Grammatikformalismen in der Generierung und ihre Verarbeitung

Günter Neumann

Deutsches Forschungszentrum für
Künstliche Intelligenz
Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11
neumann@dfki.uni-sb.de

Zusammenfassung Dieser Aufsatz gibt einen Überblick gegenwärtig verwendeter Grammatikformalismen und ihre Verarbeitung in Sprachproduktionssystemen. Es handelt sich hierbei um constraintbasierte Formalismen, systemische Grammatiken und Baumadjunktionsgrammatiken. Es wird an Hand ausgewählter Arbeiten gezeigt, wie diese Formalismen in der Generierung eingesetzt werden. Im letzten Abschnitt wird das Problem der Reversibilität von Grammatiken, d.h. die Verwendung ein und derselben Grammatik für Parsing und Generierung, diskutiert.

Einleitung

Ein zentrales Ziel im Bereich der Generierung natürlicher Sprache ist die Entwicklung von Computersystemen, die auf der Basis einer gegebenen Eingabe (in Form von zugrundeliegenden Zielen, die den kommunikativen Absichten des Gesamtsystems entsprechen) eine ihr entsprechende, kommunikativ adäquate, natürlichsprachliche Äußerung erzeugen. Es hat sich dabei als sehr hilfreich erwiesen, den sprachlichen Produktionsprozeß als einen komplexen, auf mehreren Ebenen ablaufenden Enscheidungsprozeß zu modellieren. Die Menge der zu treffenden Entscheidungen läßt sich grob in solche unterteilen, die sich auf die Inhaltsbestimmung bzw. Inhaltsrealisierung beziehen. Wichtige Prozesse der Inhaltsbestimmung sind die Auswahl und Organisation der Information, die für das aktuelle Diskursziel (z.B. Beantworten einer Frage) relevant ist. In der Inhaltsrealisierung steht die Überführung dieser Information in eine gesprochene oder geschriebene Äußerung (d.h. die konkrete Antwort) im Vordergrund. Die hierfür benötigten, zentralen Wissensquellen sind ein Lexikon und eine Grammatik. Die Entscheidungsprozesse umfassen demnach die Auswahl lexikalischer Elemente und die Konstruktion der gram-

matikalischen Struktur für die geplante Äußerung. Dieser letzte Punkt ist Gegenstand der vorliegenden Arbeit.

Zur Beurteilung der verschiedenen Formalismen ist es nützlich das Verhältnis zwischen Form und Funktion linguistischer Objekte zu betrachten. Zum einen können Äußerungen aufgrund ihrer syntaktischen Struktur klassifiziert werden (z.B. als Frage oder Antwort), unabhängig davon, was mit einer aktuellen Äußerung tatsächlich intendiert ist. Die Beantwortung solcher Fragen wird dem Bereich der Pragmatik zugesprochen. Dort werden linguistische Objekte beurteilt nach ihrer Funktion, die sie in verschiedenen Kommunikationssituationen einnehmen. Beispielsweise kann die Aufforderung etwas zu tun je nach Gesprächssituation als Bitte oder Befehl formuliert werden und umgekehrt kann eine Äußerung als Frage oder Aufforderung verstanden werden. Für ein umfaßendes Verständnis von Sprache können beide Aspekte natürlich nicht getrennt voneinder untersucht werden. Bezogen auf die Generierungsaufgabe heißt dies, daß auch die Beziehung zwischen Funktion und Form formuliert werden muß.

In natürlichsprachlichen Generierungssystemen NLG werden eine Reihe von unterschiedlichen Grammatikformalismen zur Produktion der syntaktischen Struktur verwendet. Noch bis Anfang der achtziger Jahre wurden Grammatikformalismen für Generierungssysteme im wesentlichen von Forschern aus dem Bereich der Künstlichen Intelligenz (KI) entwickelt und implementiert. Dabei stand die Integration der Grammatik in das Gesamtsystem (und damit auch die Beziehung zwischen Funktion und Form) mehr im Vordergrund, als eine umfaßende und linguistisch fundierte Beschreibung der Form von Sprache.

Mit der wachsenden Bedeutung der constraintbasierten Grammatikformalismen stieg das Interesse an Generierungsfragen im Bereich der Computerlinguistik (CL) erheblich und führte zu einer starken Systematisierung des syntaktisch orientierten Generierungsproblems. Im Unterschied zu den mehr KI-orientierten Ansätzen, in denen anwendungs- und diskursspezifische Fragestellungen im Vordergund stehen, wird in der CL hauptsächlich die Beziehung zwischen semantischer Repräsentation und syntaktischen Ausprägungsmöglichkeiten auf der Basis linguistisch fundierter Methoden diskutiert. Pragmatische Fragestellungen werden jedoch derzeit nicht berücksichtigt. So betrachtet beschäftigen sich computerlinguistische Arbeiten im Bereich der Generierung in erster Linie mit der Inhaltsrealisierenden Aufgabe.

Ausgangspunkt der Inhaltsrealisierung Die Eingabespezifikation für den inhaltsrealisierenden Generierungsteil (kurz Grammatikgenerator) beinhaltet semantische und pragmatische Information. Idealerweise sollte nur sehr wenig syntaktische Information spezifiziert sein (z.B. das Satzsymbol), denn es gilt ja gerade diese zu konstruieren. Weiterhin sollte eine gewisse Flexibilität darüber erlaubt sein, wie exakt die Eingabe spezifiziert sein muß, da i.a. nicht immer alle semantischen und pragmatischen Merkmale

verfügbar sein müssen. Der Grammatikgenerator sollte prinzipiell in der Lage sein, trotz fehlender Information adäquate Ergebnisse zu liefern.

Im Grammatikgenerator sind im wesentlichen die zwei folgenden Teilaufgaben zu lösen:

1. Auswahl der lexikalischen Elemente
2. Konstruktion der syntaktischen Struktur dieser Elemente

Beide Aufgaben können als Entscheidungsprozesse modelliert werden, die bestimmten *Beschränkungen* unterliegen. Ein wesentlicher Teil dieser Beschränkungen wird im Lexikon und in der verwendeten Grammatik formuliert. Es folgt eine Auswahl der Entscheidungen, die bei der Lösung der oben genannten Teilaufgaben getroffen werden müssen, wobei viele der Entscheidungen nur adäquat durch eine Interaktion der Teilaufgaben gelöst werden kann.

- Wahl von Inhaltswörtern (*Haus, Gebäude, Wolkenkratzer*)
- Wahl von definiten Kennzeichnungen (*der Ball* vs. *ein Ball*)
- Wahl von Proformen (*dort, er, wir*)
- Wahl von Konnektoren (*aber, jedoch, dann*)
- Realisierung von Negation (*ich habe kein Geld*, *ich gehe nicht nach Hause*)
- Realisierung von Modifikatoren (z.B. als Adjektiv *der grüne Apfel* oder Relativsatz *der Apfel, der grün ist*)
- Realisierung von Fokus (z.B. durch Passivierung oder Vorfeldverschiebung)
- Kongruenz zwischen Subjekt und Verb (*Peter fährt*, *die Kinder gehen*)
- Wohlgeformte Wortfolge (z.B. nicht: *Apfel der grün*)
- Wahl der morphologischen Flexion (*des grünen Apfels*)
- Ellision von Satzgliedern (*Peter besitzt rote, Maria grüne Klötze.*)
- Konstruktion von Infinitiven (*Ich bitte Dich zu kommen*)

Von großer Wichtigkeit hierbei ist die Frage, wie spezifisch die Eingabe bezogen auf die möglichen Freiheitsgrade in der Grammatik und dem Lexikon angenommen wird und wie auf fehlende Information reagiert werden soll. In sehr vielen Systemen wird davon ausgegangen, daß die Eingabe notwendig und hinreichend bezogen auf die Entscheidungspunkte spezifiziert ist, d.h. es wird angenommen, daß alle Information vorhanden ist, um zielsicher zur besten Realisierung zu kommen. In einigen Systemen darf die

Eingabe in dieser Hinsicht unterspezifiziert sein. Dann kann entweder durch Einsatz von Default-Werten oder durch eine Interaktion mit anderen Modulen (z.B. Dialogkomponente, Textplanner) die fehlende Information bestimmt werden. Die Beschreibung eines auf einer Interaktion basierenden Systems findet sich in [Hovy, 1987].

