Computerlinguistik
Universität Zürich

Gaudenz Lügstenmann
Stüssistr. 33
8006 Zürich
01 / 350 03 59
gaudi@swissonline.ch

Einführung in LFG

anhand der "Grammar Writer's Workbench"

betreut von S.Clematide

Seminar "Syntaxtheorien und computerlinguistische Praxis"
SS 2000
Prof. Dr. M. Hess

Inhalt:

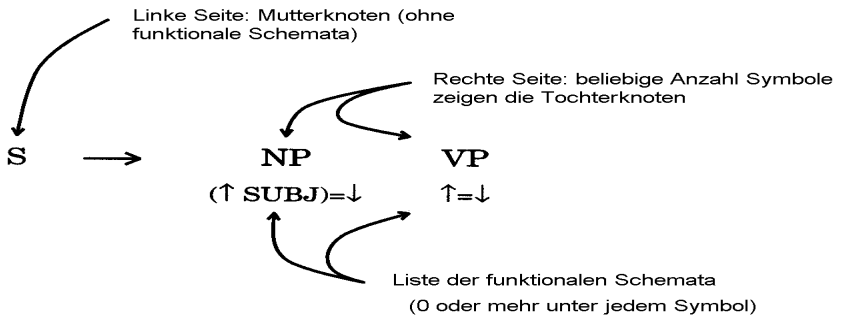
Vorbemerkungen

Der Text soll gleichzeitig in die Lexikal-Functional-Grammar, kurz **LFG** und in eine Anwendung dieser Theorie (**Grammar Writers Workbench GWB**) einführen und als Vorbereitung für das Referat vom 6.4. 2000 dienen. Die vorliegende Seminararbeit zeigt die praktische Anwendung der LFG-Theorien anhand einfacher Beispiele u.a. auch der Musterssätze des Seminars, in der GWB-Umgebung. Dieses Programm wurde entwickelt, um (LFG) Grammatiken zu schreiben und zu testen. Hauptsächlich basiert es auf der Theorie von Kaplan & Bresnan 1982.
Joan Bresnan und Ron Kaplan begannen in den späten 70er Jahren mit der Entwicklung ihrer Theorie. Sie folgten dabei Theorien der relationalen Grammatik aus der Strömung der generativen Grammatiken. Mit LFG versuchten sie eine Theorie zu entwickeln, deren Syntax nicht nur strukturbasiert aufgebaut ist. Sie vertraten die Meinung, dass Syntax mehr enthält, als mit einfachen Phrasenstrukturregeln ausgedrückt werden kann. Ihr Ziel war also eine Grammatiktheorie, die sich präzis berechnen lässt und gleichzeitig psychologisch ein realistisches Model der menschlichen Sprache darstellt.
Die Arbeit ist auch ohne LFG-Vorkenntnisse zu lesen. Zu allen **fettgedruckten** Wörtern in dieser Arbeit ist im Glossar eine kurze Erklärung zu finden.
Mein Ziel ist, dass alle SeminarteilnehmerInnen nach dem Lesen dieses Textes den LFG-Eintrag im Bussmann verstehen. Alle Rückmeldungen sind sehr willkommen.
Noch ein Hinweis zum Lesen. Die Darstellungen und Zeichnungen sind in den Text integrierte Bestandteile des Textes und für ein vertieftes Verständnis sehr wichtig.

1 Die Schreibweise in LFG

1.1 Die Regeln

Die Schreibweise der **grammatischen Regeln** in LFG sind ähnlich wie kontextfreie Phrasenstrukturregeln. Es gibt einen Regelkopf links des Pfeiles und einen Regelrumpf rechts dieses Pfeiles. Das Symbol im Kopf entspricht dem Mutterknoten und die Symbole im Rumpf entsprechen dessen Tochterknoten. Die Besonderheit von LFG kommt erst jetzt.
Jedes Symbol im Rumpf einer Regel erhält noch ein **funktionales Schema**. Diese funktionalen Schemata stehen jeweils unter dem entsprechenden Symbol. Ich werde weiter unten versuchen sehr genau zu erklären, welche Rolle diese funktionalen Schemata spielen.



Figur 1: Format der LFG-Regeln

Was ich bis jetzt beschrieben habe, können wir natürlich leicht weiterführen. Auch die Tochterknoten können Regelköpfe sein, die jeweils wieder auf einen oder mehrere untergeordnete Knoten verweisen. Und auch diese untergeordneten Knoten haben ein zugehörendes funktionales Schema.
Die kleine Beispielgrammatik (Abbildung 1) aus dem GWB zeigt drei Grammatikregeln. Aus rein technischen Gründen können im GWB die formalen Schemata jeweils nicht unter den Symbolen stehen. Wenn nach einem Tochterknoten-Symbol ein Doppelpunkt steht, folgen die funktionalen Schemata. Die funktionalen Schemata lassen sich auch an den ↑↓-Pfeilen erkennen. Wenn die Tochterknoten zwischen ( ) Klammern stehen, dann sind sie keine obligatorischen, sondern fakultative Knoten in dieser Regel. Die Knoten, die kein funktionales Schema haben geben einfach die funktionalen Schemata von unten nach oben weiter. Wenn kein funktionales Schema steht ist also ↑ = ↓ gemeint.

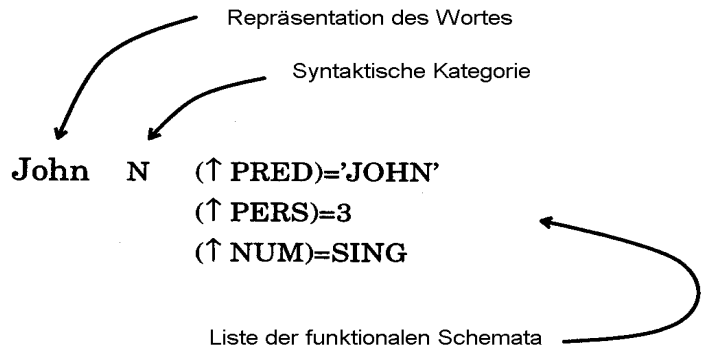
```
TOY  ENGLISH
   NP → (DET)
        (ADJ)
        N.

   S → (NP:  (↑ SUBJ)=↓
             (↓ CASE)=NOM)
       VP:  (↑ TENSE)
            ↑=↓.

   VP → V
        (NP).

----
```

Abbildung 1: drei Regeln aus der Grammatik Toy English (im GWB)

Hier muss ich noch ergänzen, dass diese, jeweils zugeordneten funktionalen Schemata recht komplex sein können. Man sagt auch es werde jedem Knoten eine Liste funktionaler Schemata zugeordnet. In der Satzregel in Abbildung 1 ist dem NP-Knoten eine Liste funktionaler Schemata zugeordnet bestehend aus zwei funktionalen Schemata.

1.2 Das Lexikon

Als nächstes möchte ich hier zeigen, wie ein einfacher **Lexikoneintrag** aussehen kann.



