufgetretenen Alternativen muß buchgeführt werden. Die Reihenfolge der Bearbeitung muß geregelt werden. (ii) Jeder Zustand, in dem eine Alternative gewählt wird, muß vollständig rekonstruierbar sein, um ggf. eine andere Wahl treffen zu können. Die klassische Lösung der beiden Aufgaben ist die folgende (vgl. Aho/Ullman 1972, 282 ff.; Winograd 1983, 99 ff.): Innerhalb eines Zustandes werden die Alternativen in der Reihenfolge ihrer Auflistung ausgewählt (also z. B. von mehreren alternativen Regeln in der Grammatik zuerst die erste, dann die zweite usw.). Treten in aufeinander folgenden Zuständen Alternativen auf, so wird immer zuerst zum jüngsten Zustand zurückgesetzt. Erst wenn dessen Alternativen sämtlich geprüft worden sind, wird der vorangehende Zustand rekonstruiert und den dortigen Alternativen nachgegangen, und so fort, bis schließlich wieder der Zustand erreicht ist, in dem die erste Alternative aufgetreten ist. Die Abarbeitung folgt also dem Prinzip Tiefe-zuerst. Zur Kontrolle benutzt

man einen Stapelspeicher. Oben auf dem Stapel wird stets die nächste unbearbeitete Alternative des jüngsten Zustandes gesetzt. Nach dem Rücksetzen, wird diese zur Bearbeitung ausgewählt und gleichzeitig vom Stapel entfernt. Falls im gleichen Zustand eine weitere unbearbeitete Alternative vorhanden ist, wird diese auf den Stapel gesetzt. In jedem Zustand, zu dem zurückgesetzt wird, müssen die bis dahin erarbeiteten Ergebnisse sowie die aktuellen Werte der Parsingvariablen (wie z. B. die erreichte Position in der Eingabe) wiederhergestellt und später hinzugefügte Ergebnisse und geänderte Werte annulliert werden. Die benötigte Information kann man jedesmal, wenn man eine Alternative auf den Stapel setzt, mit abspeichern. Ökonomischer ist es, wenn man ohnehin beim Übergang von einem Zustand zum anderen, die Information über den alten Zustand nicht überschreibt, sondern die neue Information nur hinzufügt. Dann braucht im Rücksetzspeicher nur ein Zeiger auf die jeweils aktuelle Zustandsbeschreibung gesetzt zu werden, um diese später wiederherstellen zu können. Erst beim Rücksetzen werden Zustandsbeschreibungen überschrieben, nämlich alle, die chronologisch auf den Zustand, auf den zurückgesetzt wird, gefolgt sind. (Siehe Beispiele in Art. 32.) Schematisches Backtrakking ist überaus kostspielig. Der Aufwand wächst exponentiell mit der Länge der Eingabe. So kann es ja sein, daß bereits die erste Wahl falsch war, z. B. die Wahl zwischen Vollverb und Kopula in den Sätzen *Ist die Schreibmaschine in meinem Büro?* oder *Ist die Schreibmaschine in meinem Büro defekt?* Bis der Algorithmus zu dieser Stelle zurückkehrt, verfolgt er erst alle Verästelungen nachfolgender Alternativen. Ist die Anfangsentscheidung dann korrigiert, kann es sein, daß anschließend wieder dieselben Alternativen auftreten wie vorher. Sie müssen aber wieder gänzlich neu bearbeitet werden. Zu erwägen sind daher andere Organisationsprinzipien. Eine davon ist explizites Backtracking (vgl. Koch 1979, 24). Oft ist es beim Schreiben der Grammatik abzusehen, unter welchen Umständen der Parser in eine Sackgasse geraten kann und was die Ursache dafür ist. Man kann daher an diesen Stellen in der Grammatik explizit Rücksprungbefehle zu den richtigen Alternativen einfügen. Beim obigen Beispiel ist je nach Auftreten des prädikativen Adjektivs am Schluß des Satzes klar, daß ein Scheitern des Parsers an einer Fehlinterpretation des finiten Verbs am Anfang des Satzes gelegen haben muß. Man kann also sofort zu dieser Stelle zurücksetzen lassen. Wenn der Parser nur eine einzige Analyse finden muß, kann auch eine Anordnung der Alternativen nach abnehmender Wahrscheinlichkeit die Effizienz steigern, da nach einem ersten Erfolg die übrigen Alternativen nicht mehr bearbeitet werden müssen. Anders liegt der Fall, wenn verlangt wird, daß der Parser alle Lesungen einer mehrdeutigen Eingabe erkennen muß.