Wir werden nun im folgenden einige der bekanntesten Formalismen einführen. Es handelt sich dabei um constraintbasierte Formalismen (im Besonderen FUG), systemische Grammatiken und Baumadjunktionsgrammatiken. Wir werden an Hand ausgewählter Arbeiten zeigen, wie sie in der Generierung eingesetzt werden. Im Anschluß daran werden wir auf das Problem der Reversibilität eingehen.

Constraintbasierte Grammatiken

Gegenwärtig stehen constraintbasierte Formalismen im Mittelpunkt der Grammatikorientierten Forschung. Die zentrale Idee dieser Formalismen ist die Beschreibung linguistischer Kategorien durch komplexe Merkmalsstrukturen und die Realisierung des Informationsflusses während der linguistischen Verarbeitung durch Unifikation solcher Merkmalsstrukturen. Zu den bekanntesten Vertretern dieser Formalismenklasse gehören Functional Unification Grammar [Kay, 1984], PATR II [Shieber *et al.*, 1983] und Definite Clause Grammars [Pereira und Warren, 1980], sowie Lexical Functional Grammar [Bresnan, 1982], unifikationsbasierte Kategorialgrammatiken (z.B. [Uszkoreit, 1986]) und Head-Driven Phrase Structure Grammar (HPSG, [Pollard und Sag, 1987]).

Stellvertretend für die Modellierung und Verarbeitung von linguistischem Wissen in constraintbasierten Formalismen werden wir im nächsten Abschnitt den Formalismus der Functional Unification Grammar genauer beschreiben. Auf spezielle Verarbeitungsaspekte einiger der anderen Formalismen gehen wir im letzten Abschnitt ein.

Functional Unification Grammar

Die Functional Unification Grammar FUG [Kay, 1975; Kay, 1984] wurde in zahlreichen sehr prominenten Generierungssystemen zur Verarbeitung der grammatikalischen Struktur eingesetzt (u.a. in [Appelt, 1985], [McKeown, 1985] und [McKeown *et al.*, 1990]).

Repräsentation In FUG werden alle linguistische Entitäten (d.h. Wörter, Phrasen und Sätze) mittels *funktionaler Beschreibungen* (Functional Descriptions FDs) repräsentiert. Eine FD ist eine Matrix bestehend aus Attribut/Wert-Paaren, genannt *Merkmale* (features). Sowohl die Eingabe (die von dem inhaltsbestimmenden Teil berechnet wird) als auch die Grammatik und das Lexikon werden als FDs repräsentiert. Als einzige Operation ist in FUG die *Unifikation* von FDs erlaubt. Informell führt die Unifikation

zweier FDs zur Konstruktion einer neuen FD, die die Information der Eingabestrukturen beinhaltet und mit diesen kompatibel ist.

Abbildung 1 zeigt eine FD für eine (stark vereinfachte) Beispielgrammatik des Deutschen, in der Sätze als Subjekt–Verb–Objekt oder Subjekt–Verb–Folgen definiert werden.

$$
\left[\begin{array}{l}
\texttt{Cat} = S \\
\texttt{Pat} = (< Subj > < Verb > \ \ldots) \\
\texttt{Subj} = \left[\texttt{Cat} = NP\right] \\
\texttt{Verb} = \left[\begin{array}{l}\texttt{Cat} = V \\ \texttt{Sem} = \langle\uparrow \ Sem\rangle \\ \texttt{Num} = \langle\uparrow \ Subj \ Num\rangle\end{array}\right] \\
\left\{\begin{array}{l}\left[\texttt{Obj} = NONE\right] \\ \left[\texttt{Obj} = \left[\begin{array}{l}\texttt{Cat} = NP \\ \texttt{Pat} = (\ldots \ Obj)\end{array}\right]\right]\end{array}\right\} \\
\texttt{Sem} = \left[\begin{array}{l}\texttt{Pred} = ANY \\ \texttt{Agent} = \langle\uparrow \ Subj \ Sem\rangle \\ \left\{\begin{array}{l}\left[\texttt{Goal} = NONE\right] \\ \left[\texttt{Goal} = \langle\uparrow \ Obj \ Sem\rangle\right]\end{array}\right\}\end{array}\right]
\end{array}\right]
$$

Abbildung 1: Eine einfache Beispielgrammatik in FUG.

Merkmale werden in der Form $a = v$ notiert, wobei a ein Attribut und v dessen Wert bezeichnet. Attribute sind beliebige Wörter ohne interne Struktur, wogegen Werte durch unterschiedliche Typen beschrieben werden können. Im wesentlichen wird zwischen atomaren und komplexen Werten unterschieden. In obiger Abbildung besitzt `Cat` atomare Werte und die Attribute `Subj` und `Pat` haben komplexe Werte (im Falle von `Subj` ist der Wert selbst eine FD).

Wie das Beispiel zeigt, kann eine FD auch Disjunktionen von Merkmalsmengen beinhalten. Die Disjunktion in dem Beispiel (notiert mittels geschweifter Klammern) drückt aus, daß das *Objekt* optional ist und daher Sätze aus zwei oder drei Konstituenten bestehen können. Der *spezielle* Wert `NONE` besagt, daß diese Alternative nur gewählt wird, falls in der Eingabestruktur kein Objekt spezifiziert ist. Das Attribut `Sem` kodiert die Semantik des Satzes. Sie ist identisch mit dem des Verbes (ausgedrückt durch den Pfad

$\langle\uparrow\ Sem\rangle$ in der FD des Verbes). Der Wert `ANY` besitzt den Status einer Variablen mit der zusätzlichen Bedingung, daß am Ende der Unifikation eine Substitution mit einem aktuellen Wert stattgefunden haben muß.

Der Wert des Attributes `Num` bezeichnet einen *Pfad*. Pfade werden verwendet um auf Werte anderer Attribute zu verweisen. Damit wird ausgedrückt, daß zwei Attribute (die zu unterschiedlichen Konstituenten gehören können) einen gemeinsamen Wert haben oder anders ausgedrückt, daß ihre Werte identisch sein müssen. In dem Beispiel ist somit festgelegt, daß der Numerus des Verbs mit dem Numerus des Subjekts identisch sein muß.[1] Dadurch können Phänomene wie zum Beispiel Kongruenz oder Projektion (d.h., die Beziehung zwischen einer Phrase und ihrem Head-Element; z.B. ist die Semantik eines Satzes identisch mit der des Verbes) sehr elegant formuliert werden, wofür in früheren Formalismen oft nur prozedurale Lösungen vorhanden waren (vgl. z.B. [Meteer *et al.*, 1987]).

Information über die Abfolge der Konstituenten wird mit dem Attribut `Pat` formuliert. Der Wert dieses Attributes ist eine Liste von Pfaden auf die unmittelbaren Konstituenten der FD. Diese Liste formuliert Positionsbeschränkungen für die Reihenfolge der Teilausdrücke der Konstituenten eines Satzes. Formal beschreibt die Liste einen regulären Ausdruck über einem Alphabet bestehend aus den Attributen der Konstituente in der das Merkmal `Pat` auftritt. In der Beispielgrammatik bezeichnet das Attribut `Pat` der Kategorie *S*, das der Oberflächenstring eines Satzes mit dem Subjekt beginnt gefolgt vom Verb und eventuell weiterer Konstituenten. Die Optionalität wird durch ... ausgedrückt. In der Beispielgrammatik kann diese Position durch das Objekt gefühlt werden, was bedeutet, daß das Objekt am Ende eines Satzes stehen muß. Um die Konstituenten einer FD genau zu identifizieren, verfügt FUG noch über das Attribut `CSet`, daß als Wert gerade die Menge der Pfade auf die unmittelbaren Konstituenten besitzt.[2]

Unifikation Die einzige Operation, die zur Verarbeitung eingesetzt wird, ist die Unifikation; dies gilt insbesondere auch für die Generierung von Sätzen. In FUG werden Sätze durch Unifikation der Grammatik mit der Eingabe produziert, die wie die Grammatik auch als FD formuliert sein muß. Die Eingabe für den Unifikator ist eine semantische Repräsentation dessen, was geäußert werden soll. Folgende Abbildung zeigt eine mögliche (natürlich wieder vereinfachte) Eingabestruktur für obige Bespielgrammatik.

