Mechanismen einer kontextfreien Grammatik für das Deutsche

Studentischer Beitrag zur 18. Deutschen Jahrestagung für Künstliche Intelligenz (KI-94)
Saarbrücken, 18. – 23. September 1994

Sascha Brawer · Stengelstraße 18 · D – 66117 Saarbrücken · e-Mail: brawer@coli.uni-sb.de

Zusammenfassung

Dieser Artikel stellt eine Grammatik für ein deutschsprachiges Text-to-Speech-System vor. Bei der Sprachsynthese ist eine kurze Verarbeitungszeit zwar von großer Bedeutung, die Praxis zeigt aber, daß man mit einer allzu oberflächlichen linguistischen Analyse kein hochwertiges Sprachsignal erzeugen kann — es ist also von Vorteil, auch für die Umwandlung von geschriebener zu gesprochener Sprache mindestens eine syntaktische Verarbeitung vorzunehmen. Die Computerlinguistik arbeitet heute überwiegend mit Unifikationsgrammatiken, um natürliche Sprache syntaktisch zu verarbeiten. Um Rechenzeit einzusparen, beschränkte man sich beim hier besprochenen System allerdings auf die Unifikation von atomaren Termen; es wird (informell) gezeigt, daß dieser Verzicht unmittelbar zur Folge hat, nur noch die Klasse der kontextfreien Sprachen verarbeiten zu können.
Diese Einschränkung erscheint sehr schwerwiegend, entgegen einer verbreiteten Meinung ist es aber durchaus möglich, mit einer kontextfreien Grammatik auch komplexere sprachliche Phänomene handzuhaben. Der Hauptteil dieses Artikels zeigt Mechanismen, mit denen auch mit einer auf atomare Unifikation beschränkten und somit kontextfreien Grammatik Koordination, Subkategorisierung, Fernabhängigkeiten und freie Wortstellung behandelbar sind.

Einleitung

Im Laufe des Jahres 1993 hatte ich während dreier Monate die Gelegenheit, am Text-to-Speech-System SVOX verschiedene Arbeiten auszuführen; dieses System wurde von der Gruppe für Sprachverarbeitung an der Eidgenössischen Technischen Hochschule Zürich (ETH) entwickelt.
Eine ausführliche Beschreibung des Systems findet sich in [Traber 1993]; hier kann lediglich ein grober Überblick gegeben werden. SVOX soll aus einem geschriebenen, grammatikalisch korrekten deutschsprachigen Text ein hochwertiges Sprachsignal für Telefoniezwecke erzeugen. Wegen der großen qualitativen Anforderungen baut es auf einer für ein Sprachsynthesesystem recht ausgefeilten morphologischen und syntaktischen Analyse auf: Mit Hilfe einer Wortgrammatik, eines Vollformen- und eines Morphemlexikons werden die einzelnen Wörter geparst, bevor die Struktur des ganzen Satzes anhand einer Satzgrammatik untersucht wird. Ausgehend vom resultierenden Syntaxbaum bestimmen Prolog-Regeln Betonung und Phrasierung (Sprechgruppeneinteilung). Die Satzmelodie erzeugt ein neuronales Netz, Lautdauer und Sprechrhythmus werden mittels eines linearen statistischen Modells generiert. Abschließend erfolgt die Synthese des Sprachsignals durch Diphon-Konkatenation.