Parallelverarbeitung. Die parallele Abarbeitung von Alternativen entspricht dem Prinzip Breite-zuerst. Insofern diese sequentiell implementiert werden muß, werden die einzelnen Alternativen in der Reihenfolge, in der sie auftreten, bearbeitet. Man braucht zur Kontrolle also einen Speicher, dem am Ende Alternativen hinzugefügt und am Anfang Alternativen entnommen werden. Alle Alternativen sind verarbeitet, wenn im Gefolge der letzten anstehenden Alternative keine neue mehr entstanden ist, wenn also der Anfang des Speichers sozusagen das Ende einholt. Komplexer als die Ablaufkontrolle ist die Verwaltung der alternativen Zustände und Ergebnisse. Dazu eignet sich am besten eine zentrale Datenstruktur wie die oben erwähnten Tabellen oder Charts. Der große Vorteil dieser Organisation liegt darin, daß ein bestimmtes Ergebnis nur einmal erzeugt und dann in vielen parallelen Schritten verwendet werden kann.

Beschränkung der Alternativen. Man wird auch versuchen, die Ursachen für das nichtdeterministische Verhalten des Parsers möglichst zu beseitigen. Dazu gehört eine Verfeinerung der grammatischen Beschreibung, so daß fehlerhafte Zwischenresultate vermieden werden. Mit Hilfe komplexer Kategorien und des Unifikationsmechanismus macht man sich Kongruenzbeziehungen und sonstige Selektionen bei der Entscheidung zwischen scheinbaren Alternativen zunutze. Man wählt Parsingstrategien, bei denen der Kontext berücksichtigt wird, wie z. B. der vorangehende in links-assoziativen Ableitungen oder der folgende bei der prädikativen Analyse.

Aufschub von Entscheidungen. Eine sinnvolle Taktik ist es, Entscheidungen nicht zu früh zu treffen. So wird auch der menschliche Hörer nicht nachgrübeln, welcher Kasus bei *den alten Mann* oder bei *den alten Leuten*

vorliegt, solange er nur *den alten* gehört hat, sondern abwarten, bis das Substantiv aufgetreten ist. Es gibt Versuche, diesen Aufschub von Entscheidungen auch in Parsern nachzubilden (siehe Schank/Lebowitz/Birnbaum 1980; Marcus 1980).

Effizienzmaß. Im Zusammenhang mit dem nicht-deterministischen Verhalten von Parsern stellt sich die Frage nach einem Maß für die Effizienz der verschiedenen Algorithmen. Neben dem Aufwand an Speicherplatz ist es vor allem die Rechenzeit, welche die Effizienz ausmacht. Ein Maß für den Zeitaufwand verschiedener Parsingalgorithmen muß natürlich von der Implementierung auf einer konkreten Maschine unabhängig sein. Es sind zwei Größen, relativ zu denen die Rechenzeit anzugeben ist: die Abarbeitung der Grammatik und die Länge der Eingabe. Normalerweise nimmt man an, daß die Zeit, die ein Parser braucht, um die gegebene Grammatik abzuarbeiten, konstant ist, während der Aufwand, den die Eingabe verlangt, linear bis exponentiell mit der Anzahl der Elemente in der Eingabe steigt. Die Formel für die Effizienz eines Parsers lautet daher:

$$\text{Zeit} = \text{Konstante} \times \text{Länge}^{\text{Exponent}}$$

Eines der effizientesten Verfahren, der Early-Algorithmus, benötigt im schlimmsten Fall eine Zeit relativ zur Länge hoch drei. Allerdings ist die Konstante für die Abarbeitung der Grammatik nicht minder wichtig als die Länge der Eingabe. Ist diese Konstante sehr groß, macht sie eine eventuelle Überlegenheit des Parsers bei längeren Eingaben wieder zunichte. (Vgl. Charniak/McDermott 1985, 220 ff.)