Die Ausgabe ist dann eine vollspezifizierte FD des Satzes 'Peter liebt Maria', wobei das Attribut `Pat` der Satzkonstituente die lineare Abfolge beschreibt. Verfüge unsere

[1] Das Zeichen $\uparrow$ drückt aus, daß der Startknoten des Pfades in der nächst höheren FD zu finden ist. Daß zwei Attribute sich denselben Wert teilen, wird auch als *structure sharing* bezeichnet.

[2] Damit kann FUG zu den ID/LP Formalismen (vgl. [Gazdar *et al.*, 1985]) gerechnet werden, in denen eine explizite Trennung von Konstituentenstruktur und linearer Abfolge ausgedrückt wird.

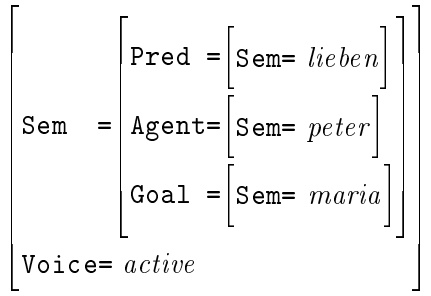

Abbildung 2: Eingabestruktur für den Satz 'Peter liebt Maria'.

Beispielgrammatik in Abb. 1 zusätzlich über Passivkonstruktionen, so könnte auch der Satz 'Maria wird von Peter geliebt' erzeugt werden, falls in Abb. 2 das Attribut `voice` mit dem aktuellen Wert `passive` belegt wäre.

Unifikation kann informell aufgefaßt werden als 'Kombination von Information'. Wenn zwei FDs unifiziert werden, sollte die resultierende FD alle Information der Eingabestrukturen beinhalten. Es sind hierbei zwei Basisfälle zu unterscheiden, jenachdem ob der Wert eines Attributs atomar oder komplex ist. Atomare Werte unifizieren nur dann, wenn sie identisch sind. Die Unifikation komplexer Werte kann als rekursive Mengenvereinigung aufgefaßt werden. Die Unifikation von Werten des Attributes `pat` ist der Durchschnitt der Werte. Während der Unifikation wird auch überprüft, ob die zu unifizierenden Strukturen kompatibel sind. Dies ist zum Beispiel nicht der Fall, wenn die Werte eines Attributes, das auf gleicher Ebene in beiden FDs auftritt, nicht unifizieren (was z.B. der Fall ist, wenn atomare Werte ungleich sind, oder der Typ der Werte unterschiedlich ist).[3] Wesentliche Aspekte der Unifikationsoperation sind: (1) Ordnungsinvarianz, d.h. daß die Ordnung der Merkmale in den Eingabestrukturen keine Rolle spielt, (2) Kommutativität (oder Bidirektionalität), (3) Monotonie und (4) Deklarativität — eine Grammatik kann damit aufgefaßt werden als eine Menge von Merkmalsbeschränkungen, die zu einer Eingabe addiert oder überprüft werden, wobei nur ausgedrückt ist, *welche* Beschränkungen wichtig sind, nicht die Reihenfolge ihrer Verarbeitung.

FUG und Generierung Der Vorteil von FUG gerade für die Generierung hängt mit der Art und Weise zusammen, wie syntaktisches Wissen kodiert wird. Für die Analyse linguistischer Strukturen hat sich eine hierarchische Baumstrukturierung als geeignet erwiesen. FUG dagegen erlaubt es, eine flachere Struktur der Ordnungsbeschränkungen zu formulieren. Damit kann eine stärkere Betonung der funktionalen Rollen der Kon-

[3] Eine formale Beschreibung der Unifikation in contraintbasierten Grammatikformalismen kann z.B. in [Shieber, 1986] gefunden werden.

stituenten repräsentiert werden, was gerade für die Generierung von großer Bedeutung ist. In der Generierung steht die paradigmatische Relation zwischen linguistischen Objekten im Vordergrund, d.h. die Auswahl zwischen Alternativen und deren Einfluß auf die Auswahl folgender Elemente. In der Verstehensrichtung steht dagegen die syntagmatische Relation im Vordergrund, d.h. Beschränkungen bezüglich der Kombination von Objekten.

Die Unifikation wird häufig (zumindest so, wie sie in FUG für die Generierung formuliert ist) als ein nicht-deterministischer top-down/depth-first Kontrollfluß realisiert. Dies bedeutet allerdings, daß die Verarbeitung erheblich ineffizient ist. Um diesem Problem zu begegnen wurde in dem Formalismus FUF [Elhadad, 1989], der in dem Generierungssystem COMET [McKeown *et al.*, 1990] verwendet wird, die Möglichkeit geschaffen, in der Grammatik Kontrollinformation zu spezifizieren. So können zum Beispiel Präferenzen für die Auswahl von Alternativen spezifiziert werden oder es kann festgelegt werden, ob die Merkmale einer FD mittels einer tiefen– oder breitenorientierten Strategie verarbeitet werden sollen.

Spezielle Verarbeitungsaspekte Für stark lexikonzentrierte linguistische Theorien, wie z.B. HPSG [Pollard und Sag, 1987] ist eine strikte top-down Verarbeitung ungeeignet. In solchen Theorien befindet sich der Großteil an grammatikalischer Information in den lexikalischen Einträgen. Die Regeln werden dagegen sehr abstrakt durch Schemata beschrieben. Zum Beispiel wird in den Regeln von der speziellen Subkategorisierungsinformation der Verben abstrahiert, da die spezifische Information bei den jeweiligen Verben eingetragen ist. Prinzipiell würde ein Regelschema der Form $M \longrightarrow H\ C^*$ ausreichen, um z.B. Verb-Komplement Strukturen zu erzeugen (M ist der Mutterknoten, H der Kopf und C^* eine beliebige, möglicherweise leere Folge von Komplementen. Wir wollen weiterhin annehmen, daß dieses Schema nichts über die korrekte Wortfolge aussagt.). Wenn wir dieses Schema auf das Verb *lieben* anwenden (das zwei Nominalphrasen subkategorisiert) ergäbe sich z.B. folgende Regelinstanz: $S \longrightarrow V\ NP\ NP$, wobei S als Projektion von V interpretiert wird.

Bei einer top-down-gesteuerten Verarbeitung müßte daher praktisch mit einer unspezifizierten Subkategorisierung begonnen werden. [Shieber *et al.*, 1991] zeigen, daß damit aber Terminierungsprobleme auftreten und schlagen als Lösung ein bottom-up Verfahren vor. In diesem Verfahren werden zuerst die zur (semantischen) Eingabe korrespondierenden lexikalischen Einträge bestimmt. Damit ihr Verfahren die strukturelle Information der Eingabe beschränkend einsetzen kann, wird zuerst für den semantischen Kopf (das ist meist das Prädikat) ein lexikalischer Zugriff durchgeführt. Danach werden mittels der grammatikalischen Struktur dieses Elementes dessen (semantische) Argumente bestimmt und der gesamte Prozeß rekursiv über den gefundenen Argumenten fortgeführt.

Systemische Grammatiken

Im Vordergrund der Entwicklung systemischer Grammatiken [Halliday, 1985; Winograd, 1983] steht die *Verwendung* von Sprache in konkreten Kommunikationssituationen, d.h. die Frage nach der sozialen Funktion von Sprache. Zentrale Idee in systemischen Grammatiken ist die Klassifikation sprachlicher Äußerungen gemäß der Funktion, die sie in Gesprächssituationen einnehmen können, d.h. ob die Äußerung z.B. als Frage, Anweisung oder Aussage zu realisieren ist. Ein zentrales Ziel systemischer Grammatiker ist es, diese Art der Klassifikation so fein und spezifisch wie möglich zu gestalten. Die Komplexität der Struktur natürlicher Sprachen wird dabei im wesentlichen gemäß der folgenden drei Dimensionen untersucht (s.a. [Winograd, 1983]): (a) Klassifikation (z.B. Satzarten, Wortklassen), (b) Gruppierung (z.B. Sequenzen von Wörtern als Konstituenten) und (c) Funktion (d.h., die funktionalen Beziehungen zwischen Elementen).

