Fachgruppe Sprachwissenschaft

Universität Konstanz

Arbeitspapier 42

Kategoriale Unifikationsgrammatik

Klaus von Heusinger

Inhalt

Kategoriale Unifikationsgrammatik[1]
Klaus von Heusinger

1. Einleitung

Kategoriale Unifikationsgrammatiken gehören einer neuen Generation von Grammatikmodellen an, die stark durch ihre Anwendung in der Computerlinguistik geprägt sind. Sie haben jedoch darüber hinaus auch linguistisch sehr interessante Eigenschaften. So stützen sie sich weitgehend auf Information, die im Lexikon definiert ist. Dies ermöglicht nicht nur, syntaktische und semantische Information am gleichen Ort zu kodieren, sondern erleichtert auch eine parallele Verarbeitung. Syntax und Semantik können mit allgemeinen Prinzipien arbeiten und brauchen keine hochspezialisierten Regeln. Auch andere Grammatiktheorien schlagen inzwischen den Weg zu "mehr Lexikon" ein.

In einer weiteren Hinsicht sind Kategoriale Unifikationsgrammatiken ausgesprochen interessant: Sie bilden die Vereinigung von Methoden, die in den Kategorialgrammatiken einerseits und den Unifikationsgrammatiken andererseits angewendet werden. Kategorialgrammatiken wurden in den 30er Jahren entwickelt und sind damit der älteste formale Grammatiktyp, zumindest älter als kontextfreie Grammatiken, Transformationsgrammatiken oder Dependenzgrammatiken. Sie zeichnen sich vor allem durch ihre begriffliche Klarheit und Einfachheit sowie ihre Möglichkeiten aus, Syntax und Semantik zu verbinden. Ihr einfacher Formalismus wird auf recht verschiedenen Gebieten genutzt. Kategorialgrammatiken bilden z.B. die Semantik der meisten logischen Sprachen und könnten darüber hinaus auch zur Klärung philosophischer und ontologischer Fragen dienen. Dann bilden sie das Bindeglied zwischen Semantik und Logik (van Benthem). In der Linguistik sind Kategorialgrammatiken zur Beschreibung natürlicher Sprachen (Bar-Hillel) und besonders ihrer Semantik (Montague, Cresswell) genutzt worden. Doch auch in der Syntax und Morphologie haben sie eine erneute Anwendung gefunden (Steedman, Moortgat). Schließlich eignen sie sich auch besonders für den Einsatz in der Computerlinguistik und Datenverarbeitung (Uszkoreit).

Der zweite Partner der Fusion, die Unifikationsgrammatik, wurde erst in den 80er Jahren entwickelt (Shieber), ist also eine Art "Enkel" der Kategorialgrammatik und ein direkter Abkömmling der Phrasenstrukturgrammatiken. Diese entwickelten eine starke Tendenz, immer mehr Information in den Kategorien zu kodieren, was schließlich zu den flexiblen Merkmalsstrukturen der Unifikationsgrammatiken führte, die von der Graphtheorie weiter beeinflußt wurden. Ihr Haupteinsatzgebiet ist die Computerlinguistik. Beide Grammatiktypen ergänzen sich in hervorragender Weise. Die Kategorialgrammatik trägt zur klaren Darstellung der Funktor-Argument-Struktur der Sprache bei, während die Unifikationsgrammatik die flexiblen Merkmalsstrukturen liefert, in denen speziellere Information kodiert werden kann.

2. Kategorialgrammatiken

Kategorialgrammatiken beruhen im wesentlichen auf drei Ideen: Einmal die Idee der Bedeutungskategorien von Wörtern bei Husserl, dann die Fregesche Idee von der Funktionalität von Sprache und schließlich die Idee der Schichtbarkeit oder der Hierarchie von Sprache bei Russell.

[1] Der vorliegende Aufsatz entspricht einer überarbeiteten Version des 4. Kapitels meiner Magisterarbeit "Kategoriale Unifikationsgrammatiken".

Ein Entwurf einer Kategorialgrammatik, in dem diese drei Ideen das erste Mal vereint wurden, ist 1922 von Le,niewski formuliert und 1935 von seinem Schüler Ajdukiewicz zur sogenannten klassischen Kategorialgrammatik ausgearbeitet worden. Bar-Hillel wendete 1953 diesen logisch-formalen Ansatz auf die in der strukturalistischen Linguistik entwickelte Konzeption der Konstituente und Konstituentenstruktur an und prägte als erster den Begriff Kategorialgrammatik (categorial grammar). Später zeigte er die schwache Äquivalenz von Konstituentenstrukturgrammatiken (phrase structure grammars) und Kategorialgrammatiken und bezweifelte im Anschluß an Chomsky die Adäquatheit dieser Grammatiken für natürliche Sprachen.
Lambek erweiterte 1958 den vorhandenen Formalismus aus mathematischer Sicht zum Lambek-Kalkül, in dem er alle Regeln streng formal herleitete. Geach (1971) erweiterte die klassische Form unabhängig von Lambek um ähnliche Regeln, gab aber eine stärker am linguistischen Gebrauch orientierte Begründung für seine Regeln.
In Kategorialgrammatiken werden sprachliche Ausdrücke kategorisiert. Namen und Sätze sind Grundkategorien (n bzw. s), während alle anderen Ausdrücke Funktorkategorien sind, die entsprechend ihrer Anwendung auf andere Kategorien gebildet werden. So erhält das Verb *spaziert* die Kategorie n\s, da es auf einen Ausdruck der Kategorie n (z.B. *Hans*) angewendet, einen Ausdruck der Kategorie s (*Hans spaziert*) ergibt. Es können auch komplexere Funktorkategorien aufgebaut werden:

(1)

Kategorie	Kategorienindex	sprachlicher Ausdruck
Satz	s	die Rose blüht
Name	n	Rose, Sokrates
einstellige Prädikate	s/n	blüht, fliegt
zweistellige Prädikate	s/n n	liebt
Attribute	n/n	schön, groß
Artikel	n/n	die, eine
Satzadverbien	s/s	nicht
Satzkonjunktionen	s/s s	und, oder, aber
Adverbien	(s/n)/(s/n)	stark, gut
Adverbien	((s/n)/(s/n))/((s/n)/(s/n))	sehr

Die Funktorkategorie in (2) sind alle rechtsgerichtet. Daher muß der Satz unächst in die sogenannte polnischen Notation umgeformt werden. In (3) wird ein Formalismus benutzt, dessen Funktorkategorien in beide Richtungen arbeiteten.

(2)

und	sehr	stark	duftet	der	Flieder	blüht	die	Rose
s/ss	((s/n)/(s/n))/((s/n)/(s/n))	(s/n)/(s/n)	s/n	n/n	n	s/n	n/n	n
							s/n	n
						n/n	n	s
	((s/n)/(s/n))/((s/n)/(s/n))		(s/n)/(s/n)			n		
		(s/n)/(s/n)		s/n				
			s/n			n		
s/ss				s				s
				s				

(3)

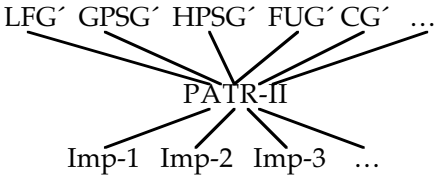
Heiner	thinks	that	Helmut	sleeps
n	n\s/s	s/s	n	n\s
			s	
		s		
	s			

3. Unifikationsgrammatik

Unifikationsgrammatiken, die ich am Beispiel des PATR-II Formalismus einführen werde, sind eine Weiterentwicklung der Konstituentenstruktur- oder Phrasenstrukturgrammatiken (PS–Grammatiken). Sie greifen eine Idee auf, die sich in den PS-Grammatiken immer weiter entwickelt hat. Die Ansicht nämlich, daß eine einfache Kategorie nicht atomar sein muß, sondern durchaus in sich strukturiert sein kann. Zunächst wurden Ausdrücke in die vier Hauptkategorien NP, VP, Adj und PP eingeteilt, entsprechend, ob sie einen verbalen und/oder nominalen Charakter haben. Dann wurden die feineren Unterteilungen in Nebenkategorien nach den Merkmalen Numerus, Genus, Kasus, Finitheit usw. eingeführt (Kratzer et alii 1973 und Gazdar et alii 1985). Dies erlaubte den PS-Grammatiken Kongruenz und Rektion zu beschreiben, von denen Chomsky und andere behauptet hatten, sie ließen sich nur in einer Transformationsgrammatik beschreiben. In diesem Sinne können Unifikationsgrammatiken als die am weitesten fortgeschrittenen PS-Grammatiken bezeichnet werden. Ihre zwei charakteristischen Merkmale sind die flexiblen Merkmalsstrukturen anstelle der unveränderlichen Kategorien, die vielfältige Erweiterungen ermöglichen, und die Unifikation, die verschiedene Merkmalsstrukturen oder strukturierte Kategorien unifiziert (= vereinigt). Dies erlaubt es den einzelnen Kategorien nur teilweise definiert zu sein ("partial information"). Die restliche Information kann durch Unifizierung erworben werden.

Weitere Merkmale von Unifikationsgrammatiken sind Monostratalität und die zentrale Rolle des Lexikons. Beide Eigenschaften besitzt auch die Kategorialgrammatik. Interessant ist, daß Kategorialgrammatiken von Anfang an lexikongetrieben waren, während andere Grammatiktypen erst seit einigen Jahren eine Entwicklung zu "mehr Lexikon" durchmachen. In diesem Sinne ist der älteste formale Grammatiktyp auch der modernste (vgl. Uszkoreit 1986, S. 23). Ferner sind Unifikationsgrammatiken so ausdrucksstark wie möglich, ganz im Gegensatz zu Transformationsgrammatiken, die oft einen stark eingeschränkten Formalismus haben, um spezielle linguistische Probleme lösen zu können. Unifikationsgrammatiken sind jedoch nicht als Grammatiktheorien, sondern als ein flexibles grammatisches Werkzeug gedacht, mit denen Sätze analysiert ("geparst") werden können (Calder et alii 1987, S. 5 und Uszkoreit 1986, S. 24). Sie erheben keinerlei Ansprüche auf eine linguistische oder psychologische Realität. Damit gehören sie zusammen mit FUG, DCG u.a. zu den "methodischen Grammatiken", die im Gegensatz zu den "substantiellen Grammatiken" wie GPSG, LFG, GB etc. stehen. Diese Grammatiktypen erheben einen Anspruch auf linguistische oder psychologische Realität. In diesem Zusammenhang kann man Unifikation auch in einem viel weiteren Sinne verstehen. Eine Unifikationsgrammatik bietet einen neutralen Formalismus an, in den die linguistisch motivierten Lösungsvorschläge sprachlicher Phänomene von substantiellen Grammatiken übersetzt werden können. Stärken und Schwächen der verschiedenen Ansätze lassen sich so vergleichen und möglicherweise zu einer aussagekräftigeren Grammatik "unifizieren" (ebd.). Unifikationsgrammatiken dienen auch Grammatikimplementationen in Computerprogrammen. Unter diesem Aspekt lassen sie sich auch als "Assembler-Sprache" verstehen. Calder et alii (1987, S. 5) stellen sich die Mitlerrolle für PATR-II folgendermaßen vor:

(4)

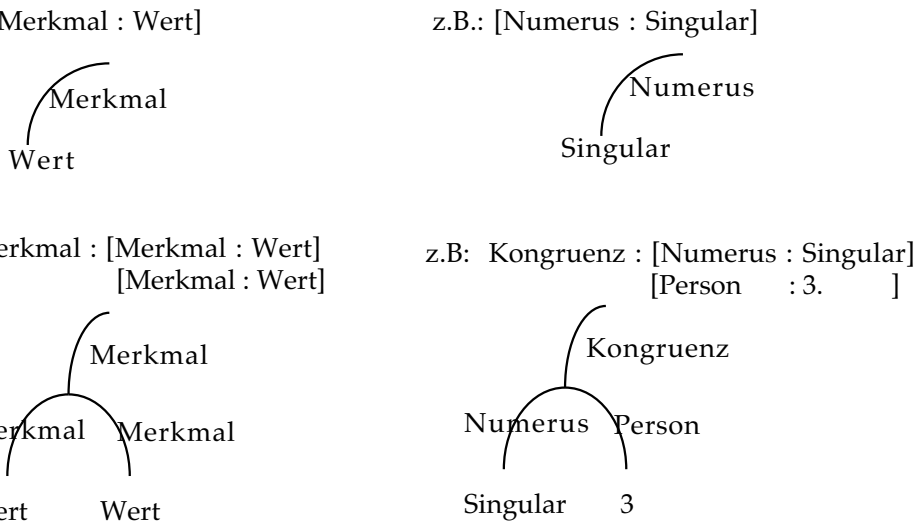


Das Apostroph "´" bezeichnet den je deklarativen Teil des Grammatiktyps, *Imp-* bezeichnet eine Implementation. Im folgenden Abschnitt wird eine Unifikationsgrammatik ausführlicher diskutiert.

3.1. Der PATR-II Formalismus

In diesem Abschnitt soll der PATR-II Formalismus stellvertretend für andere Unifikationsgrammatiken vorgestellt werden. Die Darstellung orientiert sich an Shieber (1986) mit den Modifikationen von Egli (1987). Charakterisierend für den PATR-II Formalismus ist, daß Ausdrücken nicht eine Kategorie (wie bei den Kategorialgrammatiken) sondern eine komplexe Merkmalsstruktur zugewiesen wird. Merkmalsstrukturen bestehen aus geordneten Paaren von ‹Merkmal, Merkmalswert›, wobei das Merkmal einfach, der Merkmalswert einfach oder komplex sein kann. Diese Strukturen lassen sich verschieden darstellen. So bevorzugt der PATR-II Formalismus eine Graph-Darstellung, die in einen DAG (directed acyclic graph) übersetzbar ist. Diesen werde ich meist neben dem Graphen in eckigen Klammern ("[ ]") anführen. Er läßt sich wiederum in eine einfachere Termschreibweise mit annotierten PS-Kategorien oder Regeln überführen. In der Graph-Darstellung entspricht jedem Merkmal eine "Kante", die zur Unterscheidung zu Strukturbäumen als Bogen dargestellt wird, einem einfachen Wert entspricht ein Endpunkt einer solchen Kante und einem komplexen Wert entspricht eine Struktur mit Kanten.

(5)



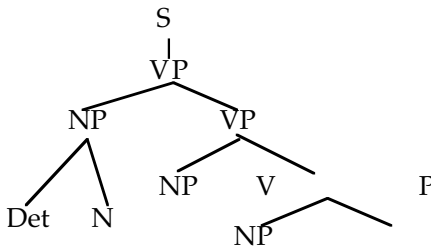
Der PATR-II Formalismus stützt sich auf die beiden Hauptoperationen der Verkettung (concatenation) und der Unifikation (unification). Die Verkettung von Konstituenten wird in einer traditionellen PS-Regel angegeben, der je einem PS-Baum entspricht. Die PS-Regeln geben an, welche Ausdrücke Konstituenten sind.

(6)