4.6. Interaktion von Syntax, Semantik und Pragmatik

Semantisches Wissen. Wenn Hörer und Leser offensichtlich ohne Schwierigkeit die Struktur von Äußerungen erfassen, liegt es wohl daran, daß sie das Gehörte/Gelesene fortgesetzt interpretieren und in ein kognitives Modell des Gegenstandsbereiches einpassen. Es liegt daher nahe, auch die Probleme des maschinellen Parsings unter Zuhilfenahme der Semantik zu lösen. Es geht dabei hier nicht um die mit der Syntax ohnehin verschränkte Satzsemantik, sondern um die mit dem Weltwissen zusammenhängende Interpretation. (Dieser Unterschied wird freilich nicht immer gesehen.) Zur Weltwissenssemantik rechne ich auch die lexikalische Bedeutung der referentiell zu verwendenden Wörter, weil deren Gebrauchsbedingungen natürlich die Außenwelt einbeziehen müssen. Repräsentiert wird das semantische Wissen u. a. durch semantische und selektionale Merkmale, semantische Netze, Begriffssysteme, Rahmen (frames, scripts, templates), Wissensbasen und Datenbanken verschiedener Art. Auf die einzelnen Formen kann hier nicht eingegangen werden. Im großen und ganzen besteht das semantische Parsing darin, aus den Wörtern und Satzteilen Begriffe zu abstrahieren und Kollokationen dieser Begriffe mit Mustern zu vergleichen, nach denen die Begriffe einander im vorliegenden Objektbereich zugeordnet sind. Formal ist semantisches Parsing daher selten mehr als ein oberflächennahes Skimming. (Vgl. Minsky 1968; Winograd 1972; Riesbeck 1975; Schank 1975; Wilks 1975; Charniak/Wilks 1976; Wilks 1976a und 1976b; Winograd 1976b und 1976c; Eisenberg 1977; Hendrix 1977a; Schank/Abelson 1977; Waltz 1978; Wilks 1978b; DeJong 1979a; Landsbergen/Scha 1979; Laubsch 1979; Harris 1980b; Dyer 1981; Kunze 1981; Charniak 1982; Bunt/Schwartzenberg 1982; Riesbeck 1982; Lebowitz 1983a; Lesmo/Torasso 1983; Ritchie 1983; Wilks 1983; Capelli/Ferrari/Moretti et al. 1984; Simmons 1984.)

Pragmatik. Es sind auch Versuche unternommen worden, die Pragmatik der Äußerungen bei der Analyse zu berücksichtigen, wie Sprecher-Hörer-Verhältnis, Dialogkontext, Thematik. Eine pragmatische Größe ist die Textkohärenz, die sich syntaktisch im Aufbau des Textes niederschlägt. In Zukunft wird daher das Parsen von Sätzen vielleicht im Parsen von Texten aufgehen. Zur Zeit existieren dazu jedoch nicht mehr als rudimentäre Ansätze. (Vgl. Weizenbaum 1967; Bobrow/Kaplan/Kay 1977; Sallis 1978; Rostek 1979; Parkinson/Colby/Faught 1977; Correira 1980; Silva/Dwiggins 1980; Hahn, W. v./Hoeppner/Jameson et al. 1980; Bátori 1981a; Hirst 1981; Wilensky 1981; Hobbs 1982; Hahn 1984; Hellwig 1984; Leinfellner/Steinacker/Trost 1984; Rollinger 1984; Sidner 1985.)

Kontrollstrukturen. Das Zusammenspiel von Syntax und Semantik (die Pragmatik in letzterer eingeschlossen) kann wie folgt geregelt werden: (i) Syntax ohne Semantik: Die Syntaxkomponente erzeugt, soweit sich Mehrdeutigkeiten nicht allein syntaktisch