Gerade die Betonung der paradigmatischen und funktionalen Beziehungen zwischen linguistischen Objekten machen systemische Grammatiken besonders attraktiv für die Generierungsaufgabe, da hierdurch beschrieben wird, welche sprachlichen Mittel welchen kommunikativen Situationen zugeordnet werden können. Nicht die Form, sondern die Funktion linguistischer Objekte steht im Vordergrund der Fragestellungen.

In einer systemischen Grammatik wird linguistisches Wissen durch ein Netzwerk von miteinander in Beziehung stehenden *Systemen* repräsentiert, wobei auf der untersten Ebene spezifischere Entscheidungen zu treffen sind (z.B. Wortwahl) und auf der obersten, die weniger spezifischeren (z.B. Satztyp). Ein einzelnes System wird durch eine Menge von Merkmalen und ihren möglichen Werten repräsentiert, wobei konjunktive und disjunktive Verknüpfungen möglich sind. Abbildung 3 zeigt einen Ausschnitt des Systems für Englische Pronomen (konjuktiv verknüpfte Entscheidungen werden durch { und disjunktive durch [notiert).

Das Pronomen *I* wird z.B. gewählt aufgrund der Information *Personal Singular First Subjective*, *she* mittels *Personal Singular Third Feminine Subjective* und *those* mittels *Demonstrative Far Plural*.

Das Generieren von Sätzen in systemischen Grammatiken resultiert als eine Serie von funktionalen Entscheidungen, die parallel entlang der verschieden funktionalen Bereiche durchgeführt werden. Das System PENMAN mit seiner systemischen Grammatik NIGEL ist das bekannteste System, das auf dieser Theorie aufbaut [Matthiesen, 1985]. Dort ist eine sehr große Grammatik des Englischen entwickelt worden. In NIGEL entsteht ein Satz durch eine Top/Down gesteuerte Traversierung der systemischen Netzwerke. Die Werte der einzelnen Merkmale werden von der Umgebung bestimmt. Wenn z.B. der Wert für das Merkmal *Modus* bestimmt werden soll (er ist entweder *imperativ* oder *indikativ*), so wird eine Anfrage an die semantische Repräsentation gestellt, ob es sich um einen Befehl handelt oder nicht. Zu diesem Zweck sind mit jedem System Auswahl-

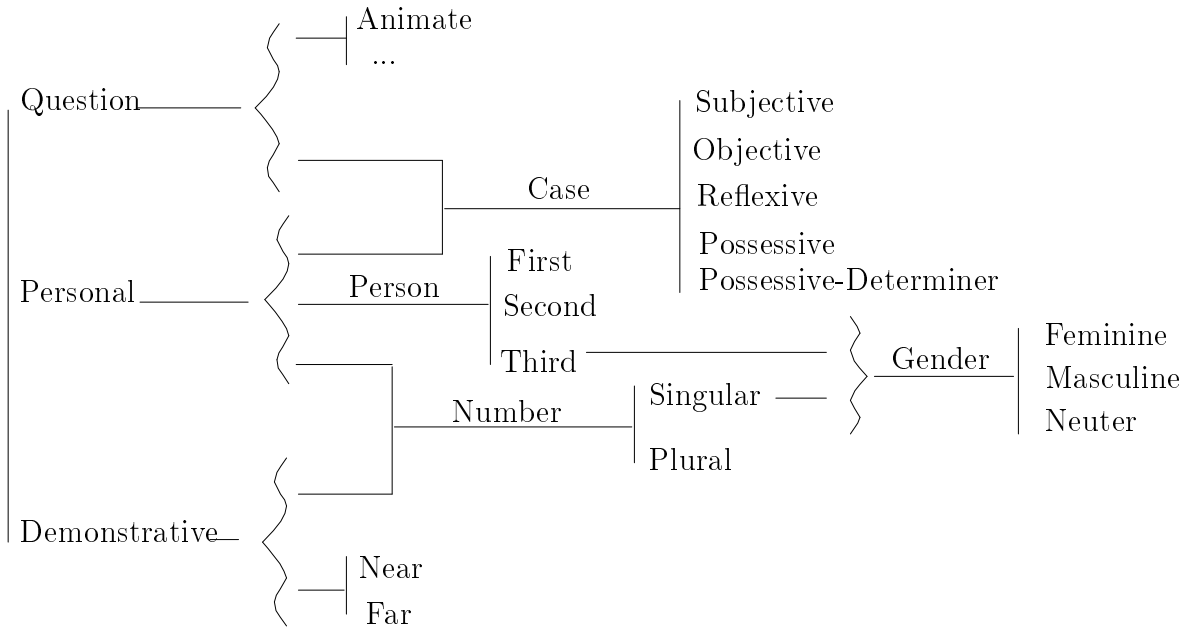

Abbildung 3: Das System Englischer Pronomen (nach [Winograd 83]).

prädikate definiert, die Tests anhand des Hintergrundprogramms durchführen, um die aktuellen Werte eines Merkmals bestimmen zu können. In Abhängigkeit der Ergebnisse solcher Anfragen und des aktuellen Systemzustands werden die entsprechenden Nachfolge-Systeme ausgewählt. Diese Art der Merkmalsbestimmung wird 'inquiry semantics' genannt und ermöglicht es der Grammatik, mit den Wissensquellen des Hintergrundprogramms zu kommunizieren. Im System PENMAN sind das Weltwissen und die semantische Repräsentation in einem KL–ONE basierten Formalismus repräsentiert. Die inquiry semantics dient als Vermittler zwischen Grammatik und Wissensrepräsentation. Nachteilig in diesem System ist allerdings, daß die Kommunikation stark prozedural realisiert ist und daher sehr abhängig von der konkreten Anwendung.

Tree Adjoining Grammars

Baumadjunktionsgrammatiken (Tree Adjoing Grammars TAGs) wurden erstmals von [Joshi *et al.*, 1975] eingeführt und im Laufe der Jahre eingehend formal und linguistisch untersucht. Es wurden eine Reihe von TAG Varianten entwickelt, wobei speziell lexikalisierte TAGs [Abeille, 1988] und synchrone TAGs [Shieber und Schabes, 1990] für die Generierung relevant sind.

In der ursprünglichen Definition besteht eine TAG aus einer endlichen Menge von elementaren Bäumen, die unterteilt wird in eine Menge von initialen und eine Menge

auxiliarer Bäume. Die Menge der elementaren Bäume konstituiert die strukturelle Basis einer TAG. Initiale Bäume repräsentieren minimale Satzstrukturen. Die Wurzel eines initialen Baumes bezeichnet das Satzsymbol S und die Blattknoten bezeichnen terminale Elemente. Alle Zwischenknoten bezeichnen nichtterminale Elemente. Die linke Graphik in Abbildung 4 zeigt graphisch die Struktur initialer Bäume. Bei auxiliaren Bäumen kann die Wurzel ein beliebiges nichtterminales Symbol bezeichnen. Bis auf ein Element x bezeichnen alle Blattknoten terminale Knoten. x bezeichnet ein nichtterminales Element gleicher Katgorie wie die Wurzel (siehe rechte Graphik in Abbildung 4).

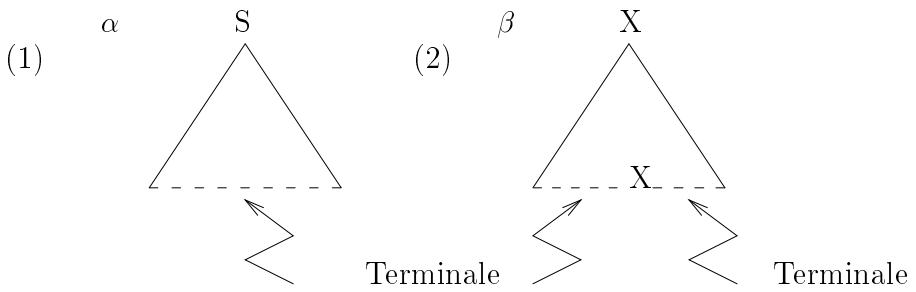

Abbildung 4: Struktur initialer (1) und auxiliarer (2) Bäume (nach [Joshi 87]).

Im Gegensatz zu kontextfeien Grammatiken (kfG), wo Regeln als Bäume mit Tiefe eins repräsentiert werden können, dürfen elementare Bäume prinzipiell beliebige endliche Tiefe haben. Damit repräsentieren elementare Bäume einen größeren Lokalitätsbereich als kfG.