Figur 2: Format der LFG Lexikoneinträge

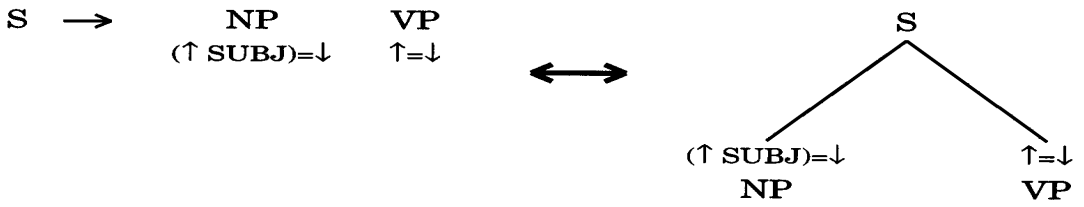
Ein Lexikoneintrag eines Wortes enthält drei verschiedene Dinge: Erstens das Item selbst. Es steht genau in der Form im Lexikon, wie es das System später erkennen soll. In LFG muss deshalb nicht zwingend ein Vollformenlexikon benützt werden. Allfällige Affixe können an zweiter Stelle im Lexikon aufgeführt werden.
An dieser Stelle steht im einfachen Fall nur die syntaktische Kategorie des Wortes. Diese Kategorien entsprechen jeweils einem Knoten aus den Regeln. Im GWB sind hier noch morphosyntaktische Makros angehängt. Dies benützt man im GWB zum Beispiel für Plural- oder Personalformen (siehe Lexikoneintrag von bite).
Drittens wird die dazugehörende Liste funktionaler Schemata aufgeführt.

Lexikoneinträge:

bite	V	S-ED	(↑ PRED)='BITE<(↑ SUBJ) (↑ OBJ)>'.
bark	V	S-ED	(↑ PRED)='BARK<(↑ SUBJ)>'.
see	V	S-ING	(↑ PRED)='SEE<(↑ SUBJ) (↑ OBJ)>'.
dog	N	S	(↑ PRED)='DOG'
			(↑ NUM)= SG
			(↑ PERS)= 3.
student	N	S	(↑ PRED)='STUDENT'
			(↑ NUM)= SG
			(↑ PERS)= 3.
Mary	N	S	(↑ PRED)='MARY'
			(↑ NUM)= SG
			(↑ PERS)= 3.

1.3 Die Konstituentenstruktur

Die **c-Struktur** ist, wie Phrasenstrukturregeln, eine Struktur die Hierarchien schafft. Nur dass die c-Strukturen in LFG die funktionalen Schemata auch in die hierarchische Baumstruktur „mitnehmen“. Aus den LFG Grammatikregeln lassen sich Ableitungsbäume zeichnen.



Figur 3: Beziehung zwischen den Regeln und den Ableitungsbäumen

Zu jedem Knoten der Baumstruktur gehört einfach auch das dazugehörige funktionale Schema. Zentral sind die funktionalen Schemata der Lexikoneinträge an den Enden der Äste. So entsteht eine vollständige c-Struktur in LFG.

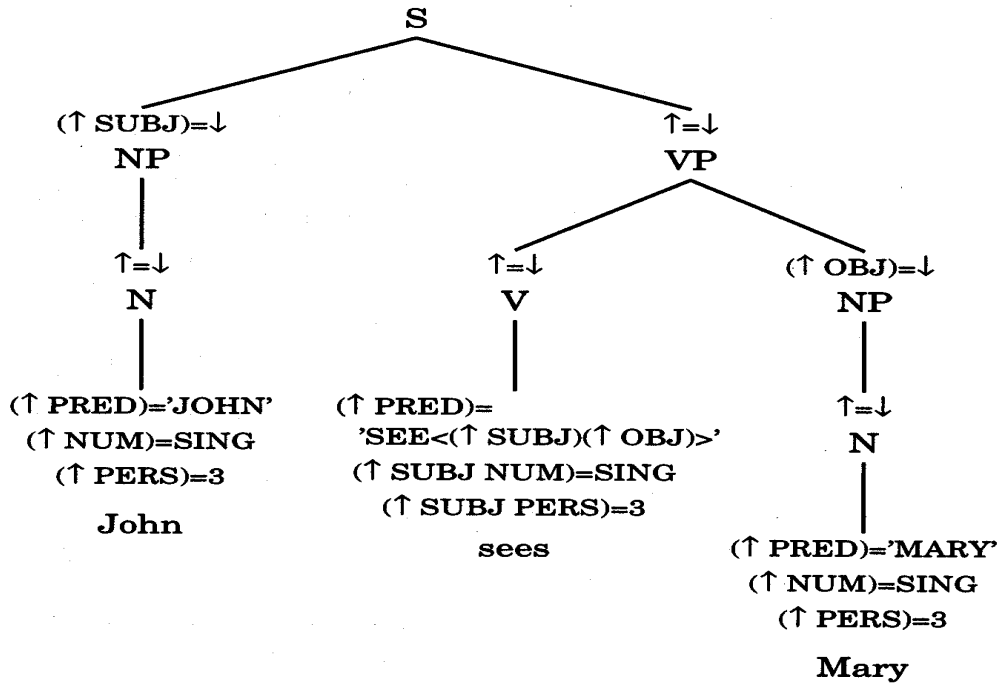


Abbildung 2: c-Struktur

Solche vollständigen Bäume könnten jetzt von einem maschinellen System weiterverarbeitet werden. Im GWB laufen diese Prozesse im Hintergrund ab. Am Bildschirm kann man nur die Ergebnisse der Berechnungen anschauen.
Übrigens: Kann das System keine vollständige c-Struktur aufbauen, ist entweder die Grammatik ungenügend, oder die Eingabe war agrammatisch.
Wenn für einen zu analysierenden String im GWB keine vollständige c-Struktur aufgebaut werden kann, gibt es zu diesem String auch keine f-Struktur aus. Falls das System aber alle Wörter kennt gibt es eine **Chart** aus. Damit lässt sich das Regelwerk einer Grammatik verbessern, denn die Einsicht in partiell geparste Einheiten hilft beim Verbessern des Regelwerks.

2 Instanzierung

Die c-Struktur wird **instanziert**. Das bedeutet, dass mit Hilfe der funktionalen Schemata ausgehend von der c-Struktur eine **f-Struktur** berechnet wird. Wie das funktioniert, zeige ich in den nächsten Abschnitten Schritt für Schritt auf.