Ursprünglich lagen diese Grammatiken als Transitionsnetzwerke vor, die um Unifikation erweitert waren. Auf diese sogenannten »UTNs« (vgl. [Russi 1990]) wandte man ein an [Pereira/Warren 1980] angelehntes Verfahren an, um vollautomatisch Definit-Klausel-Grammatiken zu erhalten, wie sie auch zum Beispiel die Programmiersprache Prolog kennt. Obwohl DCGs sehr oft direkt in Prolog abgearbeitet werden, wird in SVOX aus Gründen der Effizienz ein in Modula-2 geschriebener Chart-Parser eingesetzt.
Meine Aufgabe war es nun einerseits, die Grammatiken zu verbessern, andererseits sollte die Syntaxanalyse beschleunigt werden. Die Änderungen am Parser sind jedoch nicht Gegenstand dieses Artikels: Im wesentlichen habe ich den Earley-Algorithmus implementiert und durch Techniken wie Überprüfung der First-/Follow-Relationen erweitert, die aus dem Compilerbau bekannt sind.
Es erwies sich als nötig, große Teile der Grammatik neu zu schreiben, wobei ich großes Gewicht auf die Verarbeitungszeit legte. Das Aufwendigste bei einer Unifikationsgrammatik ist das Unifizieren selber — es lohnt sich also, hier anzusetzen, bei dieser Operation, die unter Umständen pro Satz mehrere tausend Mal durchgeführt werden muß. Nun ist es vor allem das Unifizieren von *komplexen* Termen, das viel Zeit in Anspruch nimmt; um zwei Atome miteinander zu unifizieren, braucht es dagegen nicht viel mehr als einen simplen Vergleich.
Es liegt aus diesem Grund nahe, eine Grammatik zu schreiben, die sich mit atomarer Unifikation begnügt. Wie im nächsten Abschnitt gezeigt wird, kann eine solche Grammatik jedoch nur die Klasse der *kontextfreien Sprachen* verarbeiten. Kontextfreie Grammatiken sind nun allerdings in großen Kreisen der Computerlinguistik eher verpönt (vgl. zum Beispiel die Argumente gegen kontextfreie Grammatiken in [Haugeneder/Trost 1993]): Es heißt unter anderem, mit ihnen ließen sich komplexere linguistische Phänomene kaum beschreiben. Daß man mit kontextfreien Grammatiken durchaus einige Tiefe bei der linguistischen Analyse erreichen kann, soll der Rest dieses Artikels zeigen: Nach dem Einführen von zwei nützlichen Erweiterungen des Formalismus, die auf die Sprachklasse aber keinerlei Einfluß haben, wird erläutert, wie die SVOX-Satzgrammatik mit Koordination, Subkategorisierung und freier Wortstellung umgeht.
Es sei aber bereits an dieser Stelle angemerkt, daß im beschriebenen System keine semantische Verarbeitung stattfindet. In diesem Fall wäre eine »richtige« Unifikationsgrammatik, HPSG zum Beispiel, sicherlich ein geeigneteres Werkzeug als ein reiner DCG-Formalismus.
Weiterhin sei festgestellt, daß die nachfolgend vorgestellte Grammatik keinerlei theoretische Ansprüche erheben will: SVOX ist ein Programmpaket, das in der Praxis angewandt werden soll; es kann daher manche Frage einfach ignorieren, die man einer Grammatik-*Theorie* stellen müßte.

Warum ist die Grammatik kontextfrei?

Wie bereits erwähnt wurde, benutzt das System SVOX sowohl für die Darstellung des syntaktischen wie auch des morphologischen Wissens Grammatikregeln, die abgesehen von einer etwas anderen Syntax den Definit-Klausel-Grammatiken (DCGs) von Prolog entsprechen.
Mit Definiten Klauseln können durchaus auch kontext-*sensitive* Sprachen repräsentiert werden. Es sei hier lediglich auf [König/Seiffert 1989] verwiesen, wo sich auf Seite 112 eine

DCG für die kontextsensitive Sprache $a^n\ b^n\ c^n$ findet.[1] Obwohl der Parser von SVOX auch komplexe Terme miteinander unifizieren kann, also dieselbe Mächtigkeit wie der ursprüngliche DCG-Formalismus besitzt, beschränkte man sich beim Entwickeln der Grammatiken auf atomare Werte für die Attribute.

Diese Einschränkung hat schwerwiegende theoretische Konsequenzen: Man beschränkt sich damit auf die Klasse der kontextfreie Sprachen. Der Beweis soll hier lediglich skizziert werden; in der Einleitung wurde begründet, weshalb man bereit war, einen auf den ersten Blick so schwerwiegenden Einschnitt hinzunehmen.

Sei A die — endliche — Menge aller voneinander unterscheidbarer Atome, die in der zu parsenden Grammatik vorkommen. Jede instantiierte Variable muß einen Wert aus A besitzen, wenn man sich auf atomare Werte beschränkt. Man kann nun eine beliebige Regel R aus der Grammatik nehmen, die eine Variable v enthält, und für jeden Wert a, den v annehmen könnte, eine modifizierte Regel zur Grammatik hinzufügen, in der alle Vorkommen von v durch a ersetzt wurden. Formal: Man entfernt eine beliebige Regel R mit Vorkommen einer Variablen v aus der Grammatik und fügt stattdessen für jedes $a \in A$ eine neue Regel der Form $R\ [v/a]$ zur Grammatik hinzu. Dies wird so lange wiederholt, bis keine der Regeln mehr eine Variable enthält.

Das Ergebnis eines solchen Ersetzungsschritts ist zur ursprünglichen Regel äquivalent, weil man mittels Unifikation nicht feststellen kann, ob eine Variable v vorkommt oder ob für v stattdessen alle möglichen Werte zugelassen sind: Womit auch immer man v unifiziert, das Resultat der Unifikation wird in beiden Fällen dasselbe sein. Somit ist die neue Grammatik äquivalent zur ursprünglichen, von der man ausgegangen war.

Man hat nun eine Grammatik erhalten, die vollkommen variablenfrei ist; alle Literale haben somit die Form $L\ (a_1, \ldots, a_n)$ mit $a_i \in A$. Durch Umbenennen in $L\$a_1\$\ldots\$a_n$ ($ sei ein neues Zeichen) erhält man eine äquivalente kontextfreie Grammatik.

Erweiterung des Formalismus

Abgesehen von einer etwas anderen Syntax wurde der Standard-DCG-Formalismus einerseits aus den genannten Gründen auf atomare Werte der Attribute eingeschränkt; andererseits wurden zwei Erweiterungen vorgenommen, die auf die verarbeitete Sprachklasse allerdings keinen Einfluß haben.