(1) S -> VP

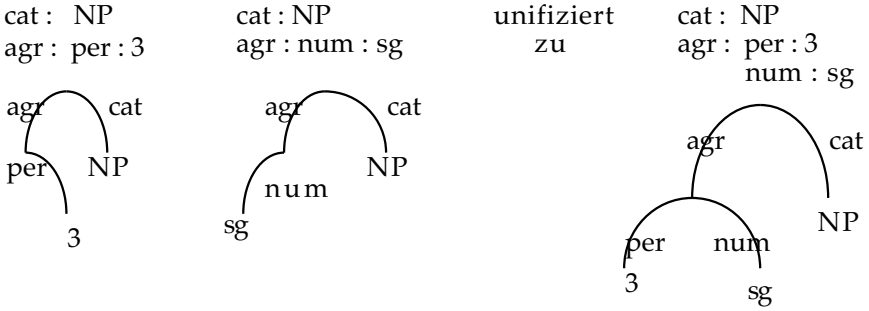
(2) VP -> NP VP

(3) NP -> Det N



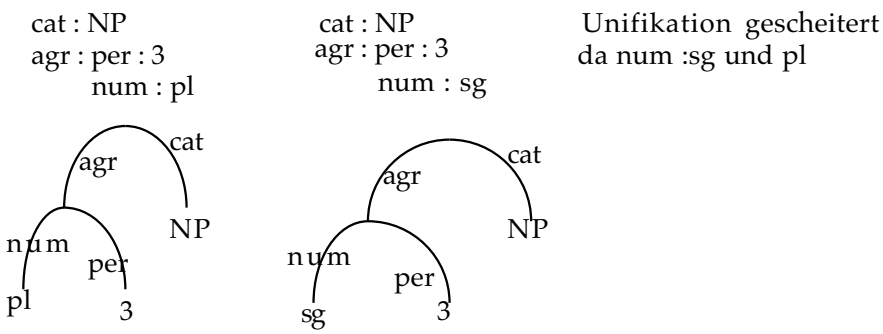
Die zweite Hauptoperation, die Unifikation, ist folgendermaßen definiert: Zwei Konstituenten unifizieren, wenn die Merkmalswerte für ein Merkmal entweder den gleichen Wert haben oder mindestens ein Merkmalswert eine Variable oder nicht spezifiziert ist. (Calder et alii 1987, S. 9ff). Für die Graph-Darstellung läßt sich die Unifikation auch als ein "Übereinanderlegen" der Graphen verstehen. Decken sich dabei die Bögen und Werte, ist die Unifikation gelungen, stehen hingegen verschiedene Werte übereinander, ist sie gescheitert. Dies sieht in der Graph- und DAG-Schreibweise so aus (ich verzichte hier auf die eckigen Klammern für die DAGs):

(7)



Im folgenden Beispiel scheitert die Unifikation, da das Merkmal *num* (Numerus) einmal den Wert *pl* und einmal den Wert *sg* hat.

(8)



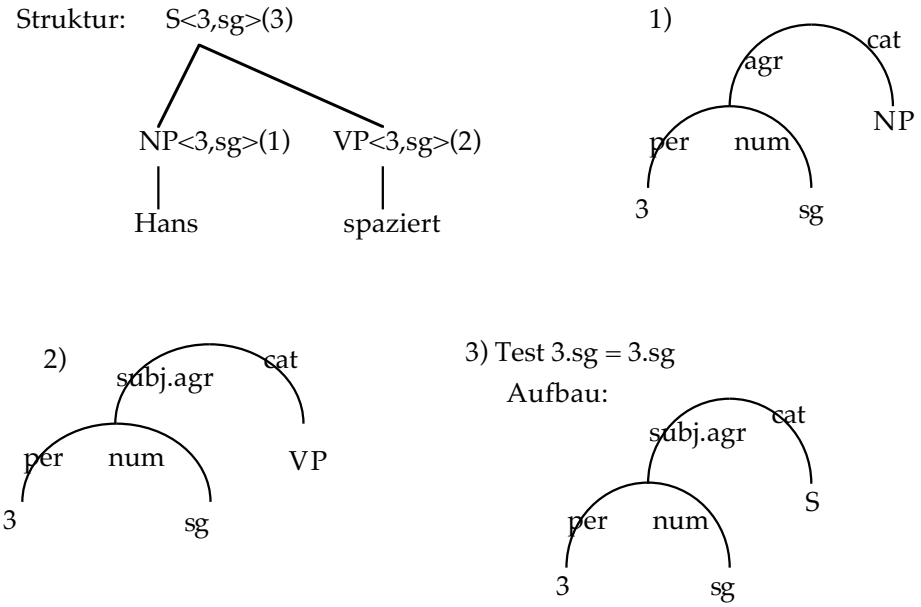
Der Unifikation lassen sich zwei Funktionen zuordenen. Einmal soll sie die Zusammenfassung von Konstituenten und deren Merkmalen steuern. Dies ist auch unter dem Namen "pattern matching" ("Mustervergleich") oder "equality-testing" ("Gleichheitstest") bekannt (Shieber 1986, S. 12). Dann muß sie auch dafür sorgen, daß notwendige Informationen an die Mutterkonstituente weitergereicht werden (Vererbung oder "feature passing"). Beide Funktionen lassen sich als Gleichungen darstellen.

Traditionell werden die Gleichungen, die prüfen, ob zwei Strukturen unifizierbar sind, und die Gleichungen, die die Information weitergeben, nicht getrennt (vgl. Shieber 1986, Uszkoreit 1986 und Calder et alii 1987). Wir können jetzt die Struktur einer PATR-II Regel angeben.

(9) Eine PATR-II Regel besteht aus:
1. Einer Strukturregel in Form einer PS-Regel
2. Testgleichungen, die überprüfen, ob zwei Merkmalsstrukturen unifizierbar sind.
3. Aufbaugleichungen, die die notwendigen Merkmale und Werte an die Mutterkonstituente weitergeben, sofern eine Unifikation nach (2) möglich ist.

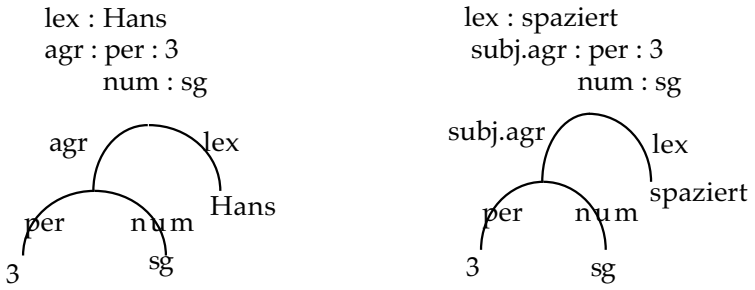
Machen wir dies an dem Beispiel der Kongruenz von Subjekt und Prädikat im Verbalsatz deutlich. Dazu gebe ich die PATR-II Regel an, um dann an der Graph-Darstellung die Unifikation nachzuvollziehen. Jeder Knoten im Strukturbaum wird mit einem Graphen beschrieben, was darauf hinweist, daß die Merkmalsstrukturen (Graphen) nur lokal sind bzw. je anstelle einer Kategorie stehen. Das Merkmal *cat* (Kategorie) ist ausgezeichnet, da es nur von der Strukturregel, nicht aber von der Testgleichung betroffen ist. Daher können zwei verschiedene Kategorien unifizieren:

(10)
Struktur	S -> NP VP
Test:	NPagr = VPsubj.agr
Aufbau:	Ssubj.agr = VPsubj.agr



Mit diesen PATR-II Regeln lassen sich auch die Kongruenz in Nominalphrasen, Subkategorisierung und Verbvalenz sowie Lückenregel angeben. Abschließend möchte ich noch eine weitere Regel vorstellen, um die Flexibilität des Formalismus zu zeigen. Wie die Kategorialgrammatiken sind auch die Unifikationsgrammatiken lexikon-getrieben, d.h. die Information für den Aufbau von komplexeren Ausdrücken kommt aus dem Lexikon. Lexikoneinträge werden im PATR-II Format folgendermaßen aussehen. Das Merkmal *lex* soll das Merkmal für Lexem sein. Calder et alii (1987) schlagen an dieser Stelle ein Merkmal für die phonetische Form vor (s.u. 4.2). Die Strukturen könnten natürlich mit weiteren Merkmalen wie Verbvalenz, Zeitangaben u.v.a. erweitert werden.

(11)



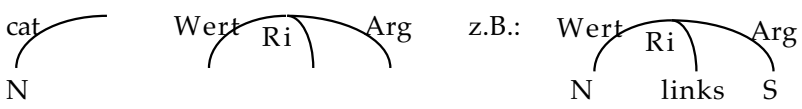
4. Die Entwicklung zur Kategorialen Unifikationsgrammatik

Die Idee, Unifikationsgrammatik und Kategorialgrammatik zu "unifizieren" entstand an zwei Stellen gleichzeitig. 1986 erwähnt Karttunen (1986) die *Categorial Unification Grammar* (CUG), die er zur Beschreibung des Finnischen benutzt. Uszkoreit (1986) formuliert sie etwas expliziter. Er entwickelt sie als eine Computerimplementation für HPSG in Stanford/USA. Gleichzeitig entwickelten Calder et alii 1987 in Edinburgh/ Schottland eine sehr verwandte Version, die sie *Unification Categorial Grammar* (UCG) nennen und die genau wie die CUG Uszkoreits auf einem PATR-Formalismus beruht und in ihm formuliert ist. Beide Versionen sind nur in Ansätzen dokumentiert und stark durch ihre Implementationen auf verschiedenen Computern geprägt.
Uszkoreit (1986) diskutiert nur die Graph-Darstellung von Funktorkategorien, der Funktionalapplikation und der Funktionalkomposition und beläßt es ansonsten mit Hinweisen, was seine CUG zu leisten vermag. Calder et alii (1987) benutzen eine reine DAG-Darstellung, in der sie Phonologie, syntaktische Kategorie, Semantik und Richtung (der Applikation) angeben.

4.1. Die Funktion-Argument-Struktur in Graphen - CUG

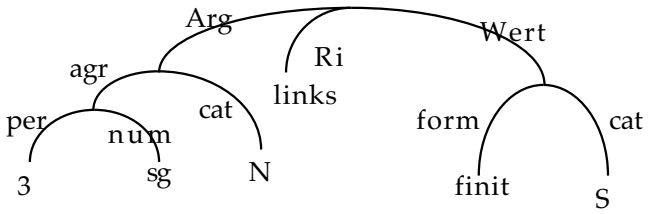
Im letzten Abschnitt hatten wir die Darstellung von einfachen Kategorien in Graphen vorgestellt. Uszkoreit (1986, S. 5) stellt auch komplexe Kategorien im Sinne Ajdukiewicz (= Funktorkategorien) als Graphen dar. So ist eine Kategorie nicht mehr der atomare Wert des Merkmales *cat*, sondern selbst wieder eine komplexe Struktur aus dem Wert, der Richtung der Applikation und des Argumentes, wie in (12). Für die komplexe Funktorkategorie N\S (=VP) mit Nebenkategorien sieht der DAG und der Graph wie in (13) aus. Dies ließe sich als Term einfach als N [3 sg] \ S [finit] schreiben.

(12)



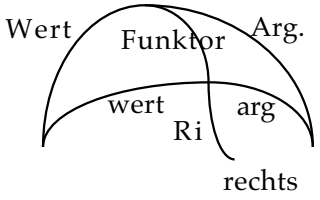
(13)

Arg: cat: N
agr: per: 3
num: sg
Ri: links
Wert: cat: S
form: finit



(14)

Wert -> Funktor Arg.
‹Wert› = ‹Funktor.wert›
‹Arg.› = ‹Funktor.arg›
‹Funktor-Ri› = rechts