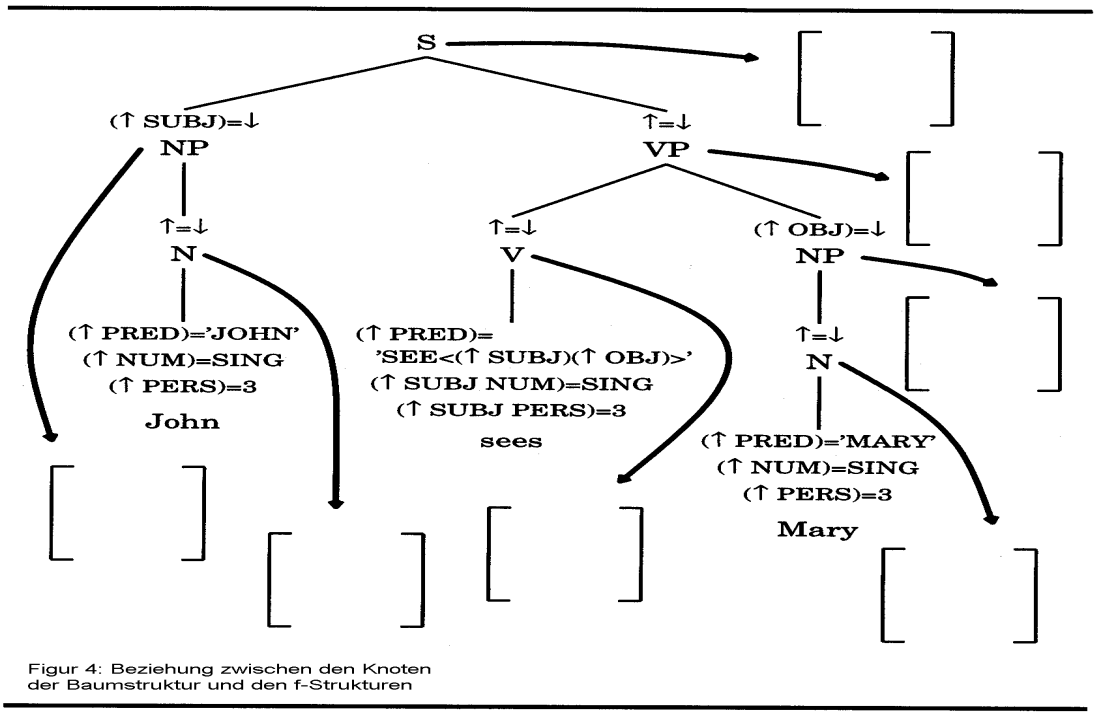
2.1 Koindizierung

Wie gesagt, ist das Ziel der Instanzierung eine f-Struktur des Satzes zu erhalten. F-Strukturen sind Merkmalspaare, die in eckigen Klammern aufgelistet werden.

$$f_n \begin{bmatrix} A & f_m \begin{bmatrix} B & C \\ D & E \end{bmatrix} \\ F & G \\ H & I \end{bmatrix}$$

Abbildung 4: Format einer funktionalen Struktur (f-Struktur)

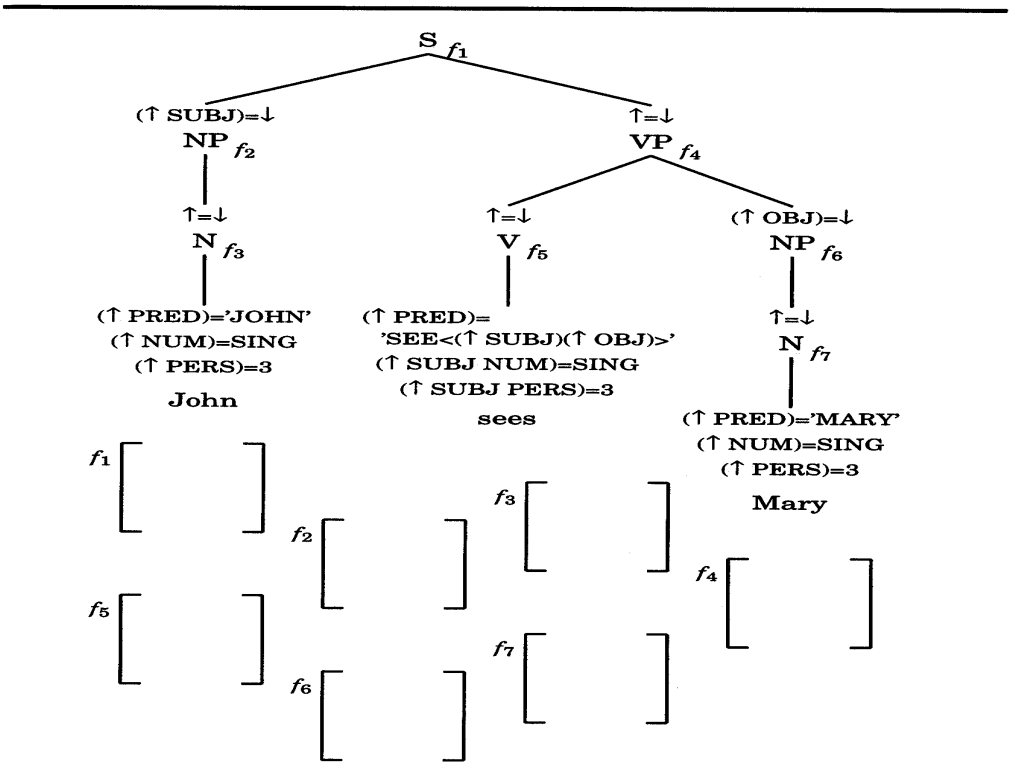
In der Merkmalstruktur N hat das Attribut A den Wert f_m (eine weitere Merkmalstruktur), das Attribut F den Wert G und das Attribut H den Wert I.
Die Berechnung der f-Struktur aus einer c-Struktur ist ein ganz wichtiger Mechanismus der LFG. Die Verknüpfung zwischen diesen Strukturen beginnt bei den Knoten der c-Struktur. Jedem Knoten der c-Struktur ist eine f-Struktur zugeordnet und jede dieser f-Strukturen erhält einen eigenen Namen (f_1 – f_n). Vorerst sind diese Merkmalsstrukturen noch leer.



Figur 4: Beziehung zwischen den Knoten der Baumstruktur und den f-Strukturen

Auch wenn zwei Knoten den gleichen Namen haben, bekommen sie eine eigene f-Struktur zugeordnet. Wir sagen, die Knoten eines Satzes werden **koindiziert**. Die zugehörende f-Struktur erhalten jeweils den gleichen Namen.

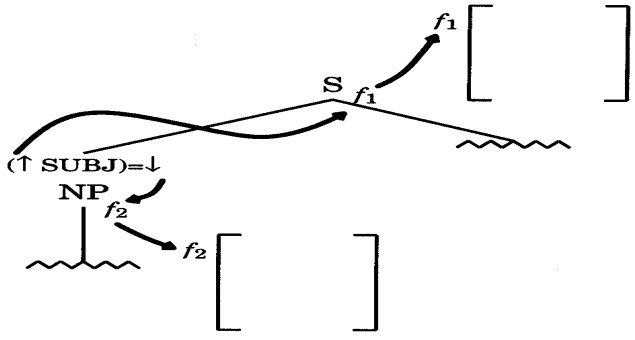
Jetzt sind wir für die Instanzierung bereit. Wie fliessen jetzt die Informationen?



Figur 5: Namengebung für die f-Strukturen und koindizieren der Knoten

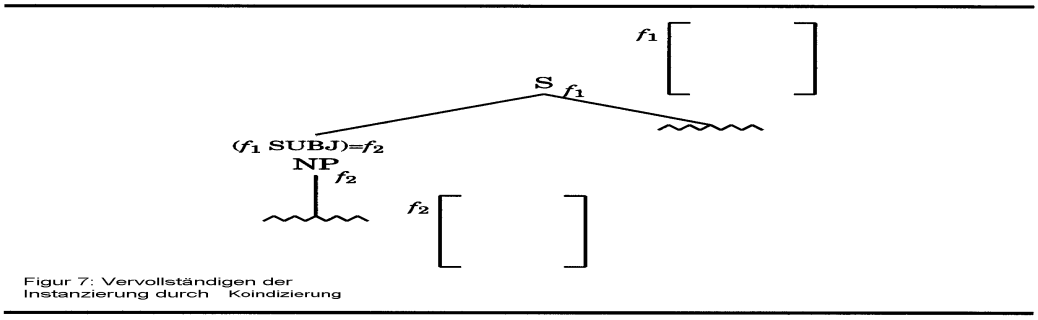
2.2 Durch Metavariablen-Bindung zur funktionalen Beschreibung

Die Pfeile werden in der Literatur Metavariablen genannt. Der ↑- Pfeil ist die Mutter-Metavariable, weil er auf das funktionale Schema des Muterknotens verweist. Der ↓- Pfeil wird Ego- oder Selbst-Metavariable genannt, weil er auf das eigene funktionale Schema zeigt.