Das SVOX-System soll in jedem Fall eine Ausgabe liefern, die einigermaßen vernünftig ist, auch wenn der Satz nicht der Grammatik entspricht: Die Grammatik muß daher eher großzügig sein, was die erkannten Konstruktionen betrifft; sie ist in hohem Maße übergenerierend. Es ist eine Illusion, zu erwarten, lediglich echte Ambiguitäten als Ausgabe zu erhalten; das SVOX-System liefert bei manchen Sätzen mehrere hundert Alternativen, von denen manche tatsächlich ambig sind, manche exotische Lesarten, manche aber auch schlicht falsch. Für die Sprachsynthese reicht es aber aus, wenn die Syntax *mehr oder weniger* korrekt analysiert wird; weitaus schlimmer wäre es, viele Sätze überhaupt nicht parsen zu können.

[1] Die dort beschriebene Grammatik »merkt« sich die Anzahl der aufgetretenen a's, b's und c's als Zahlen und unifiziert sie miteinander, wodurch sichergestellt ist, daß alle gleich häufig vorkommen. Für jedes Auftreten wird der Wert eines Attributs von eingebettetem Prolog um 1 erhöht. Man kann sich jedoch auch eine Behandlung denken, welche die Zahlen als Länge einer Liste kodiert und so ohne explizite Prolog-Befehle auskommt.

Aus diesem Grund möchte man bestimmte Konstruktionen als wahrscheinlicher als andere einstufen. Die eine Erweiterung des Formalismus ist daher eine Bewertung jeder Regelanwendung in Form von »Strafpunkten«, die über die gesamte Ableitung hinweg aufsummiert werden. Die Ableitung mit der niedrigsten Punktzahl gilt als wahrscheinlichste Alternative und wird als einzige den nachfolgenden Komponenten übergeben.
Die zweite Erweiterung sind als »unsichtbar« markierte Regeln. Sie werden ganz normal verarbeitet, ihre Anwendung erscheint aber nicht im Baum, der an die anderen Module weitergegeben wird.
Nachfolgend werden einige konkrete Anwendungen dieser beiden Erweiterungen gezeigt, die für die Behandlung von Koordination, Subkategorisierung und freier Wortstellung hilfreich sind.

Pseudo-Operatoren

Das Einführen von echten Operatoren bzw. von Funktionen, wie es beispielsweise in HPSG möglich ist, hätte einen gewissen Aufwand bedeutet; insbesondere wäre es nötig geworden, den Modula-Quelltext des Parsers zu ändern, um neue Funktionen zu implementieren — was einerseits sehr unschön wäre und andererseits eine gewisse Vermischung der Wissensrepräsentation mit der Verarbeitung bedeuten würde. Durch die Möglichkeit, eine Regelanwendung nicht im Syntaxbaum erscheinen zu lassen, kann man jedoch »Pseudo-Operatoren« deklarativ einführen, welche zu einem gewissen Grad dieselbe Funktionalität bieten, ohne eine echte Erweiterung des Formalismus vornehmen zu müssen. Ich möchte das Prinzip nachfolgend am Beispiel des logischen »Oder« zeigen.
Für das Erzeugen der Satzmelodie ist es von grundlegender Bedeutung, zu wissen, ob der Satz eine Frage oder eine Aussage darstellt. Taucht irgendwo im Satz eine Konstituente auf, die durch bestimmte Wörter oder durch die Wortstellung auf eine Frage hindeutet, soll der gesamte Satz als Frage betrachtet werden. Die oberste Konstituente trägt daher ein Attribut »Frage«, dessen Wert von unten nach oben »vererbt« wird.[2] Dazu wird ein »Operator« benötigt, der das boolesche »Oder« berechnet. Diese »Funktion« kann folgendermaßen notiert werden:[3]

```
or(f, f, f).                          0 Strafpunkte · unsichtbar
or(t, _, t).                          0 Strafpunkte · unsichtbar
or(_, t, t).                          0 Strafpunkte · unsichtbar
```

Ein Beispiel für die Benutzung:

```
s(F) --> np(F1),
         vp(F2),
         or(F1,F2,F).                 1 Strafpunkt · sichtbar
```

Neu ist eine solche deklarative Definition von Funktionen zwar nicht, nur wird sie eher auf dem Gebiet der Logischen Programmierung als beim Schreiben von Grammatiken benutzt. In einer »traditionellen« Definit-Klausel-Grammatik müßte der »Funktionsaufruf« entweder

[2] Es ist eigentlich nicht korrekt, bei einer Unifikationsgrammatik von »Vererbung« zu sprechen — Information wird beim Unifizieren nicht weitergegeben, sondern via *Structure Sharing* geteilt. Dennoch verdeutlicht dieser Sprachgebrauch vielleicht die Funktionsweise der Grammatik.