Die zentrale Operation zur Konstruktion komplexer Baumstrukturen ist die Adjunktion. Mittels Adjunktion kann aus einem auxilaren Baum β und einem Baum α ein neuer Baum γ bestimmt werden. γ wird dann als abgeleiteter Baum bezeichnet. Sei α ein Baum, der einen Knoten X beinhaltet und β ein auxiliarer Baum dessen Wurzel ebenfalls X bezeichnet. Abbildung 5 zeigt graphisch wie ein neuer Baum γ durch Adjunktion des auxiliaren Baumes β am Knoten X von α abgeleitet wird (w_i und v_i seien Teilstrings).

Ein Vorteil der Adjunktion ist, daß mit ihr die Rekursion von lokalen Abhängigkeiten (wie bei kfG der Fall) herausfaktorisiert werden kann. Es konnte gezeigt werden, daß TAGs stärkere Ausdrucksfähigkeit haben als kfG, da mit ihnen schwach kontext–sensitive Sprachen beschrieben werden können [Weir, 1988].

Ein Nachteil des ursprünglichen Formalismus war die sehr redundante Formulierung von Teilbäumen. Daher hat [Abeille, 1988] als zusätzliche Operation die Substitution eingeführt, die es erlaubt Teilbäume in elementaren Bäumen durch entsprechende Nichtterminale zu ersetzen. Damit sind nun auch Nichtterminale als Blattknoten erlaubt, die durch das Substitutionszeichen $\downarrow$ gesondert gekennzeichnet werden. Die extrahierten

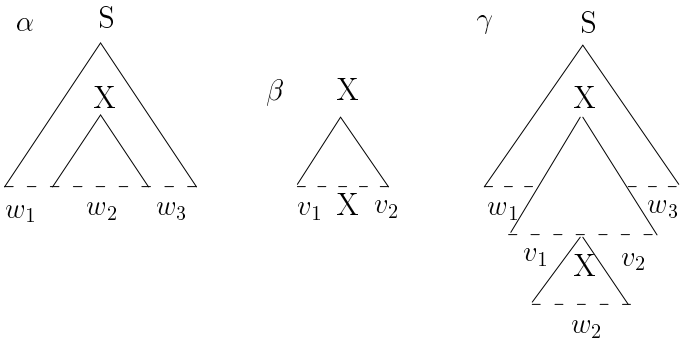

Abbildung 5: Die Adjunktion in TAG.

Teilbäume unterscheiden sich von initialen Bäumen nur dadurch, daß ihr Wurzelknoten ein beliebiges Nichtterminal bezeichnen kann. Sie werden als lexikalische Bäume bezeichnet. Es konnte gezeigt werden, daß die Hinzufügung der Substitution die formalen Eigenschaften von TAGs nicht beeinflußt, jedoch zu einer redundanzfreieren und eleganteren Beschreibung linguistischer Objekte führt. Abbildung 6 zeigt ein Beispiel für einen kompakteren initialen Baum und einen extrahierten Teilbaum. Die Substitution ist eine obligatorische Operation, d.h. daß ein Ableitungsbaum erst dann vollständig ist, wenn alle Substitutionen ausgeführt wurden.

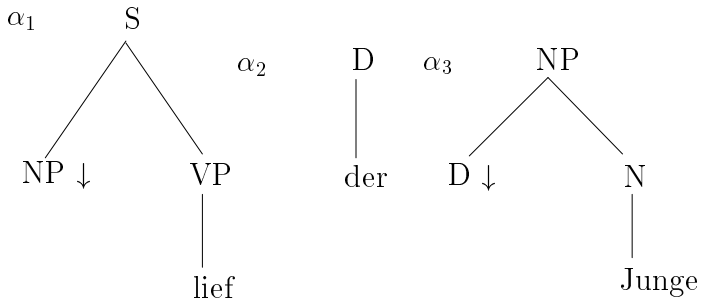

Abbildung 6: Initiale Bäume in lexikalisierten TAGs.

Relevanz von TAGs für die Generierung TAGs wurden in erster Linie zur Formulierung syntaktischer Strukturen entwickelt. Sie wurden bereits relativ früh auch in Generierungssystemen eingesetzt, namentlich in dem System MUMBLE–86 [McDonald und Pustejovsky, 1985]. [Joshi, 1987] geht speziell auf die Relevanz von TAGs für die Generierung ein. Die Kombination folgender Eigenschaften von TAGs ist für die Gener-

ierung bedeutsam: (1) größerer Lokalitätsbereich als bei kontextfreien Grammatiken, (2) Möglichkeit Dominanz und lineare Abfolge zu trennen, (3) Verwendung von Unifikation, (4) geeignet für inkrementelle Verarbeitung.

Wir wollen nun kurz zwei neuere Arbeiten vorstellen, die einige dieser Aspekte für die Generierung modellieren.

TAGs und systemische Grammatiken Die zentrale Idee des in [McCoy *et al.*, 1992] beschriebenen Ansatzes liegt in einer Kombination von systemischen Grammatiken und TAGs, wobei die systemische Grammatik als Mittel zur Beschreibung der funktionalen Beziehungen zwischen linguistischen Objekten und die TAG zur Beschreibung der Form der Objekte dient. Hierzu werden im TAG Formalimus die syntaktischen Strukuturen repräsentiert. Elementare syntaktische Bäume werden gemäß funktionaler Kategorien klassifiziert. Die gesamte Strukutur wird als TAG–Netzwerk bezeichnet. Eine systemische Grammatik wird eingesetzt, um von einer Eingabestruktur die Menge an funktionalen Merkmalen zu bestimmen, so daß mit dieser Information durch Traversierung des TAG–Netzwerkes die relevante Menge der elementaren Bäume bestimmt werden kann. Die so gewählten Bäume werden dann aufgrund ihrer funktionalen Beziehungen zusammengefügt. Tatsächlich sind in dem Verfahren die oben aufgeführten Schritte integriert. Ausgangspunkt für ein integriertes Verfahren ist die Head/Modifier Struktur der elementaren Bäume. Zur Darstellung dieser Beziehungen wird eine lexikalisierte TAG verwendet, in der Prädikate lexikalisiert und ihre zugeordneten Argumente durch nichtterminale Elemente repräsentiert sind (vgl. Abb. 6). Insgesamt ergibt sich damit ein an der Head/Modifier Struktur der Eingabe orientierter rekursiver Generierungsprozeß der systemischen Grammatik und des TAG–Netzwerkes.

Sei z.B. die Merkmalsstruktur in Abb. 7 Ausgangspunk für den Generierungsprozeß (in Anlehnung an [McCoy *et al.*, 1992]).[4] Diese Information ist Ausgangspunkt für die Traversierung der systemischen Grammatik. Ziel hierbei ist es, (a) die Head/Modifier Struktur zu bestimmen und (b) die Menge der funktionalen Merkmale, die für die Traversierung des TAG–Netzwerkes benötigt wird, zu extrahieren. In einem ersten Schritt wird zuerst das Head–Element berechnet (in unserem Beispiel *think*) und unter Berücksichtigung seiner Struktur wird der Prozeß dann rekursiv über den Argumenten (*you* und *hit*) fortgeführt. Für jeden auf diese Art determinierten Baum wird verwendete funktionale Information protokolliert. Insgesamt erhält man also eine an der funktionalen Information orientierte Struktur elementarer Bäume, die in einem letzten Schritt gemäß Substitution und Adjunktion zu einer Gesamtstruktur verknüpft werden.

Ein wesentlicher Vorteil des beschriebenen Verfahrens ist, daß in dieser Architektur funktionale Aspekte der Generierung mit syntaktischen Aspekten elegant zusam-

[4] Abb. 7 zeigt eine vereinfachte Darstellung der Eingabe. Im allgemeinen besteht die Eingabe aus einer Menge von Merkmalen, die zum Traveriseren der systemischen Grammatik verwendet wird.