Figur 6: Festlegen der Stellvertreter für die Metavariablen

Bei der Instanzierung einer c-Struktur werden also alle Metavariablen ersetzt durch die Namen der f-Strukturen, bzw. durch die Namen der Knoten.



Figur 7: Vervollständigen der Instanzierung durch Koindizierung

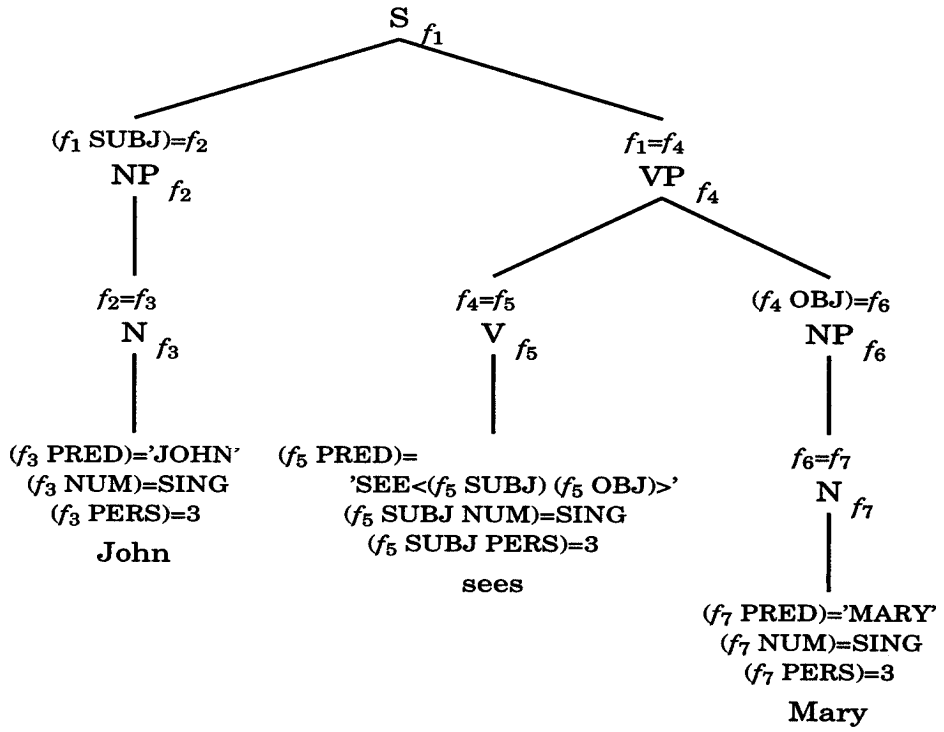
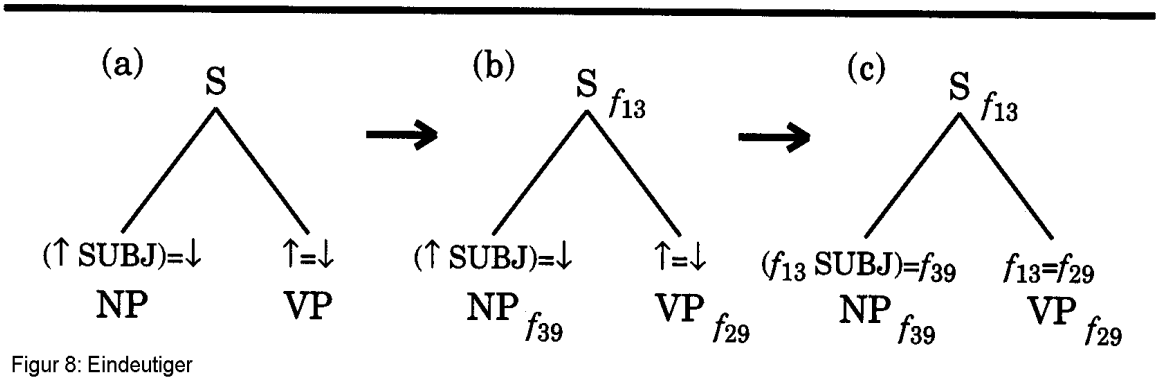


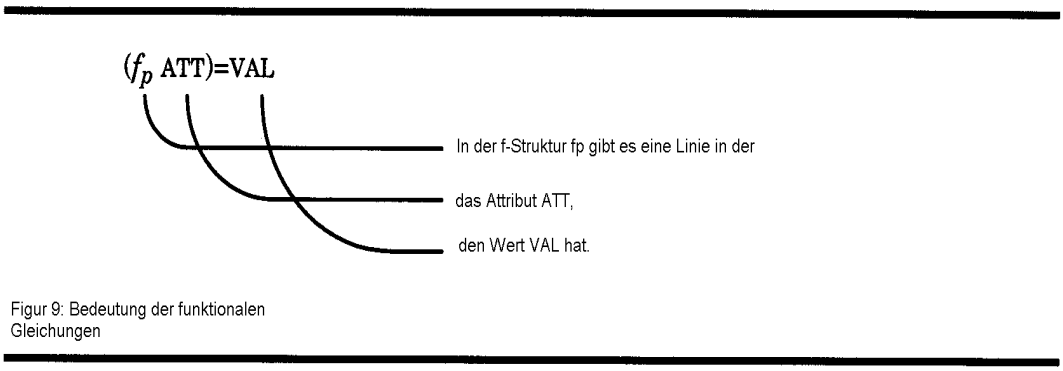
Abbildung 3: c-Struktur mit eingesetzten Variablen

Die Namen der Metavariablen f_1-f_n sind frei wählbar und unterliegen einzig der Bedingung, dass sie nicht mehrfach vorkommen.



Figur 8: Eindeutiger Instanzierungs-Algorithmus

Wir haben jetzt nicht mehr funktionale Schemata, sondern **funktionale Gleichungen (f-equation).** Wenn wir alle funktionalen Gleichungen aus der Baumstruktur herausschreiben und auflisten, erhalten wir eine **funktionale Beschreibung (f-description)**.



Figur 9: Bedeutung der funktionalen Gleichungen

Die Berechnung der f-Strukturen basiert nicht direkt auf den c-Strukturen, sondern auf den f-Beschreibungen. Trotzdem können wir sagen, dass bei der Instanzierung einer c-Struktur die f-Beschreibung die Verbindung zwischen c-Struktur und f-Strukturen bildet. Die hierarchische Information ist in der f-Beschreibung nicht mehr explizit enthalten

2.3 Von der funktionalen Beschreibung zur funktionalen Struktur

Der nächste Schritt ist, wie wir von dieser f-Beschreibung zu einer f-Struktur kommen. Nochmals: f-Strukturen sind Merkmalspaare in eckigen Klammern, in denen jeweils einem Attribut ein Wert zugeordnet wird. Dabei kann ein Wert selber auch wieder aus einer f-Struktur bestehen. Die Sache bleibt übersichtlicher, wenn wir ausserhalb der Klammern jeweils den Namen der Merkmalsstruktur notieren.

$$f_n \begin{bmatrix} A & f_m \begin{bmatrix} B & C \\ D & E \end{bmatrix} \\ F & G \\ H & I \end{bmatrix}$$

Abbildung 4: f-Struktur

Hier möchte ich auch noch in Erinnerung rufen, dass es in den Merkmalsstrukturen keine Rolle spielt, in welcher Reihenfolge die Wertepaare aufgelistet werden. Die Merkmalstrukturen 6 a und 6 b sind äquivalent.

$$\text{a.} \begin{bmatrix} A & B \\ C & D \\ E & \begin{bmatrix} F & G \\ H & I \end{bmatrix} \end{bmatrix} \qquad \text{b.} \begin{bmatrix} C & D \\ E & \begin{bmatrix} H & I \\ F & G \end{bmatrix} \\ A & B \end{bmatrix}$$

Abbildung 5: zwei gleichwertige f-Strukturen

Es ist nicht gestattet einem Attribut verschiedene Werte zuzuordnen. F-Strukturen sind nur unter dieser Bedingung akzeptabel. Die Attribut-Wert-Paare müssen eindeutig sein. f_p bei Abbildung 6 ist demnach keine konsistente f-Struktur.