[3] Die Grammatiken des SVOX-Systems verwenden eine etwas andere Notation; ich habe sie hier zur besseren Verständlichkeit an den DCG-Formalismus von Prolog angeglichen.

durch partielle Evaluation direkt in die Regel integriert werden, welche den Operator benutzt (im Beispiel würde das zu drei S-Regeln führen), oder man ruft innerhalb der DCG Prolog-Klauseln auf, was wiederum eine Vermischung von Wissensrepräsentation und -verarbeitung wäre. Zudem werden die Grammatiken ja der Effizienz zuliebe nicht von Prolog abgearbeitet; die Beschränkung auf reine Grammatikregeln ohne prozedurale Elemente drängt sich daher schon aus diesem Grund auf.
Der Parser des SVOX-Systems entfernt die als »unsichtbar« markierten Regelanwendungen nachträglich aus der Ableitung — eine echte Erweiterung ist das nicht, aber recht nützlicher *syntactic sugar*.
Eine interessante Anwendung dieser »unsichtbaren Regeln« ist die sogenannte schwache Unifikation, von der die SVOX-Grammatik ausgiebig Gebrauch macht.

Schwache Unifikation

Auf die eben beschriebene Weise läßt sich zum Beispiel ein »Operator« definieren, dessen Anwendung nicht bestraft wird, wenn seine Argumente unifizierbar sind; andernfalls »kostet« die Regelanwendung 50 Strafpunkte.

```
WeakUnif50(A, A).                    0 Strafpunkte · unsichtbar
WeakUnif50(_, _).                   50 Strafpunkte · unsichtbar
```

Zugegebenermaßen ist es notwendig, innerhalb des Parsers gewisse Vorkehrungen zu treffen, damit ein solcher »Mißbrauch des Formalismus« nicht die Effizienz mindert.[4]

Koordination

Zum eigentlichen Thema dieses Artikels: Wie behandelt man bestimmte linguistische Phänomene, wenn man sich auf atomare Unifikation und somit auf kontextfreie Grammatiken beschränkt? Es sei an dieser Stelle nochmals ausdrücklich betont, daß die Problemstellung keine linguistisch saubere Behandlung von sprachlichen Phänomenen erforderte: So ist die mit Strafpunkten arbeitende Behandlung der Koordination linguistisch eher abwegig, sie liefert aber mit relativ niedrigem Rechenaufwand Resultate, die für die Sprachsynthese durchaus zu gebrauchen sind.
In diesem Abschnitt soll gezeigt werden, wie sich das doch eher komplexe Phänomen der Koordination einigermaßen zufriedenstellend durch eine kontextfreie DCG implementieren läßt. Als erstes seien einige sprachliche Beispiele aufgeführt:

A		Ich sah Anna.
A, B *Konjunktion* C		Ich sah Anna, Bruno und Cäcilie.
A, B	??	Ich sah Anna, Bruno.
A *Konjunktion* B, C	??	Ich sah Anna und Bruno, Cäcilie.
A *Konjunktion* B *Konjunktion* C		Ich sah Anna sowie Bruno und Cäcilie.
$Konjunktion_1$ A $Konjunktion_2$ B		Ich sah weder Anna noch Bruno.

[4] Für die Analyse der Wörter verwendet SVOX eine Modifikation des Earley-Algorithmus, für die Satzanalyse einen Bottom-Up-Chart-Parser. Lassen sich die beiden Atome im Beispiel unifizieren, werden *sowohl die gut als auch die schlecht bewertete* Kante in die Chart eingefügt. Der Parser müßte vor dem Einfügen einer leeren, inaktiven Kante überprüfen, ob eine ähnliche Kante nicht schon vorhanden ist, und gegebenenfalls lediglich die Zahl der »Strafpunkte« minimieren. Allerdings wurde dies bisher noch nicht implementiert.

Es ist zwar nicht der Fall, daß Konstruktionen wie »Ich sah Anna, Bruno« völlig ungrammatisch wären, sie sind bei Adjektiven sogar sehr häufig: »Das große, schöne Haus«. Sie sollen daher nicht völlig ausgeschlossen, sondern lediglich als eher schlecht bewertet werden, wobei die Zahl der Strafpunkte je nach Wortart variiert.
Betrachten wir nun die Grammatik für eine allgemeine Konstituente X, die koordinierbar sei. Gezeigt werden nur diejenigen Features, die für die Koordination von Bedeutung sind; es ist klar, daß zum Beispiel Agreement ebenfalls behandelt werden muß, oder daß Vorkehrungen zu treffen sind, damit eine Konstruktion wie »weder Anna als auch Bruno« ausgeschlossen wird.

```
x --> x-mult(f, f).                1 Strafpunkt nicht koordiniert
x --> x-mult(f, t).               30 Strafp.    ohne Konj. koordiniert
x --> x-mult(t, t).                1 Strafp.    mit Konj. koordiniert
```