auflösen lassen, mehrere Ergebnisse. Die Lesungen werden anschließend an die Semantikkomponente zur Entscheidung weitergegeben (serielle Architektur). (ii) Syntax mit Semantik unter syntaktischer Kontrolle: Die Semantikkomponente wird von der Syntax gezielt aufgerufen, wenn Ambiguitäten auftreten. Unterschiede ergeben sich daraus, wie frühzeitig dies geschieht. Hierher sind auch selektionale semantische Merkmale zu rechnen, die syntaktischen Positionen zugeordnet werden. (Zur Abwägung des Nutzens siehe Woods 1973b, 144. Es ist keineswegs sicher, daß semantische Routinen dem Parser mehr Arbeit ersparen, als die autonome Syntaxanalyse dem semantischen Interpreter Arbeit abnimmt.) (iii) Semantik ohne Syntax oder Syntax unter semantischer Kontrolle: Die Eingabe wird nach Begriffen durchmustert, die sich in ein Modell des Objektbereiches einpassen lassen, das z. B. Beschreibungen typischer Situationen, Gegenstände, Personen, Handlungen usw. umfaßt. Im Extremfall wird die Syntax überhaupt nicht beachtet. Ansonsten wird sie nur hinzugezogen, wenn die Begriffsfolgen mehrere Interpretationen erlauben. Im Ansatz gehört auch der Versuch hierher, die syntaktische Analyse von sog. Tiefenkasus her zu steuern. (iv) Syntax und Semantik integriert: Die Syntax erzeugt fortwährend eine Eingabe für die Semantik (kaskadierte Architektur) oder die Kontrolle wechselt: einmal führen syntaktische, einmal führen semantische Kriterien zum nächsten Schritt. (Vgl. Woods 1980; Gehrke 1982; Christaller/Metzing 1983; Phillips 1983; Ritchie 1983; Granger/Eiselt/Holbrook 1984; Charniak/McDermott 1985, 238 ff.; Christaller 1985; Crain/Steedman 1985.)

4.7. Ausgabe einer Strukturrepräsentation

Strukturbeschreibung. Von einem Parser verlangen wir eine Darstellung des Aufbaus des Eingabesatzes. Von einem Transduktor verlangen wir eine disambiguierte Repräsentation der Eingabe, die für die Zwecke der Weiterverarbeitung innerhalb des natürlichsprachigen Systems geeignet ist. Als Darstellung des Aufbaus kommen vor allem Phrasenstrukturbäume oder Dependenzbäume infrage. (In Entwicklungssystemen sind auch Herleitungen der Strukturen als Ausgabe wünschenswert.) Es gibt zwei Möglichkeiten, für die Ausgabe eines Strukturbaumes zu sorgen: (i) Der Erkenner produziert Daten, auf deren Grundlage anschließend ein separater Algorithmus einen Strukturbaum erzeugt. Die Daten können z. B. aus den Nummern der Regeln, die für eine rechte oder linke Ableitung der Eingabekette benötigt wurden, bestehen (d. i. ein sog. 'Kennwort' oder 'Parse', Mayer 1978, 25). Oder es sind zu den ermittelten syntaktischen Einheiten jeweils die unmittelbaren Konstituenten notiert worden (vgl. Brockhaus 1971, 34 f.). (ii) Es werden parallel zum Erkennen der syntaktischen Einheiten entsprechende Strukturbäume aufgebaut. Diese können sogar zur Ablaufkontrolle des Erkenners benutzt werden.

Repräsentation. Als Repräsentation für die Weiterverarbeitung kommen beliebige Sprachen infrage. Häufig werden Logik-Formalismen oder Datenbank-Abfrage-Sprachen verwendet. Es gibt zwei Möglichkeiten eine Repräsentation herzustellen: (i) Nachdem ein Strukturbaum zur natürlichsprachigen Eingabe erzeugt wurde, wird dieser mit Hilfe eines separaten Algorithmus in die Repräsentation übersetzt. (ii) Parallel zur Syntaxanalyse der natürlichen Sprache wird bereits die angestrebte Repräsentation aufgebaut. Die Voraussetzung dafür ist allerdings, daß sich die Teilstrukturen der natürlichen Sprache und der Repräsentationssprache korrelieren lassen (rule-to-rule Prinzip). Der Unifikationsmechanismus bietet eine gute Handhabe, parallel zwei Strukturen zu erzeugen, wobei auch etwaige Restriktionen der semantischen Repräsentationssprache auf die syntaktische Analyse der natürlichen Sprache zurückwirken. Auf diese Weise können Satzsyntax und Satzsemantik elegant integriert werden. (Zum zwei-phasigen Verfahren vgl. Berry-Rhogge/Wulz 1977; Bronnenberg/Bunt/Landsbergen et al. 1979; Landsbergen 1981; Bunt/Schwartzenberg 1982; Halvorsen 1982a und 1982b; Frey 1984; zum rule-to-rule Verfahren Brockhaus/Stechow 1971; Friedman/Warren 1978; Bohnert/Backer 1979; Gawron/King/Lamping et al. 1982; Root 1982; Warren/Friedman 1982; Karttunen/Kay 1985; Gazdar/Klein/Pullum et al. 1985.)

Peter Hellwig, Heidelberg (Bundesrepublik Deutschland)