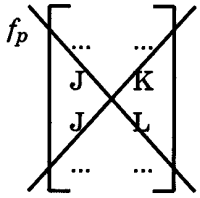


Abbildung 6: Inkonsistente f-Struktur

Es gibt noch weitere Wohlgeformtheitsbedingungen für f-Strukturen. Diese möchte ich aber erst im Zusammenhang mit der Semantik[1] in der LFG weiter unten beschreiben.

Kehren wir zurück zu unserer f-Beschreibung. Auch hier haben wir für ein entsprechendes Attribut einen zugeordneten Wert. Mit dem Ziel vor Augen, für einen Satz eine f-Struktur zu erhalten, bedeuten die f-Beschreibung, dass es in der f-Struktur eine Zeile gibt, in der die Attribut-Wert-Zuordnung der f-Beschreibung aufgelistet ist (siehe Figur 9).
Wenn wir verstehen, was die f-Beschreibung bedeutet, können wir sicher die f-Strukturen daraus ableiten.
(fn A) = B ⇒ fn [A B]
(fn C) = D ⇒ fn [C D]
Ganz wichtig ist beim Ableiten der f-Beschreibung in f-Strukturen, dass keine Informationen verloren gehen und keine Informationen zusätzlich aufgenommen werden. Wir können diese Bedingungen prüfen, indem wir mit jeder Zeile einer f-Struktur eine f-Beschreibung erschlagen können sollten. Geht das nicht, haben wir nicht die **minimale f-Struktur** der bearbeiteten f-Beschreibung vor uns. Dies ist am formalen Beispiel sehr einfach ersichtlich.
Nehmen wir die f-Gleichungen
(f_r A) = B
(f_r C) = D
Sie sind für den Moment unsere f-Beschreibung. Die Ableitung wird eine f-Struktur f_r ergeben. Für die erste f-Gleichung ergibt sich eine f-Struktur der Form
f_r [A B]
Jetzt gehen wir zur zweiten f-Gleichung unserer f-Beschreibung, in der steht, dass die f-Struktur f_r eine Zeile enthalten soll in der das Attribut C den Wert D hat. Ergänzen wir f_r mit dieser Information erhalten wir

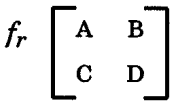


Abbildung 7: minimale f-Struktur

Dies ist die minimale f-Struktur zu unserer oben angenommenen f-Beschreibung. Im Gegensatz dazu ist

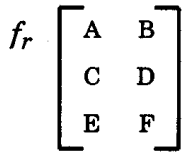


Abbildung 8

keine minimale f-Struktur, weil sie auch noch das Attribut E mit dem Wert F enthält. Dazu finden wir in unserer f-Beschreibung keine f-Gleichung. Oder anders gesagt, wir können mit der f-Beschreibung nicht verifizieren, ob (f_r E) = F wahr ist oder nicht.

2.3.1 Typen funktionaler Gleichungen

Wir unterscheiden hier vier verschiedene Typen von f-Gleichungen.[2]

1. Die einfachen f-Gleichungen von der Form (fn A) = B
2. Die f-Gleichungen der Form fn = fm
3. Die f-Gleichungen mit Subkategorisierung ((fn A) B) = C
4. Die f-Gleichungen der Form (fn A) = fm

1. Zuerst die einfachen f-Gleichungen. Daraus lassen sich sehr einleuchtend f-Strukturen bilden. Dazu fassen wir die Merkmalspaare der f-Gleichungen mit den gleichen Namen in eine f-Struktur mit diesem Namen zusammen.

c. (f_3 PRED)='JOHN'
d. (f_3 NUM)=SING
e. (f_3 PERS)=3
h. (f_5 PRED)='SEE<(f_5 SUBJ) (f_5 OBJ)>'
m. (f_7 PRED)='MARY'
n. (f_7 NUM)=SING
o. (f_7 PERS)=3

Abbildung 9: funktionale Beschreibung (ein Auszug davon)

Aus dieser f-Beschreibung lassen sich drei f-Strukturen aufbauen.

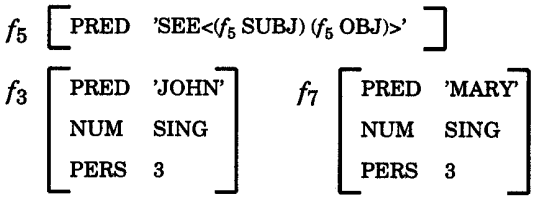


Abbildung 10: f-Strukturen zu den f-Gleichungen von Abbildung 10

2. Die f-Strukturen der Form $f_n = f_m$ bedeuten, dass zwei Variablen dieselbe f-Struktur sind. Oder dass dieselbe f-Struktur gleichzeitig zwei Namen bekommt. Machen wir dazu noch ein paar Beispiele aus dem Syntaxbaum (Abbildung 4).

---

[1] in den Prädikat-Argument-Strukturen
[2] Weiter unten beschreibe ich noch einen übergeordneten Typ funktionaler Gleichungen, die constraining equations

$f_2 = f_3$

$f_1 = f_4$

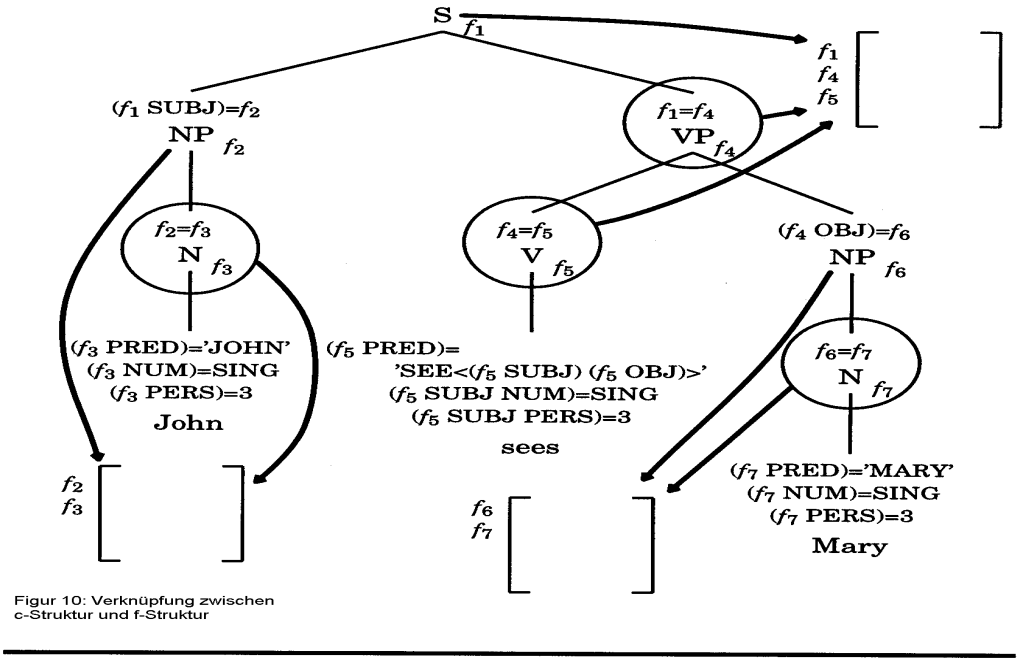
$f_4 = f_5$

$f_6 = f_7$

Dadurch können also auch f-Beschreibungen zusammengefasst[3] werden, bzw. in die gleiche Merkmalstruktur eingegliedert werden.