Der Wert des ersten Attributs ist gleich `t`, wenn die Konstruktion eine Konjunktion enthält; der Wert des zweiten zeigt, ob es sich überhaupt um eine koordinierte Konstruktion handelt. Wie zu sehen ist, bestraft die Grammatik Konstruktionen, die ausschließlich durch Aufzählen koordiniert sind.

```
x-mult(f, f) --> x1.                1 Strafp.    [Anna]
x-mult(Kj, t) --> x1, x-mult(Kj, _).1 Strafp.    [Anna] [Bruno und C.]
x-mult(t, t) --> x1, x-konj.        1 Strafp.    [Bruno] [und Cäcilie]
x-mult(t, t) --> x-konj1, x-konj2.  1 Strafp.    [weder A.] [noch B.]

x-konj --> konj, x.                 1 Strafp.    [sowie] [A. und B.]
x-konj1 --> konj1, x.               1 Strafp.    [weder] [Anna]
x-konj2 --> konj2, x.               1 Strafp.    [noch] [Anna]
```

Offensichtlich muß die Grammatik rekursiv sein: Man will beliebig lange Aufzählungen zulassen, und es sollen auch beliebig tiefe Verschachtelungen möglich sein.
Ein Problem dieser Implementation sind Konstruktionen wie in »Morgen heiraten Anna und Bruno, Cäcilie und Daniel und Emil und Frieda«, bei denen die richtige Lesart zwar erkannt, aber sehr schlecht bewertet wird. Eine korrekte Analyse (und damit eine verständliche Aussprache) könnte in diesem Fall aber nur mit Hilfe einer semantischen Verarbeitung erreicht werden. In der Mehrzahl der Fälle zeigte sich jedoch, daß Koordinationen vernünftig analysiert werden.

Subkategorisierung, Fernabhängigkeiten und freie Wortstellung

Jedes Verb »verlangt« (subkategorisiert) nach bestimmten Kategorien; das Verb »sehen« erwartet zum Beispiel ein Subjekt und meistens ein Akkusativobjekt: »*Ich* sehe *ein Einhorn* im Garten.« Diese Konstituenten werden Komplemente genannt; freie Zusätze (»im Garten«) heißen Adjunkte. Ein bekanntes Problem beim Entwickeln von Grammatiken für das Deutsche ist die sehr freie Wortstellung der Komplemente und Adjunkte. »Im Garten sah er gestern ein Einhorn«, »Gestern sah er ein Einhorn im Garten«, »Ein Einhorn sah im Garten gestern er« — manche der Varianten mögen seltsam klingen, vollkommen ungrammatisch ist aber keine. Ein System, das von Menschen stammende Sätze des Deutschen verarbeiten soll, muß damit rechnen, Adjunkte und Komplemente in beliebiger Reihenfolge anzutreffen. Ein Formalismus, der Regeln für die hierarchische, vertikale Abhängigkeit (immediate dominance) mit Regeln für die Wortstellung innerhalb einer Konstituente (linear precedence) ver-

mischt, wird hier große Schwierigkeiten haben. Dazu zählen die im SVOX-System verwendeten kontextfreien Grammatiken ebenso wie zum Beispiel LFG. Dagegen ist in HPSG die Trennung der Regeln für *immediate dominance* und *linear precedence* formalisiert (vgl. [Pollard/Sag 1987], insbesondere Kapitel 7).

Es gibt nun mehrere Möglichkeiten, das Problem auch mit einer Grammatik anzugehen, die *immediate dominance* und *linear precedence* nicht trennt. In Frage kommen beispielsweise besondere Regeltypen, deren rechte Seiten beliebig permutierbar sind; für LFG wurde ein Permutationsoperator vorgeschlagen.

Im folgenden sei erläutert, wie die hier beschriebene Grammatik mit Subkategorisierung allgemein und dem Phänomen der freien Wortstellung im Besondern umgeht. Als Beispiele dienen Verben, da die SVOX-Grammatik zur Zeit weder subkategorisierende Nomina (die Tatsache, daß …) noch Adjektive (begierig, ihn zu sehen) kennt. Es wäre allerdings kein sehr großer Aufwand, die Grammatik entsprechend zu erweitern.

Verben subkategorisieren unter anderem nach folgenden Kategorien:[5]

Ich gehe.	Subjekt
Mich friert. Ich nehme *den Schal*.	Akkusativobjekt
Ich kaufte *der Marktfrau* einen Apfel ab.	Dativobjekt
Ich scheine *immer alles zu vergessen*.	»Zu«-Satz
Ich will *einen Salat essen*.	Infinitiv-Konstruktion
Ich verspreche, *daß ich gehe*.	»Daß«-Satz

Die Information, nach welchen Konstituenten ein Verb subkategorisiert, ist im Lexikon festgehalten. Üblicherweise wird dazu für das Deutsche wegen der freien Wortstellung eine Subkategorisierungs-Menge benutzt, bzw. (falls der Formalismus keine Mengen kennt) eine Subkategorisierungs-Liste, die als Simulation einer Menge dient. Nun wurde bereits zu Beginn dieses Artikels erläutert, daß man beim SVOX-System aus Effizienzgründen auf komplexe Features verzichten wollte. Stattdessen tragen Verben für jedes potentielle Element der Subkategorisierungs-Liste ein Attribut, dessen Wert anzeigt, ob das Verb nach der entsprechenden Konstituenten subkategorisiert.