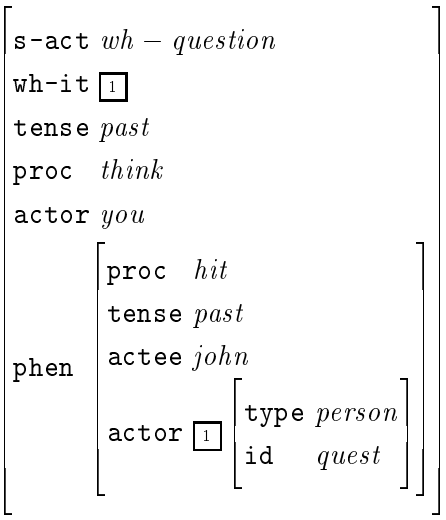

Abbildung 7: Eingabe für *Who did you think hit John?*

mengeführt werden können. Beide Aspekte werden dabei separat durch spezifische Formalismen repräsentiert, die in dem Verfahren aber gegenseitig beschränkend kombiniert werden.

Inkrementelle Generierung mit TAGs In [Harbusch *et al.*, 1991] wird der TAG Formalismus zur inkrementellen Generierung eingesetzt. Im allgemeinen wird inkrementelle Generierung als Voraussetzung für das Erzeugen von *spontaner* Sprache betrachtet. Inkrementelle Generierung ist durch zahlreiche Untersuchungen psychologisch motiviert und in [DeSmedt und Kempen, 1987] wurde erstmals ein psychologisch motiviertes inkrementelles Generierungssystem vorgestellt, wobei allerdings nur die Inhaltsrealisierung exakt modelliert wurde. [Reithinger, 1991] stellt ein inkrementelles System dar, in dem auch eine inkrementelle Verarbeitung auf der Ebene der Inhaltsbestimmung realisiert ist. Reithinger behauptet jedoch keine psychologische Adäquatheit seines Modells.

In [Kempen, 1987] und [Neumann und Finkler, 1990] werden als Voraussetzungen für eine inkrementelle Inhaltsrealisierung folgende Punkte aufgeführt: (1) Verwendung lexikonzentrierter Grammatiken, (2) explizite Trennung von Dominanz und linearer Abfolge, (3) Möglichkeit der Reformulierung, (4) lokale Entscheidungskriterien und (5) Operationen zur Einfügung, sowie top-down und bottom-up Expansion syntaktischer Strukturen.

In dem inkrementellen Generierungssystem von [Harbusch *et al.*, 1991] wird eine lexikalisierte LD/LP-UTAG verwendet, wobei LD/LP für 'Local Dominance/Linear Presedence' und U für Unifikation steht. In diesem Formalismus werden elementare

Bäume als Mobile interpretiert, wobei die Menge der möglichen Permutationen durch lineare Präzedenzregeln beschränkt werden (z.B. erlaubt die Regel $det < adj$ nur präterminale Folgen wie *det adj n* oder *n det adj*, aber nicht eine Folge wie *adj det n*). Die Unifikation in TAGs erlaubt es, elementare Bäume mit Merkmalsbeschränkungen zu annotieren. Damit können neben Angaben zur Kongruenz auch semantische Restriktionen mit den syntaktisch-orientierten Bäumen assoziiert werden.

Die Verwendung einer lexikalisierten TAG wird damit motiviert, daß zumindest in einem inkrementellen Modell die syntaktische Verarbeitung am besten lexikalisch gesteuert werden sollte [Levelt, 1989]. Es wird angenommen, daß der inhaltsbestimmende Teil als Eingabestruktur für den Realisierungsteil eine semantisch orientierte Funktor/Argument-Struktur der zu erzeugenden Äußerung bestimmt. Es gilt nun, für diese Struktur mittels Wortwahl entsprechende lexikalische Einträge zu finden. Jeder einzelne Eintrag ist dann Ausgangspunkt für die Suche 'seiner' zugeordneten elementaren Bäume. Als Eingabe für die Realisierung mit TAGs im System von [Harbusch *et al.*, 1991] werden Lemmata und funktionale Beziehungen zwischen ihnen erwartet. Nehmen wir als Beispiel die lexikalisierten Bäume aus Abb. 6, dann wären *Junge* und *lief* mögliche Lemmata und *agens(lief) = Junge* eine funktionale Beziehung. Die einzelnen Bäume können mittels der funktionalen Beziehung kombiniert werden, wobei Realisierungen weiterer Lemmata durch Adjunktion eingefügt werden. Die so kombinierten Strukturen werden in einem nächsten Schritt linearisiert und flektiert.

Die Besonderheit in einem inkrementellen Generierungssystem liegt darin, daß bereits Teilstrukturen zur Verbalisierung zur Verfügung gestellt werden, bevor die gesamte Struktur berechnet ist. Um spontane Generierung zu ermöglichen, wird versucht, für diese partielle Information (oder *Segmente*) eine lokal vollständige syntaktische Struktur zu erzeugen, damit diese in der nachfolgenden morphologischen Komponente flektiert und sofort geäußert werden kann. Es hängt jedoch vom Zeitpunkt der eintreffenden Lemmata ab, ob die Strukturen tatsächlich unmittelbar geäußert werden können. Nehmen wir in obigen Beispiel an, daß *Junge* vor *lief* eintrifft. Das Problem mit dieser Reihenfolge ist, daß *Junge* nicht spontan geäußert werden kann, da für seine Flektion die Kasusinformation notwendig ist, diese aber vom Verb bestimmt wird. Prinzipiell gibt es drei Möglichkeiten, auf das Problem einzugehen: (a) Verwendung von Default-Werten [DeSmedt und Kempen, 1987], (b) Information explizit anfordern [Finkler und Neumann, 1989] oder warten, bis das Verb bekannt ist [Harbusch *et al.*, 1991]. Der Vorteil in (a) ist, daß spontane Generierung realisiert wird. Der gravierende Nachteil ist jedoch, daß im weiteren Verlauf des Generierungsprozesses möglicherweise eine Reihe von Korrekturen notwendig werden, da die gemachten Annahmen durch neue Information revidiert werden müssen. Dies wird in (b) vermieden, hat allerdings zum Nachteil, daß nur eine verzögerte spontane Äußerung möglich ist. Der Vorteil von (b) zu (c) liegt darin, daß bei (b) nicht unnötig lange gewartet werden muß, da die fehlende Information

direkt angefordert wird.

Welcher der aufgeführten Ansätze zu wählen ist, hängt sicherlich auch mit der Frage der Modellierung der Größe der Segmente zusammen, also ob z.B. jedes einzelne Adjektiv einer komplexen Nominalphrase geäußert werden soll, bevor das Nomen bekannt ist, oder ob die gesamte Nominalphrase als ganzes geäußert werden sollte.

Kurzes Fazit Bevor wir nun im nächsten Abschnitt den Aspekt der Reversibilität einführen, ein kurzes Fazit. Constraintbasierte Formalismen (CG) und TAGs eignen sich besonders zur Repräsentation des syntaktischen Wissens, wobei in CGs darüber hinaus die Relation zwischen Syntax und Semantik betont wird. In systemischen Grammatiken werden die funktionalen Beziehung zwischen linguistischen Objekten in den Vordergund gestellt. In [McCoy *et al.*, 1992] wird ein Ansatz vorgestellt, wie systemische Grammatiken mit TAGs in Beziehung gebracht werden können. Ein Beispiel, wie systemische Grammatiken mit CGs kombiniert werden können, findet sich in [Bateman *et al.*, 1992]. Welcher der eingeführten Formalismen tatsächlich am geeignetsten für die Generierung ist, läßt sich abschließend nicht sagen. Es könnte sein, daß eine Kombination, wie sie oben aufgeführt ist, den praktischen Gegebenheiten am nächsten kommt. Ein weiteres Beurteilungskriterium kann aber auch der Aspekt der Reversibilität sein.