Weiter oben habe ich beschrieben, dass zu jedem Knoten der c-Struktur ein funktionales Schema gehört. Wir können jetzt erkennen, warum das nicht f-Strukturen sind. Die Bestandteile der c-Struktur können nicht 1:1 auf die f-Struktur abgebildet (**mapping**) werden. Es sind zwei verschiedene Darstellungsebenen, mit je eigenen Gesetzmässigkeiten.

Sehen wir uns das Beispiel nochmals genauer an, damit wir sehen wie die c-Strukturen in die f-Strukturen abgebildet werden.



Figur 10: Verknüpfung zwischen c-Struktur und f-Struktur

Die dazugehörenden f-Strukturen sehen so aus:

$$f_1, f_4, f_5 \begin{bmatrix} \text{PRED} & \text{'SEE<}(f_5\ \text{SUBJ})\ (f_5\ \text{OBJ})\text{>'} \end{bmatrix}$$

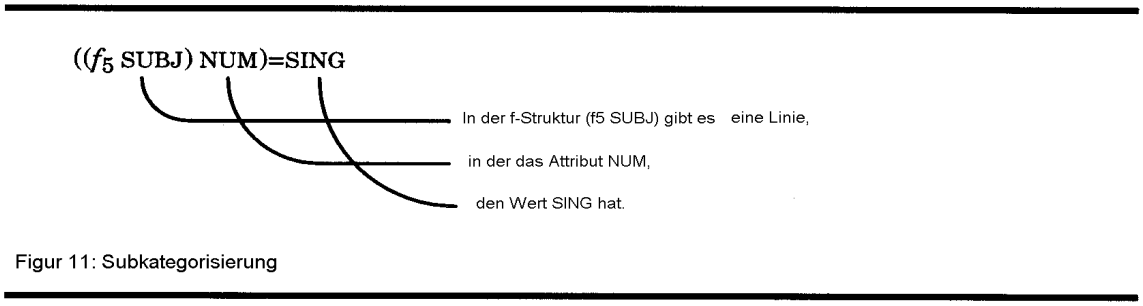
$$f_2, f_3 \begin{bmatrix} \text{PRED} & \text{'JOHN'} \\ \text{NUM} & \text{SING} \\ \text{PERS} & 3 \end{bmatrix} \qquad f_6, f_7 \begin{bmatrix} \text{PRED} & \text{'MARY'} \\ \text{NUM} & \text{SING} \\ \text{PERS} & 3 \end{bmatrix}$$

Abbildung 11: f-Struktur zu Figur 10

3. Die nächsten f-Gleichung, die wir uns ansehen wollen, haben zwei Attribute.
((fn A) B)= C

[3] Wir könnten auch sagen, dass diese f-Strukturen so den gleichen Namen erhalten.

Wie wir wissen, darf jedes Attribut nur einen Wert haben. Die Konsequenz dieser Bedingung ist, dass wir für jeden Wert einen Pfad angeben können, wo in einer bestimmten f-Struktur wir ihn finden, bzw. wo er stehen soll. Sehen wir uns dazu das Beispiel (Figur 11) an.



Figur 11: Subkategorisierung

Diese f-Gleichungen erlauben die Darstellung der Subkategorisierung[4] in den f-Strukturen. Die erweiterte f-Struktur sieht so aus:

$$f_1 \begin{bmatrix} \text{PRED} & \text{'SEE<}(f_5\ \text{SUBJ})\ (f_5\ \text{OBJ})\text{>'} \\ \text{SUBJ} & \begin{bmatrix} \text{NUM} & \text{SING} \end{bmatrix} \end{bmatrix}$$

$$f_2 \begin{bmatrix} \text{PRED} & \text{'JOHN'} \\ \text{NUM} & \text{SING} \\ \text{PERS} & 3 \end{bmatrix} \qquad f_6 \begin{bmatrix} \text{PRED} & \text{'MARY'} \\ \text{NUM} & \text{SING} \\ \text{PERS} & 3 \end{bmatrix}$$

Abbildung 12: f-Struktur mit Subkategorisierung

4. Jetzt möchte ich noch die f-Gleichung der Form (fn A)= fm beschreiben. Sie können f-Strukturen verknüpfen, können aber auch der Kontrolle dienen.
   Nehmen wir die f-Gleichungen
   (f_1 SUBJ) = f_2
   (f_4 OBJ) = f_6
   Zuerst zum Verknüpfen. Die zweite dieser f-Gleichungen lässt sich leicht in die bestehende f-Struktur einbauen.

$$f_1 \begin{bmatrix} \text{PRED} & \text{'SEE<}(f_5\ \text{SUBJ})\ (f_5\ \text{OBJ})\text{>'} \\ \text{SUBJ} & \begin{bmatrix} \text{NUM} & \text{SING} \\ \text{PERS} & 3 \end{bmatrix} \\ \text{OBJ} & f_6 \begin{bmatrix} \text{PRED} & \text{'MARY'} \\ \text{NUM} & \text{SING} \\ \text{PERS} & 3 \end{bmatrix} \end{bmatrix}$$

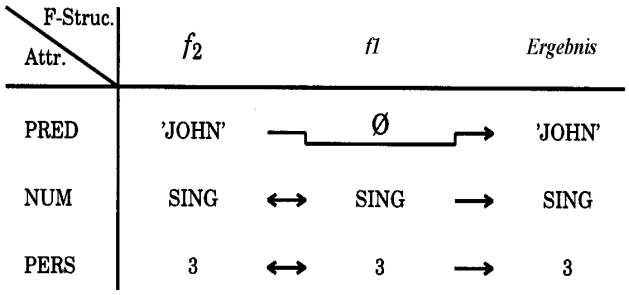
$$f_2 \begin{bmatrix} \text{PRED} & \text{'JOHN'} \\ \text{NUM} & \text{SING} \\ \text{PERS} & 3 \end{bmatrix}$$

Abbildung 13

Die beiden f-Strukturen verschmelzen zu einer einzigen f-Struktur, oder mit einem anderen Wort: sie unifizieren. Dadurch kommen wir unserem Ziel, für einen Satz eine einzige f-Struktur zu haben, einen entscheidenden Schritt näher.

[4] oder der Pfadangabe

Die Gleichung (f_1 SUBJ) = f_2 soll jetzt auch noch mit der bestehenden f-Struktur unifiziert werden. Dabei können wir einen Kontrollmechanismus beobachten.
Da die f-Struktur f_2 gleichgesetzt wird mit der f-Struktur zum Attribut SUBJ in der f-Struktur f_1 müssen, die sich entsprechenden Attribut-Wert-Paare, übereinstimmen. Dazu werden die Werte zu jedem Attribut der verschiedenen f-Strukturen verglichen.