Mit diesem Mechanismus werden (endliche) Mengen simuliert: Jedes Objekt kann höchstens einmal Element der Menge sein, und auf die einzelnen Elemente kann direkt zugegriffen werden, ohne eine Liste durchgehen zu müssen.

Fakultative Komplemente lassen sich auf diese Weise sehr effizient durch Unterspezifikation darstellen. Anstelle von zwei Einträgen für »geben« — einmal mit und einmal ohne Dativobjekt (»ich gab alles« vs. »ich gab dir alles«) — enthält das Lexikon nur einen Eintrag, bei dem der Wert des entsprechenden Attributs nicht spezifiziert ist. Konkret heißt dies, daß man in solchen Fällen die anonyme Variable angibt, die sowohl mit `t` (Komplement muß stehen) als auch mit `f` (Komplement darf nicht stehen) unifiziert werden kann.

Würde SVOX keine ausgefeilte Morphologie betreiben, sähen Lexikoneinträge ungefähr folgendermaßen aus — die Features stehen für Subjekt, Akkusativ-, Dativ-, Genetivobjekt, »zu«-Satz, Infinitivkonstruktion und »daß«-Satz:

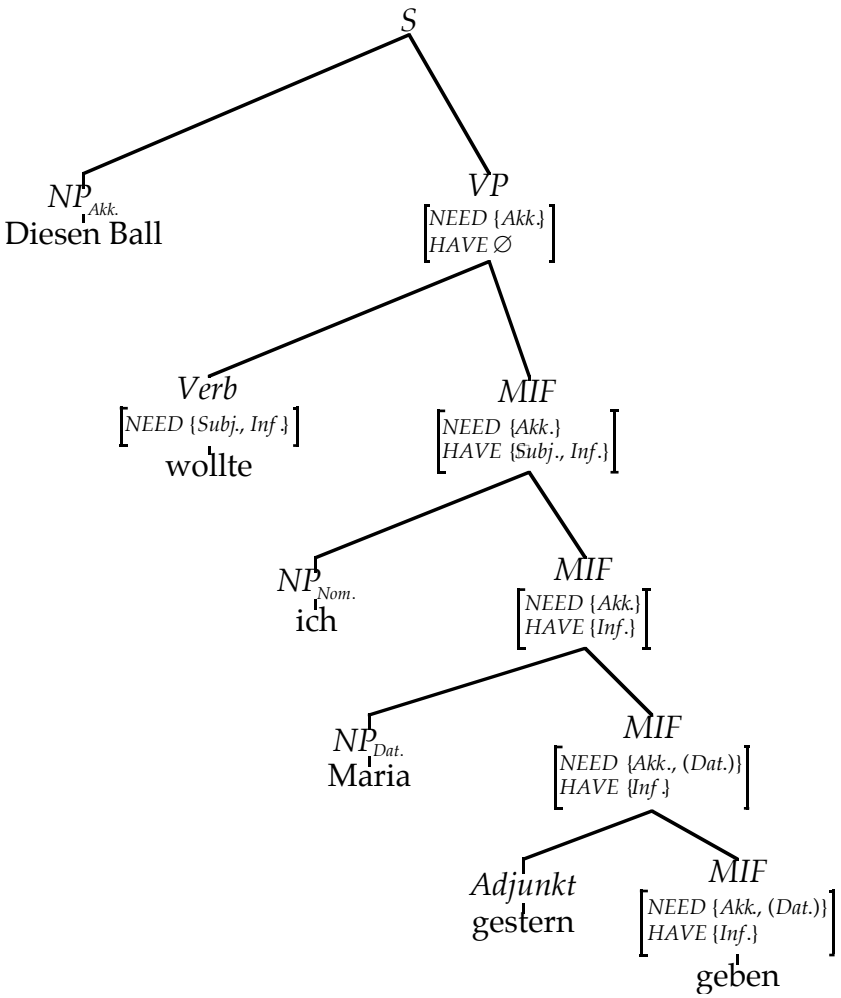
```
verb(t,f,_,f,f,f,f,…) --> [geben].     »Ich gab (dir) alles«
verb(t,f,f,f,f,t,f,…) --> [wollen].    »Ich will gehen«
verb(t,f,f,f,f,f,t,…) --> [wollen].    »Ich will, daß Du gehst«
verb(t,f,_,f,t,f,f,…) --> [scheinen]. »Anna scheint (mir) zu wachsen«
```

[5] Diese Liste ist nicht vollständig; die besprochene Grammatik kennt noch weitere Kategorien wie zum Beispiel »Prädikative« — »Bruno ist groß«.