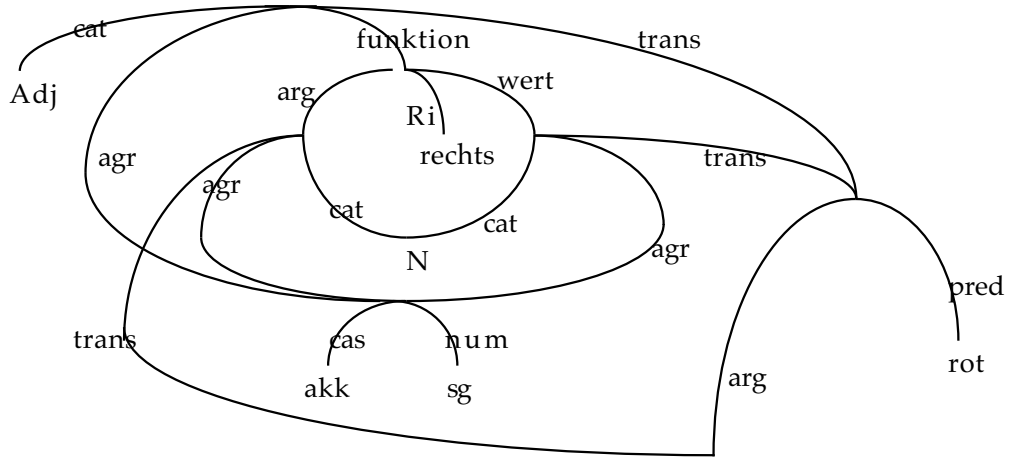
(14) stellt die Funktionalapplikation in der CUG nach Uszkoreit (1986, S. 7) dar. Diese komplexe Formel, besagt nicht mehr als die einfache Fassung der Funktionalapplikation (X/Y Y -> X). Sie leistet jedoch dann mehr, wenn die Kategorien noch weiter spezifiziert sind und es wichtig ist, welche Informationen vererbt werden. Das Merkmal Funktor-Ri(chtung) gibt nur die Applikationsrichtung an. Ich ersetze es durch die gewohnte Schreibweise. In unsere Struktur der PATR-II Regel (vgl. (9)) übersetzt, sieht dies so aus:

(15)

	Struktur:	Wert -> Funktor/Arg.
	Test:	Arg. = Funktor.arg
	Aufbau:	Wert = Funktor.wert

In Worten: Prüfe, ob das Argument dem Argument oder Nenner (bei Ajdukiewicz) der Funktorkategorie entspricht. Ist dies der Fall, unifiziere die beiden Kategorien. Ihr Ergebnis ist der Wert (Zähler) der Funktorkategorie. Uszkoreit erweitert seine Graphen noch um das Merkmal *trans*(lation), das die Semantik angeben soll, und um das Merkmal *Funktion*, in das die Funktorkategorie eingetragen wird. Das Merkmal *cat* benutzt er für die Bezeichnung der Wortart. Die Graph-Darstellung ermöglicht es auf einfache Weise, gleichen Merkmalswerte verschiedenen Teilgraphen zuzuordnen (z.B. die Werte für *agr*).

(16)



Dieser zwar recht ästhetische aber auch komplexe Graph (Uszkoreit 1986, S. 13) läßt sich etwas übersichtlicher als DAG, wie in (17) darstellen. Die <1> soll einen gemeinsamen Wert bezeichnen, gibt also genau das an, was im Graphen durch Zusammenlaufen von Bögen dargestellt wird.

(17)

```
⎡ cat:        Adj                                                    ⎤
⎢ agr:        ⎡ cas:   akk⎤ <1>                                      ⎢
⎢             ⎣ num:   sg ⎦                                          ⎢
⎢ funktion:   ⎡ arg:   ⎡ agr:    <1>]                    ⎤ ⎤        ⎢
⎢             ⎢        ⎢ cat:    N]                      ⎥ ⎢        ⎢
⎢             ⎢        ⎣ trans:  [arg:  [ pred:  rot]]]  ⎦ ⎢        ⎢
⎢             ⎢ Ri:    rechts]                             ⎢        ⎢
⎢             ⎢wert:   ⎡ agr:    <1>                ⎤     ⎢        ⎢
⎢             ⎢        ⎢ cat:    N                  ⎥     ⎢        ⎢
⎢             ⎣        ⎣ trans:   [ pred:  rot]]     ⎦     ⎦        ⎢
⎣ trans:      [ pred:  rot]                                          ⎦
```

Noch einfacher ist jedoch die Termschreibung:

(18) Adj [akk, sg] : N [akk, sg] / N [akk, sg] : rot

Es handelt sich also um ein Adjektiv im Akkusativ Singular, das attributiv auf ein Nomen im Akkusativ Singular angewendet werden kann. Für den Gebrauch im Deutschen müßte noch das Genus angegeben sein. Erstaunlicherweise gibt Uszkoreit neben dem Merkmal *cat*, unter dem er die Wortart (hier: Adjektiv) einträgt noch die Funktion als Attribut (hier: N/N) an. Uszkoreit gibt keine Begründung für diese Verdopplung an. Weiterhin ist nicht klar, was das Merkmal *trans* bedeutet. Ein weiterer Nachteil der Graph- und auch DAG-Darstellung wird hier schon deutlich. Sie werden sehr schnell sehr komplex und unübersichtlich. Mit diesen kritischen Bemerkungen verlassen wir Uszkoreits CUG und wenden uns der Edinburgher UCG zu.

4.2. Komplexe Merkmalsstrukturen - UCG

Der Edinburgher Ansatz, Ideen aus der Kategorialgrammatik und der Unifikationsgrammatik in einem Formalismus gemeinsam zu verwenden, sieht etwas anders als der von Uszkoreit aus. Die UCG (Unification Categorial Grammar) wurde als grammatische Basis für einen Parser in PATR-II formuliert und in C-PROLOG implementiert. Es wird auf eine Graph-Darstellung verzichtet und eine Art Term-Schreibweise benutzt, die in eine DAG-Schreibung übersetzbar, selbst jedoch sehr viel kürzer ist, wie ich das im letzten Abschnitt bereits vorgeführt habe (Zeevat et alii 1987, S. 220). Die erste Version der UCG wurde in Calder et alii (1987) *Problems of Dialog Parsing* entwickelt (zur weiteren Geschichte siehe dort, Vorwort, S. i). Ich werde die revidierte Fassung von Zeevat et alii (1987) vorstellen, da ihr Formalismus klarer und leichter verständlich ist und auch dem bisher Behandelten näher steht.
Die Merkmalsstrukturen oder flexiblen Kategorien der UCG, die auch Zeichen ("sign") genannt werden, bestehen aus vier Hauptmerkmalen und ihren Werten. Die Hauptmerkmale sind *Phon* für phonologische Information, *Kat* für syntaktische, *Sem* für semantische und *Ri* für die Richtung der Anwendung der jeweiligen