Figur 12: Werteabgleich zwischen den f-Strukturen

Die Kontrolle beim Bilden von f-Strukturen in LFG geschieht also bei der Unifikation von zwei f-Strukturen. Enthalten die f-Strukturen nämlich sich widersprechende Attribut-Wert-Paare, dann schlägt die Unifikation fehl. Widersprechen sich die Attribut-Wert-Paare nicht, werden die beiden f-Strukturen unifiziert und die resultierende f-Struktur enthält schliesslich alle Informationen der ursprünglichen f-Beschreibung.

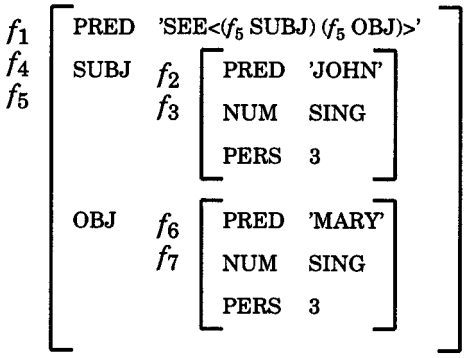


Abbildung 14

Das Ziel ist erreicht, wir haben aus unserem Satz eine c-Struktur und eine f-Struktur erhalten.

3 Keine Kapitulation, sondern …

Ich möchte kurz rekapitulieren, bevor wir noch weitere Aspekte der LFG anschneiden.
Als Ausgangslage hatten wir einen Satz, ein Lexikon und die Grammatikregeln. Die Lexikoneinträge sind wie Merkmalstrukturen organisiert und die Grammatikregeln entsprechen Phrasenstrukturregeln, die mit funktionalen Schemata annotiert sind. Merkmalstrukturen ordnen jedem Attribut einen Wert zu und die Konstituentenregeln schaffen eine Hierarchie bzw. den Syntaxbaum. Die Verbindung dieser beiden Ebenen gelingt dank den funktionalen Schemata in der c-Struktur und den f-Beschreibungen. Diese Mechanismen sind LFG spezifisch.

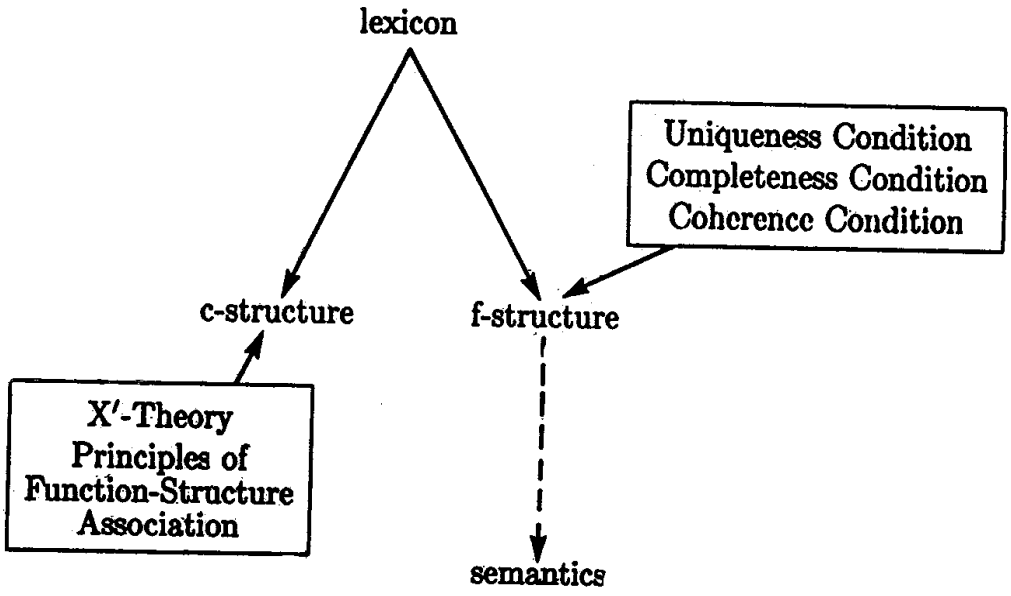


Abbildung 15: LFG-Übersicht nach Sells (Seite 137)

Wichtig scheint mir noch zu sehen, dass die Grammatikregeln übergenerierend sind und die theoretisch vielen Möglichkeiten durch die Lexikoneinträge, bzw. ihren funktionalen Schemata wieder eingeschränkt werden. Im nächsten Teil dieser Arbeit wird es um diese Einschränkungen gehen.

4 Formale Bedingungen für funktionale Strukturen

Damit ein Satz in LFG als grammatisch korrekt bezeichnet werden kann, muss dieser zwei Kriterien erfüllen. Erstens muss die Grammatik in der Lage sein, zu diesem Satz eine c-Struktur anzugeben und zweitens auch noch eine wohlgeformte f-Struktur. Diese f-Struktur muss dabei zur c-Struktur passen und daneben auch drei formalen Prinzipien für f-Strukturen genügen. **Konsistenz**, **Kohärenz** und **Vollständigkeit**.

4.1 Konsistenz

Konsistenz (oder uniqueness condition) ist eine Eigenschaft von f-Beschreibungen. Wir sagen eine f-Beschreibung sei konsistent, wenn es gelingt für diese f-Beschreibung eine f-Struktur f_n zu schreiben, mit der alle f-Gleichungen erfüllt werden können. Oder anders gesagt: Eine f-Beschreibung ist

konsistent, wenn sich die f-Gleichungen nicht widersprechen. Das folgende Beispiel zeigt, wann eine f-Beschreibung inkonsistent ist. Wie hier dargestellt:
(fn A) = B
(fn A) = C
Unter diesen Bedingungen ist es nicht möglich, eine f-Struktur zu aufzubauen.
Bei den f-Strukturen gilt Konsistenz als übergeordnetes Prinzip. Attribut-Wert-Paare dürfen sich nicht widersprechen.

LFG benützt **grammatische Funktionen** wie SUBJ, OBJ usw. Die f-Strukturen zeigen, wie diese grammatischen Funktionen in einem Satz zusammenhängen. Dies geschieht in den f-Gleichungen mit dem Attribut PRED. Hier können wir die semantisch motivierten Beziehungen zwischen den grammatischen Funktionen herauslesen. Ausgedrückt werden die Beziehungen zwischen <Klammern> wieder mit f-Struktur-Variablen. Die Reihenfolge der f-Struktur-Variablen spielt keine Rolle. Diese Prädikat-Argument-Strukturen beschreibt also die Beziehungen zwischen den grammatischen Funktionen.
Dazu ein Beispiel aus einem LFG-Lexikon:
write V S-EN (↑ PRED) = 'WRITE<(↑ SUBJ) (↑ OBJ)>'.[5]
Die Argumente dieser Prädikat-Argument-Struktur sind semantisch zu verstehen. Alles zwischen den einfachen ´Anführungszeichen´ ist semantische strukturierte Information. Aus diesen Argumenten wollen wir keine f-Strukturen ableiten! Aber sie dienen einem weiteren Kontrollmechanismus in der LFG. Wird ein Satz analysiert und mit der Grammatik eine c-Struktur gezeichnet, dann muss der f-Struktur dieses Satzes für alle Argumente eine untergeordnete f-Struktur vorhanden sein. Dies tönt ziemlich kompliziert, ist aber am Beispiel leicht nachzuvollziehen.
bite V S-ED (↑ PRED)='BITE<(↑ SUBJ) (↑ OBJ)>'.[6]
Das Verb "bite" verlangt also in der f-Struktur die beiden grammatischen Funktionen SUBJ und OBJ als Attribute. Die vollständige f-Struktur des Satzes:
"The dog bites the student "
sieht demnach so aus (Ausschnitt aus dem GWB):