Die Wortgrammatik sorgt dafür, daß Verben im Infinitiv nicht nach einem Subjekt, sondern nur nach den anderen möglichen Komplementen subkategorisieren können. Dadurch ist es nicht notwendig, Konstruktionen mit Kontrollverben (Raising-/Equi-Verben) gesondert zu behandeln.
Nun ist bekannt, welche Kategorien vorhanden sein müssen, damit ein Satz vollständig ist; es bleibt die Frage, mit welchem Mechanismus die einzelnen Komplemente abgebunden werden. Betrachten wir dazu zunächst, an welchen Stellen im Satz Komplemente auftreten:

Gestern	hat	Maria vor Zeugen	behauptet,	ein Einhorn gesehen zu haben.
Vorfeld		**Mittelfeld**		**Nachfeld**
Genau eine Konstituente		*Null bis mehrere Konstituenten in beliebiger Reihenfolge*		

Ein Aussagesatz besteht aus dem Vorfeld (mit genau einer Konstituente), dem finiten Verb, dem Mittelfeld und eventuell aus einem Partizip; die Behandlung des Nachfelds geschieht mittels besonderer Mechanismen, die hier nicht erläutert werden.
Welche Komplemente vorhanden sein müssen, hängt von der Subkategorisierung des Verbs ab; die SVOX-Grammatik trennt daher erst das Vorfeld ab und unterteilt dann den Rest in die drei Teile finites Verb, Mittelfeld und Partizip. Die Hauptaufgabe der Satzgrammatik ist es, die Komplemente im Mittelfeld »aufzusammeln« und mit dem Vorfeld und dem Subkategorisierungsrahmen des Verbs so in Beziehung zu setzen, daß nur solche Sätze akzeptiert werden, denen weder ein Komplement fehlt noch eines überzählig haben.
Angelehnt an die Behandlung der Fernabhängigkeiten (Unbounded Dependencies) in HPSG verwendet die hier beschriebene Grammatik *zwei* Mengen, um diese Aufgabe zu lösen. Zum einen trägt jede Konstituente des Mittelfeldes die Information, welche noch nicht abgebundenen Komplemente sie enthält (im folgenden »HAVE« genannt), und zum anderen gibt eine »NEED« genannte Menge an, welche Komplemente benötigt werden, bisher aber noch nicht abgebunden werden konnten. Anders als HPSG arbeitet die SVOX-Grammatik jedoch ohne Traces.
Das Vorgehen der Grammatik soll nachfolgend anhand des Satzes »Diesen Ball wollte ich Maria gestern geben« erläutert werden; auf der nächsten Seite ist der Syntaxbaum in vereinfachter Form abgedruckt, welchen der Parser als Ergebnis liefert. Insbesondere sind die NEED- und die HAVE-Menge als Mengen dargestellt, obwohl sie wie die Subkategorisierungsinformation der Verben als Bündel von separaten Features implementiert sind.
Das Mittelfeld wird rekursiv aufgebaut: Das Verb »geben« subkategorisiert als Infinitiv nach einem Akkusativobjekt und optional nach einem Dativobjekt; die Wortgrammatik sorgt dafür, daß Infinitive niemals nach einem Subjekt subkategorisieren, damit Konstruktionen mit Kontrollverben behandelbar sind. Somit enthält das aus »geben« alleine bestehende Mittelfeld eine nicht abgebundene Infinitiv-Konstruktion, und es müssen noch ein Akkusativobjekt und ein Dativobjekt gefunden werden.
Ein Adjunkt wie »gestern« ändert nichts an den beiden »Mengen«; die Werte der Attribute werden unverändert übernommen. Zusammen mit der Dativ-NP »Maria« ergibt sich ein Mittelfeld, dessen HAVE-Menge eine Infinitivkonstruktion enthält und dessen NEED-Menge anzeigt, daß noch ein Akkusativobjekt zu finden ist.



Grafik: Subkategorisierung wird in der beschriebenen Grammatik ähnlich wie die Fernabhängigkeiten in HPSG behandelt, allerdings ohne leere Kategorien zu benötigen.

Die Nominativ-NP »ich« stellt ein Subjekt dar; die NEED-Menge von »Maria gestern geben« zeigt aber nicht an, daß ein Subjekt fehlen würde. Somit wird das Subjekt zur HAVE-Menge hinzugefügt.

Auf diese Weise wird das Mittelfeld schrittweise rekursiv aufgebaut. Die benötigten Grammatikregeln werden am Beispiel des Dativobjekts erläutert; die meisten anderen möglichen Komplemente werden ähnlich behandelt. Lediglich Subjekte sind wegen der Übereinstimmung mit dem finiten Verb in Person und Numerus etwas komplizierter.

- Ein Dativ ergibt zusammen mit einem Mittelfeld, das bisher keinen Dativ benötigte, ein neues Mittelfeld, das nun einen Dativ überzählig hat. Die übrigen Features bleiben unverändert.
- Ein Dativ ergibt zusammen mit einem Mittelfeld, das einen Dativ benötigte, ein neues Mittelfeld, das nun keinen Dativ mehr braucht. Die übrigen Features bleiben unverändert.