Funktorkategorie. Eine so definierte Merkmalsstruktur kann mit der Variable Z für Zeichen bezeichnet werden. Ich habe die Merkmalsnamen dem deutschen Sprach- und Abkürzungsverständnis angepaßt. Die Merkmale können untereinander oder mit Doppelpunkten getrennt nebeneinander aufgeführt werden (Zeevat et alii 1987, S. 197):

(19)
Phon
Kat oder: Phon : Kat : Sem : Ri
Sem
Ri

Aus Übersichtsgründen werden meist beide Varianten miteinander kombiniert, wie das auch in dem typischen Eintrag für das Verb visit der Fall ist (ebd., S. 198):

(20)
Phon: visit
Kat: s[finit]/ P_1 : np : x : vor/ P_2 : np : y : nach
Sem: [e] VISIT (e, x, y)
Ri: O

Der Wert für das Merkmal Phonologie ist die orthografische Darstellung oder eine Variable P_1, P_2, … Dies entspricht in etwa dem Merkmal *lex* in (11). Zeevat et alii (1987) geben keine weiteren Informationen in diesem Merkmal an, sondern nur vage Andeutungen, was möglich sein könnte (ebd., S. 217). Ich werde dieses Merkmal benutzen, um phonologische Information darin zu kodieren.
Unter dem Merkmal *Kat*(egorie) steht entweder eine einfache Kategorie (s, n oder np) oder eine Kombination aus komplexen Merkmalsstrukturen (Zeichen), mit der Angabe ihrer Hauptkategorien. Diese werden wie in (19) nebeneinander geschrieben, um nicht zu große Strukturen wie z.B. (17) zu erhalten. Der Wert des Merkmales *Kat* in (20) sähe untereinandergeschrieben so aus:

(21)

Kat:	Phon:	⌈ ⌉	⌈ P_1 ⌉	⌈ P_2 ⌉
	Kat:	|s[fin] | /	|np | /	|np |
	Sem:	| | /	|x | /	|y |
	Ri:	⌊ ⌋	⌊ vor⌋	⌊ nach ⌋

Diese Darstellung entspricht natürlich der kategorialgrammatischen (n\s)/n, nur daß hier die Kategorien noch in sich strukturiert sind. Da es in diesem Formalismus nur einen Querstrich ("/") gibt, der daher auch keine Richtung anzeigen kann, wird ein Merkmal *Ri*(chtung) ("order") eingeführt, das angibt, wo das Argument zu stehen hat, das mit der Funktorkategorie unifiziert werden soll (vgl. auch 4.1. z.B. (12)) Die Variable O in (20) gibt an, daß dies Merkmal nicht spezifiziert ist.
Die Kategoriensymbole (s, n, np) können durch weitere Nebenkategorien wie Finitheit, Kasus usw. näher spezifiziert werden. Hier liegt eine entscheidende Möglichkeit, Sprache adäquater zu beschreiben. So sollten in dem Beispiel (20) die NPs mit dem Merkmal Kasus spezifiziert sein:

(22) Kat: s[finit]/ P_1 : np[nom] : x : vor/ P_2 : np[akk] : y : nach

Eine Einschränkung der Möglichkeiten des Formalismus sei hier gleich angemerkt: Der Wert des Merkmals *Kat* entspricht einer kategorialgrammatischen Regel, die normalerweise nur auf benachbarte Konstituenten angewendet werden kann. Weiterreichende grammatische Beziehungen können entweder nicht oder nur nach

Einführung weiterer Regeln beschrieben werden. Diese zusätzlichen Regeln würden den Formalismus komplizierter machen, und es müßte sichergestellt sein, daß sie nicht übergenerieren. Dies könnte jedoch durch die annotierten Kategorien gewährleistet sein. Die Semantik wird in der "Indexed Language" (auch InL) angegeben. Sie ist aus der Diskursrepresentationstheorie von Kamp und der Verbbehandlung von Davidson entstanden (Zeevat et alii 1987, S. 202). Die genaue Form dieser Semantik kann hier nicht behandelt werden. Die Variable *e* soll auf ein Ereignis hinweisen.
Das Interessante ist hier das Zusammengehen von Syntax und Semantik. Es werden einerseits zwei getrennte Merkmale (*Kat* und *Sem*) angegeben, doch sind sie miteinander so verbunden, daß sie die gleichen Variablen benutzen. Die Variable *x* unter dem Merkmal *Kat* in *np[nom] : x : vor*, die die Bedeutung der einzusetzenden NP angibt, ist die gleiche, die in der Semantik unter *Sem* in *[e] VISIT (e, x, y)* wieder auftaucht. Diese Behandlung erlaubt ein paralleles Verarbeiten von Syntax und Semantik, aber auch gewisse Abweichungen von einer strikten Parallelität, sofern sich diese als notwendig und sinnvoll erweisen (Uszkoreit 1986, S. 12).
Grundsätzlich kann an jeder Stelle, für jeden Wert eine Variable stehen. Es wird vereinbart, daß ein Merkmal nicht aufgeführt zu werden braucht, wenn es nicht spezifiziert ist.
Die Verkettung (Konkatenation) von zwei Konstituenten geschieht nach der Funktionalapplikation. Dies wird in komplexer Schreibung für beide Richtungen formuliert:

(23) FA: X <- X/Y Y
R1 P_1 P_2 : Kat : Sem -> P_1 : C/ Z : Sem Z(P_2 : vor)
R2 P_1 P_2 : Kat : Sem -> Z(P_2 : nach) P_1: C / Z : Sem

Diese Schreibweise ist sehr ungewohnt. Man muß zuerst die beiden Ausdrücke rechts vom Pfeil von links nach rechts lesen und erhält dann den Wert der Funktionalapplikation links vom Pfeil. In Worten liest sich R1: Wenn ein Funktor mit der Phonologie P_1, der Kategorie C/X und der Semantik S einem Argument Z mit der Phonologie P_2 und der Richtung *vor* vorangeht und Z läßt sich mit X unifizieren, dann ist das Ergebnis ein Ausdruck mit der Phonologie P_1 P_2, der Kategorie C und der Semantik S, wobei S und C durch Unifikation abgeändert sein können. Das Gleiche gilt für R2 mit der Modifikation durch *nach* statt *vor* (Zeevat et alii 1987, S. 202). Diese Anweisung verliert ein wenig dadurch, daß der Test, ob die Unifikation möglich ist, und der Aufbau der entstehenden Konstituente nicht klar getrennt sind. Außerdem ist sie nicht ganz so klar wie die Funktionalapplikation in der reinen Kategorialgrammatik, da den durch Unifikation veränderten Kategorien Rechnung getragen werden muß.

Zum Abschluß dieser kurzen Einführung in die UCG soll ein Beispiel vorgerechnet werden. Der Ausdruck *visit* soll zunächst mit dem Ausdruck *John* verbunden werden. Ich gebe die Merkmalsstrukturen für beide Ausdrücke an und unifiziere dann.

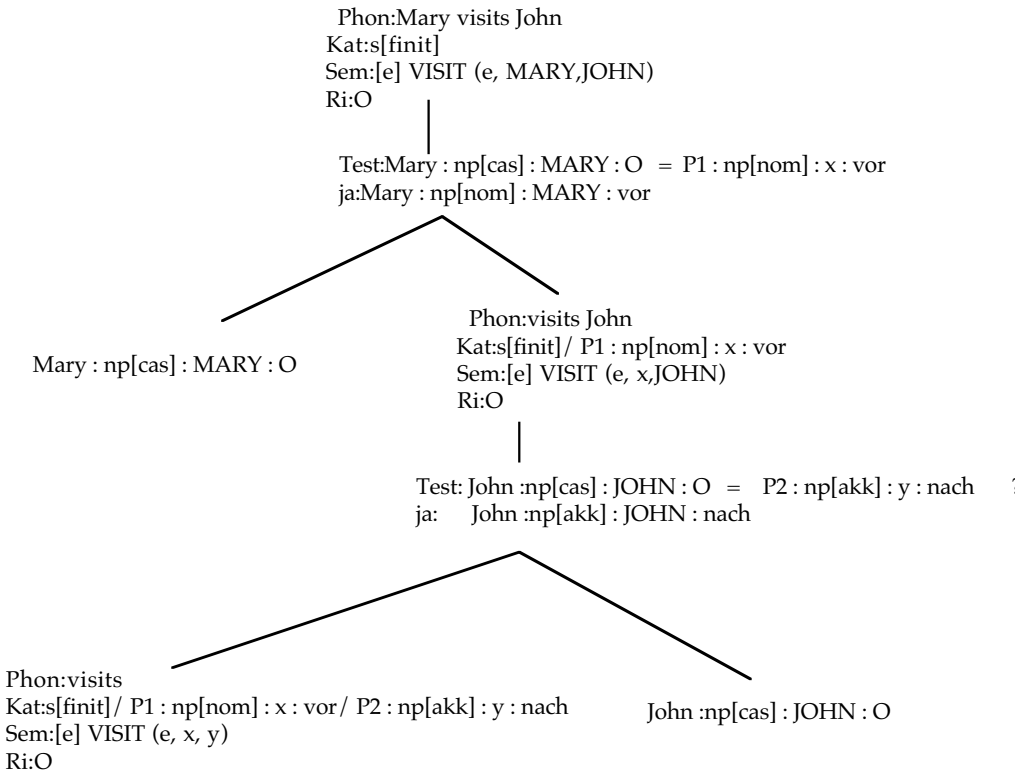
(24a) Phon: visits
Kat: s[finit]/ P_1 : np[nom] : x : vor/ P_2 : np[akk] : y : nach
Sem: [e] VISIT (e, x, y)
Ri: O
Test: John :np[cas] : JOHN : O = P_2 : np[akk] : y : nach ?
ja: John :np[akk] : JOHN : nach

Aufbau: Phon: visits John
Kat: s[finit]/ P_1 : np[nom] : x : vor
Sem: [e] VISIT (e, x,JOHN)
Ri: O

Test: Mary : np[cas] : MARY : O = P_1 : np[nom] : x : vor
ja: Mary : np[nom] : MARY : vor

Aufbau: Phon: Mary visits John
Kat: s[finit]
Sem: [e] VISIT (e, MARY,JOHN)
Ri: O

(24b)



Die Verkettung und Unifikation finden in zwei Schritten statt. Zunächst prüfe man, ob das Argument der Funktorkategorie (hier np[akk] : y : nach) mit den Merkmalen des zweiten Ausdruckes (John :np[cas] : JOHN : O) unifizierbar ist. Ist dies der Fall kann man die neue Merkmalsstruktur nach der Regel R2 (23) aufbauen. Diese verbinde ich dann entsprechend mit dem Ausdruck *Mary*. (24a) ist von oben nach unten, (24b) von unten nach oben zu lesen.