Reversibilität

Die Idee einer reversiblen Grammatik, d.h. die Verwendung ein und derselben Grammatik für die Analyse und Produktion von Sätzen, wurde schon oft als erstrebenswert skizziert (s. [Neumann, 1991a] für einen Überblick). Jedoch erst seit wenigen Jahren werden reversible Grammatiken konkret erforscht, fast ausschließlich in der Computerlinguistik. Ausgangspunkt für die Entwicklung reversibler Grammatiken sind die constraintbasierten Formalismen. Da mit ihnen der Anspruch erhoben wird, grammatikalisches Wissen deklarativ zu formulieren, sollten sie prinzipiell in beiden Richtungen einsetzbar sein. Es hat sich aber relativ schnell gezeigt, daß Grammatiken oder Formalismen, die bisher hauptsächlich für eine Richtung verwendet wurden, nicht ohne Probleme auch in der anderen Richtung eingesetzt werden können. Am Beispiel der Lexical Functional Grammar (LFG, [Bresnan, 1982]) wollen wir dies kurz erläutern. In LFG werden Sätze einer natürlichen Sprache durch eine kontextfreie Grammatik beschrieben (c-Struktur), deren Kategorien mit einer Menge von Gleichungen annotiert sind. Diese definieren eine Abbildung auf eine funktionale Ebene (f-Struktur), die die Funktor-Argument Beziehungen expliziert. In dem ursprünglichen Ableitungsbegriff wurde davon ausgegangen, daß zuerst die c-Struktur eines Satzes bestimmt werden muß, bevor durch Auswerten der funktionalen Gleichungen die f-Struktur berechnet werden kann. Dieses Vorgehen ist jedoch für die Generierung nicht adäquat, da i.a. mit einer f-Struktur die

Generierung begonnen werden sollte, eine direkte Beziehung zwischen f-Struktur und c-Struktur aber nicht formuliert ist. Daher müßte zuerst eine c-Struktur konstruiert werden, bevor ihre f-Struktur mit der Eingabe verglichen werden kann. Aus diesen Gründen schien LFG besser geeignet für Parsing als für Generierung [Block, 1987]. Wedekind konnte aber in [Wedekind, 1988; Wedekind, 1991] zeigen, daß eine Umformulierung des Ableitungsbegriffes möglich ist, so daß Parsing und Generierung gleichermaßen effektiv durchgeführt werden können, ohne aber wesentliche Eigenschaften der Theorie zu beeinflussen. Die zentrale Idee hier ist die simultane Beschreibung von c-Struktur und f-Struktur. Damit kann in jedem Schritt der Ableitung überprüft werden, ob die f-Struktur der annotierten Gleichungen kohäherent mit der entsprechenden Beschreibung in der Eingabe ist. Dieses Verfahren setzt eine Top/Down Strategie für die Verabeitung voraus.

Typen von reversiblen Systemen Die aktuellen Arbeiten auf dem Gebiet der reversiblen Grammatiken lassen sich grob in drei Typen unterteilen:

Typ A Kompilation spezifischer Parsing- und Generierungsgrammatiken aus einer Grammatik

Typ B Verwendung einer Grammatik, aber unterschiedliche Prozesse

Typ C Verwendung einer Grammatik und ein uniformer Prozess

In Systemen von Type A wird das linguistische Wissen für Parsing und Generierung in einer gemeinsamen Grammatik formuliert, die für den Laufzeitmodus des System in eine spezielle Parsing- und Generierungsgrammatik kompiliert wird. Der Vorteil dieser Methode ist, daß die Quellgrammatik für den jeweiligen Einsatz gesondert 'getunt' werden kann, ein großer Nachteil liegt darin, daß während der Laufzeit das grammatikalische Wissen im Gesamtsystem redundant vorliegt. Ansätze, die dieser Methode folgen, sind z.B. [Block, 1991] und [Dymetman *et al.*, 1990].

Dieser Nachteil wird in Systemen von Typ B vermieden, da Parsing und Generierung auf derselben Grammatik operieren. Eine weiterer Vorteil ist, daß die Grammatik einfacher getestet und verändert werden kann. Ein Nachteil dieser Methode ist, daß z.Z. die verwendeten Verfahren zu ineffizient sind.

Systeme vom Typ C verfolgen den radikalsten Ansatz von Reversibilität: Nicht nur eine gemeinsame Grammatik wird benutzt, sondern auch derselbe Basisprozeß. Eine der ersten uniformen Architekturen ist in [Shieber, 1988] beschrieben und folgt dem Paradigma 'Linguistische Verarbeitung als Deduktion'; Shieber verwendet als gemeinsamen Basisprozeß eine Variante des Early-Algorithmus. In [Emele und Zajac, 1990] wird eine uniforme Architektur vorgestellt, die auf dem Paradigma 'Linguistische Verarbeitung als Typeninferenz' beruht. Sie verwendet als einzige Verarbeitungsstrategie

die vollständige Typexpansion durch Unifikation. Der größte Nachteil beider Ansätze ist ihre erhebliche Ineffizienz. Daher sind sie für einen realistischen Einsatz in einem NLG (zumindest gegenwärtig) nicht geeignet.

Zur Effizienzsteigerung der letzten beiden Typen von reversiblen Systemen ist in letzter Zeit jedoch ein Reihe von Vorschlägen gemacht worden, u.a. effiziente Indizierungstechniken für das Lexikon oder der Einsatz von verzögerter Typexpansion. [Uszkoreit, 1991] schlägt vor, Kontrollinformation in Form von Präferenzen an Merkmalsstrukturen zu annotieren. Diese Kontrollinformation kann verwendet werden, um die Reihenfolge der Abarbeitung von Konjunkten oder Disjunkten zu steuern, oder um Information auszublenden. Prinzipiell wäre es möglich, unterschiedliche Präferenzsysteme für Parsing und Generierung zu etablieren, um so spezifische Kontrollaspekte zu formulieren.

Integration von Parsing und Generierung In den meisten reversiblen Ansätzen wird grammatikalische Verarbeitung unabhängig vom Diskurskontext einer Äußerung verarbeitet. In [Appelt, 1989] und [Neumann, 1991b] wird jedoch gezeigt, daß eine strikte Verwendung reversibler Grammatiken einen erheblichen Einfluß auf das Design eines NLG zur Folge hat, insbesondere ergibt sich hier das Problem der Paraphrasenauswahl. Im allgemeinen kann in einem NLG nicht kontrolliert werden, ob und in welchem Grad der erzeugte Oberflächenstring ambig ist. Beispielsweise ist der Satz *Lösche den Ordner mit den Systemfiles!* ambig, da nicht klar ist, ob der Ordner mit Hilfe der Systemfiles gelöscht werden soll oder der Ordner, der die Systemfiles beinhaltet. Daher besteht für ein NLG das Risiko, aufgrund von Mehrdeutigkeiten mißverstanden zu werden. In [Neumann und van Noord, 1992] wird zur Lösung dieses Problems eine Methode vorgeschlagen, die auf einer strikten Integration von Parsing und Generierung unter Verwendung einer reversiblen Grammatik beruht. Diese enge Verzahnung ermöglicht es, den Parser zur Unterstützung der Generierung einzusetzen und umgekehrt. Während der Generierung kann Parsing z.B. eingesetzt werden, um relevante Quellen von Ambiguität festzustellen, die dann in einem Revisionsschritt aufgelöst werden können.

Abschließende Bemerkungen

In diesem Aufsatz wurde ein Überblick gegenwärtig verwendeter Grammatikformalismen gegeben. Anhand ausgewählter Beispiele wurde gezeigt, wie systemische Grammatiken (SG), constraintbasierte Formalismen (CG) und Baumadjunktionsgrammatiken (TAG) in der Generierung eingesetzt werden.

Dieser Aufsatz kann natürlich nur einen Einblick in den Aspekt der Inhaltsrealisierung geben, insbesondere konnten aus Platzgründen so spannende Themen wie Wortwahl oder die Verwendung von Grammatikenformalismen in der maschinellen Übersetzung nicht problematisiert werden. Ich hoffe jedoch gezeigt zu haben, daß 'Grammatikfor-

malismen in der Generierung' immer noch ein heißes Thema ist, insbesondere wenn der Aspekt der Reversibilität ins Spiel kommt.

Literatur

[Abeille, 1988] A. Abeille. Parsing french with tree adjoining grammar: some linguistic accounts. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING)*, Budapest, 1988.

[Appelt, 1985] D. E. Appelt. *Planning English Sentences.* Cambridge University Press, Cambridge, 1985.

[Appelt, 1989] D. E. Appelt. Bidirectional grammars and the design of natural language generation systems. In Y. Wilks, editor, *Theoretical Issues in Natural Language Processing 3*, pages 206–212. Hillsdale, N.J.: Erlbaum, 1989.

[Bateman *et al.*, 1992] J. A .Bateman, M. Emele, und S. Momma. The nondirectional representation of systemic functional grammar and semantics as typed feature structure. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, Nantes, 1992.