Da Mengen durch Attribute mit booleschem Wertebereich simuliert werden, lassen sich diese beiden Regeln mit Hilfe einer Negation zu einer einzigen zusammenfassen: Das neue

Mittelfeld hat genau dann einen Dativ überzählig, wenn das alte keinen benötigte. Somit enthält die Grammatik für jedes der möglichen Komplemente eine einzige Regel. Es wird hierbei angenommen, daß kein Dativ auf ein Mittelfeld trifft, das bereits einen nicht abgebundenen, »überzähligen« Dativ enthält. Dies ist eigentlich nicht korrekt, hat aber bisher zu keinen falschen Analysen geführt. Der Grund dafür ist, daß selten zwei gleiche Konstituenten nicht-lokal abgebunden werden.
Ist das Mittelfeld vollständig auf die beschriebene Art aufgebaut worden, wird es mit dem Verb zur »klassischen« Verbalphrase verknüpft. Dies bewerkstelligt ein besonderer Abbinde-Operator:

Verb	Mittelfeld		Verbalphrase	
NEED	NEED	HAVE	NEED	HAVE
t	f	t	f	f
t	f	f	t	f
f	t	f	t	f
f	f	t	f	t
f	f	f	f	f

Benötigt ein Verb beispielsweise einen Dativ und»bietet« das Mittelfeld einen Dativ an, hat die Kombination von beiden weder einen Dativ »zuviel« noch einen »zuwenig«. Die übrigen Zeilen können analog gelesen werden. Manche Kombinationen fehlen in der Tabelle; zum Beispiel ist es ausgeschlossen, daß ein Mittelfeld gleichzeitig einen Dativ überzählig hat und einen braucht, denn in diesem Fall wäre der Dativ schon beim Aufbau des Mittelfeldes abgebunden worden.
Als letztes muß die »Verbalphrase« mit dem Vorfeld verbunden werden: Ein Satz besteht aus einem Dativobjekt und einer Verbalphrase, die noch einen Dativ benötigt, aber nichts überzählig hat. Ähnliches gilt für die anderen möglichen Komplemente, zusätzlich kann im Vorfeld auch ein Adjunkt (»Gestern brannte es«) oder ein Expletivum (»Es brennt ein Feuer im Garten«) stehen; in diesen beiden Fällen müssen sowohl die NEED- als auch die HAVE-Menge leer sein.

Schlußbemerkungen

Das Anwendungsgebiet der Sprachsynthse läßt niedrige Rechenzeiten besonders wünschenswert erscheinen: Die Grammatiken des SVOX-Systems beschränken sich daher auf atomare Unifikation, was die syntaktische Analyse stark beschleunigt. Allerdings hat dies unmittelbar zur Folge, nur noch die Klasse der kontextfreien Sprachen verarbeiten zu können. Entgegen einer in der Computerlinguistik verbreiteten Meinung war es aber durchaus möglich, mit der beschriebenen Grammatik auch komplexere linguistische Phänomene zu behandeln.
Das vorgestellte Verfahren kann für Anwendungsgebiete, die lediglich eine syntaktische Analyse erfordern, einen Mittelweg zwischen einer linguistisch sehr sauberen, aber zeitintensiven Unifikationsgrammatik einerseits und einer nur oberflächlichen, daher qualitätsmindernden Analyse andererseits darstellen.

Dank

Abschließend möchte ich der Gruppe für Sprachverarbeitung der ETH Zürich danken; einmal für die interessanten Aufgabenstellungen, an denen ich als Student arbeiten durfte, ganz besonders aber Christof Traber und Schamai Safra, die so manche Diskussionen mit mir führten, von denen ich viel profitieren konnte.

Literatur

[Haugeneder/Trost 1993] Haugeneder, Hans/Trost, Harald: Beschreibungsformalismen für sprachliches Wissen. In: Görz, Günther (Hrsg.): Einführung in die künstliche Intelligenz. Bonn [u. a.]: Addison Wesley, 1993. ISBN 3-89319-507-6. Seite 372 – 424.

[König/Seiffert 1989] König, Esther/Seiffert, Roland: Grundkurs Prolog für Linguisten. Tübingen: Francke, 1989. ISBN 3-7720-1749-5.

[Pereira/Warren 1980] Pereira, F. C. N./Warren, D. H. D.: Definite Clause Grammars for Language Analysis — A Survey of the Formalism and a Comparison with Augmented Transition Networks. Artificial Intelligence, Vol. 13, 1980. Seite 231 – 278.

[Pollard/Sag 1987] Pollard, Carl J./Sag, Ivan A.: Information-based syntax and semantics. Vol. 1: Fundamentals.CSLI Lecture Note No. 13. Stanford: CSLI, 1987. ISBN 0-93707-24-5 (Paperback) und 0-937073-23-7 (gebunden).

[Russi 1990] Russi, Thomas: A Framework for Syntactic and Morphological Analysis and its Application in a Text-to-Speech System. Dissertation Eidgenössische Technische Hochschule Zürich Nr. 9328.

[Traber 1993] Traber, Christof: Syntactic Processing and Prosody Control in the SVOX TTS System for German. In: Proc. Eurospeech '93, vol. 3, S. 2099 – 2102, 1993.