[Block, 1987] R. Block. Can a 'parsing grammar' be used for natural language generation? the negative example of lfg. In *Proceedings of the first European Natural Language Generation Workshop*, Abbey de Royaumont, 1987.

[Block, 1991] H. U. Block. Compiling trace & unification grammar for parsing and generation. In *Proceedings of the ACL Workshop Reversible Grammars in Natural Language Processing*, Berkeley, 1991.

[Bresnan, 1982] J. Bresnan, editor. *The Mental Representation of Grammatical Relations.* MIT Press, 1982.

[DeSmedt und Kempen, 1987] K. DeSmedt und G. Kempen. Incremental sentence production, self–correction and coordination. In G. Kempen, editor, *Natural Language Generation*, pages 365–376. Martinus Nijhoff, Dordrecht, 1987.

[Dymetman *et al.*, 1990] M. Dymetman, P. Isabelle, und F. Perrault. A symmetrical approach to parsing and generation. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING)*, pages 90–96, Helsinki, 1990.

[Elhadad, 1989] M. Elhadad. Extended functional unification programmers. Technical Report Technical Report No. CUC-420-89, Department of Computer Science, Columbia University, 1989.

[Emele und Zajac, 1990] M. C. Emele und R. Zajac. Typed unification grammars. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING)*, pages 293–298, Helsinki, 1990.

[Finkler und Neumann, 1989] W. Finkler und G. Neumann. Popel-how: A distributed parallel model for incremental natural language production with feedback. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1518–1523, Detroit, 1989.

[Gazdar *et al.*, 1985] G. Gazdar, E. Klein, G. Pullum, und I. Sag. *Generalized Phrase Structure Grammar*. Blackwell, 1985.

[Grosz *et al.*, 1986] B. Grosz, K. Sparck Jones, und B. L. Webber, editors. *Readings in Natural Language Processing*. Morgan Kaufmann, 1986.

[Halliday, 1985] M. A. K. Halliday. *An Introduction to Functional Grammar*. London: Edward Arnold, 1985.

[Harbusch *et al.*, 1991] K. Harbusch, W. Finkler, und A. Schauder. Incremental syntax generation with tree adjoining grammars. Technical report, DFKI Saarbrücken, 1991. RR-91-25.

[Hovy, 1987] E. H. Hovy. *Generating Natural Language under Pragmatic Constraints*. PhD thesis, Yale University, 1987.

[Joshi *et al.*, 1975] A.K. Joshi, L.S. Levy, und M. Takahashi. Tree adjunct grammars. *Journal Computer Systems Science*, 10(1), 1975.

[Joshi, 1987] A. K. Joshi. The relevance of tree adjoining grammar to generation. In G. Kempen, editor, *Natural Language Generation*, pages 233–252. Martinus Nijhoff, Dordrecht, 1987.

[Kay, 1975] M. Kay. Syntactic processing and functional sentence perpective. In R. Schank und B.L. Nash-Webber, editors, *Theoretical Issues in Natural Language Processing*, 1975.

[Kay, 1984] M. Kay. Functional unification grammar: A formalism for machine translation. In *Proceedings of the 10th International Conference on Computational Linguistics and the 22nd Annual Meeting of the Association for Computational Linguistics (COLING)*, pages 75–78, Standford, 1984.

[Kempen, 1987] G. Kempen. A framework for incremental syntactic tree formation. In *Tenth IJCAI*, pages 655–660, Mailand, 1987.

[Levelt, 1989] W. J. M. Levelt. *Speaking: From Intention to Articulation.* MIT Press, Cambridge, Massachusetts, 1989.

[Matthiesen, 1985] C. Matthiesen. The systemic framework in text generation: Nigel. In J. Benson und W. Greaves, editors, *Systemic Perspectives on Discourse*, volume 1. Ablex, Norwood NJ, 1985.

[McCoy *et al.*, 1992] K. F. McCoy, K. Vijay-Shnaker, und G. Yang. A functional approach to generation with tag. In *30th Annual Meeting of the Association for Computational Linguistics*, Newark, Delaware, 1992.

[McDonald und Pustejovsky, 1985] D. D. McDonald und J. D. Pustejovsky. Description–directed natural language generation. In *Ninth IJCAI*, pages 799–805, Los Angeles, 1985.

[McKeown *et al.*, 1990] K. R. McKeown, M. Elhadad, Y. Fukomoto, J. Lim, C. Lombardi, J. Robin, und F. Smadja. Natural language generation in comet. In Robert Dale, Chris Mellish, und Michael Zock, editors, *Current Research in Natural Language Generation*, pages 103 – 139. Academic Press, London, 1990.

[McKeown, 1985] K. R. McKeown. *Text Generation: Using Discourse Strategies und Focus Constraints to Generate Natural Language Text.* Cambridge University Press, Cambridge, 1985.

[Meteer *et al.*, 1987] M. W. Meteer, D. D. McDonald, S. D. Anderson, D. Forster, L. S. Gay, A. K. Huettner, und P. Silbun. Mumble-86: Design and implementation. Technical Report COINS Technical Report 87-87a, University of Massachusetts at Amherst, 1987.

[Neumann und Finkler, 1990] G. Neumann und W. Finkler. A head-driven approach to incremental and parallel generation of syntactic structures. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING)*, pages 288–293, Helsinki, 1990.

[Neumann und van Noord, 1992] Günter Neumann und Gertjan van Noord. Self-monitoring with reversible grammars. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, Nantes, 1992.

[Neumann, 1991a] G. Neumann. A bidirectional model for natural language processing. In *Fifth Conference of the European Chapter of the Association for Computational Linguistics*, pages 245–250, Berlin, 1991.

[Neumann, 1991b] G. Neumann. Reversibility and modularity in natural language generation. In *Proceedings of the ACL Workshop on Reversible Grammar in Natural Language Processing*, pages 31–39, Berkeley, 1991.

[Pereira und Warren, 1980] F. C.N. Pereira und D. Warren. Definite clause grammars for language analysis - a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13, 1980. reprinted in [Grosz *et al.*, 1986].

[Pollard und Sag, 1987] C. Pollard und I. A. Sag. *Information Based Syntax and Semantics, Volume 1.* Center for the Study of Language and Information Stanford, 1987.

[Reithinger, 1991] N. Reithinger. Popel: A parallel and incremental natural language generation system. In C. L. Paris et al., editor, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pages 179–199. Kluwer, 1991.

[Shieber und Schabes, 1990] S. M. Shieber und Y. Schabes. Synchronous tree–adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING)*, Helsinki, 1990.

[Shieber *et al.*, 1983] S. M. Shieber, H. Uszkoreit, F. C.N. Pereira, J. Robinson, und M. Tyson. The formalism and implementation of PATR-II. In B. J. Grosz und M. E. Stickel, editors, *Research on Interactive Acquisition and Use of Knowledge.* SRI report, 1983.

[Shieber *et al.*, 1991] S. M. Shieber, F. C. N. Pereira, G. van Noord, und R. C. Moore. Semantic-head-driven generation. *Computational Linguistics*, 16:30–42, 1991.

[Shieber, 1986] S. M. Shieber. *Introduction to Unification-Based Approaches to Grammar.* Center for the Study of Language and Information Stanford, 1986.

[Shieber, 1988] S. M. Shieber. A uniform architecture for parsing and generation. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING)*, Budapest, 1988.

[Uszkoreit, 1991] H. Uszkoreit. Strategies for adding control information to declarative grammars. In *29th Annual Meeting of the Association for Computational Linguistics*, Berkeley, 1991.

[Uszkoreit, 1986] H. Uszkoreit. Categorial unification grammar. In *Proceedings of the 11th International Conference on Computational Linguistics (COLING)*, Bonn, 1986.

[Wedekind, 1988] J. Wedekind. Generation as structure driven derivation. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING)*, Budapest, 1988.

[Wedekind, 1991] J. Wedekind. Unifikationsgrammatiken und ihre logik. Technical report, Arbeitspapiere des Sonderforschungsbereichs 340, 1991. Bericht Nr. 8.

[Weir, 1988] D. Weir. *Characterizing mildly context-sensitive grammar formalisms.* PhD thesis, University of Pennsylvania, 1988.

[Winograd, 1983] T. Winograd. *Language as a Cognitive Process.* Reading, Mass.: Addision-Wesley, 1983.