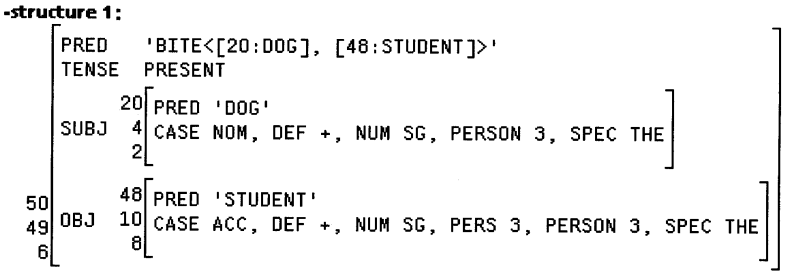


Abbildung 16: f-Struktur aus dem GWB

4.2 Vollständigkeit (completness)

Vollständigkeit bedeutet, dass für jedes Argument in der Prädikat-Argument-Struktur eine untergeordnete f-Struktur vorhanden sein muss. Es darf kein Argument fehlen. Mit diesem Prinzip kann sichergestellt werden, dass bestimmte Konstituenten vorhanden sind.
Nehmen wir den Satz:

[5] Dieser Eintrag müsste natürlich noch weiter spezifiziert werden, da "write" auch ohne die grammatische Funktion OBJ auskommen kann. Zum Beispiel in: "John writes."
[6] Hier gilt die gleiche Bemerkung wie bei Fussnote 1

"John likes."
Der Lexikoneintrag von "likes" sieht so aus:

like V S-ED (↑ PRED) = 'LIKE<(↑ SUBJ) (↑ OBJ)>'
(↑ SUBJ NUM) = SING
(↑ SUBJ PERS) = 3.

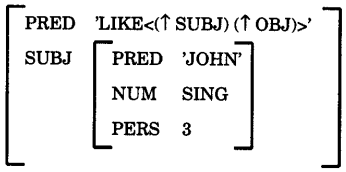


Abbildung 17: f-Struktur zum Satz John likes

Wir stellen fest, dass der f-Struktur das Attribut OBJ fehlt. Sie ist nicht vollständig.

4.3 Kohärenz (coherence)

Kohärenz können wir als Gegenstück zur Vollständigkeit auffassen. Hier wird sichergestellt, dass in der f-Struktur eines Satzes keine Konstituente zuviel vorkommen kann.
Falls es in der f-Struktur eines Satzes eine solche überflüssige Konstituente (oder untergeordnete f-Struktur) gibt, sagt man, sie decke sich nicht mit der Prädikat-Argument-Struktur. In diesem Fall wird das System den Satz nicht akzeptieren.
Gegeben sei der Satz:
"John falls Mary."
Der Lexikoneintrag von "falls" ist

fall V S-ED (↑ PRED) = 'FALL<(↑ SUBJ)>
(↑ NUM) = SING
(↑ PERS) = 3.

Für den Satz "John falls Mary" erhalten wir die f-Struktur:

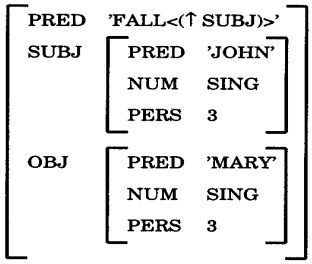


Abbildung 18: f-Struktur für den Satz John falls Mary

Im Prädikat ist keine grammatische Funktion OBJ vorgesehen. Deshalb ist diese f-Struktur nicht kohärent.
Eine Ausnahme zu diesen Prinzipien ist die NULL-Funktion ∅. Darauf möchte ich hier aber nicht näher eingehen.[7]

5 Erzwingende f-Gleichungen (constraining equations)

Im nächsten Abschnitt möchte ich die Bedeutung der constraining equations ($=_c$) erklären. Mechanisch gesehen, werden die funktionalen Gleichungen in zwei Gruppen aufgeteilt. Die constraining equations (erzwingende Gleichungen) und die nicht erzwingenden Gleichungen. Beim Abarbeiten einer funktionalen Beschreibung wird zuerst aus den nicht erzwingenden Gleichungen eine minimale f-Struktur gebildet.
Hier eine kurze f-Beschreibung:

(fn A) = B
(fn C) = D
(fn C) $=_c$ D

Aus den ersten beiden f-Gleichungen entsteht die f-Struktur:

$$f_n \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

Die constraining equations dienen jetzt zur Kontrolle der gebildeten f-Struktur. Können alle constraining equations anhand der minimalen f-Struktur als wahr bezeichnet werden, ist die minimale f-Struktur konsistent, ansonsten ist sie inkonsistent.
(fn C) $=_c$ D kann aufgrund der vorher entstandenen f-Struktur als wahr bezeichnet werden. Durch die constraining equations kann sichergestellt werden, dass bestimmte Werte zu bestimmten Attributen zwingend vorhanden sein müssen.

a.	*a*	Det	(↑ DEF) = −
b.	*girl*	N	(↑ PRED) = 'girl' (↑ PERS) = 3 (↑ NUM) = SG
c.	*me*	N	(↑ PRED) = 'PRO' (↑ PERS) = 1 (↑ NUM) = SG (↑ CASE) $=_c$ ACC

Abbildung 19: Lexikoneinträge nach Sells

Bei diesem Lexikoneintrag erzwingt die constraining equation in der letzten Zeile, dass beim Attribut CASE der Wert ACC eingesetzt wird.

6 Subjektkontrolle / Objektkontrolle

In LFG gibt es verschiedene grammatische Funktionen, darunter auch Komplemente (COMP) und Adjunkte (ADJUNCT). Komplemente sind obligatorische Elemente einer Prädikat-Argument-Struktur. Wir finden sie in den subkategorisierenden Lexikoneinträgen beim Attribut PRED, falls ein Verb ein Komplement verlangt. Wenn ein Komplement semantisch vollständig ist, nennen wir dieses ein geschlossenes Komplement. Offene Komplementen (XCOMP) fehlt ein Element. Dieses unausgesprochene Element (Subjekt oder Objekt) wird dann durch ein anderes Argument der Prädikat-Argument-Struktur kontrolliert. Sehen wir uns dazu ein Beispiel an.

[7] dazu steht mehr bei Neidle Seite 229 ff.

seem V S-ED (↑ PRED) =' SEEM<(↑ XCOMP) (↑ SUBJ)>'
((↑ XCOMP SUBJ) = (↑ SUBJ)).

Offene Komplemente wie bei seem verlangen nach dem Subjekt oder dem Objekt des Hauptprädikates mit dem sie verknüpft sind. Diese Beziehung wird durch eine Kontroll-Gleichung ausgedrückt. Hier setzt die Kontroll-Gleichung

(↑ XCOMP SUBJ) = (↑ SUBJ).

das Subjekt des Hauptprädikates gleich dem Subjekt des Komplements. Formal geschieht dieses Gleichsetzen entweder durch Verbinden der beiden f-Strukturen mit einer Bogenlinie (wie im GWB) oder durch gleiche Benennung der f-Strukturen.

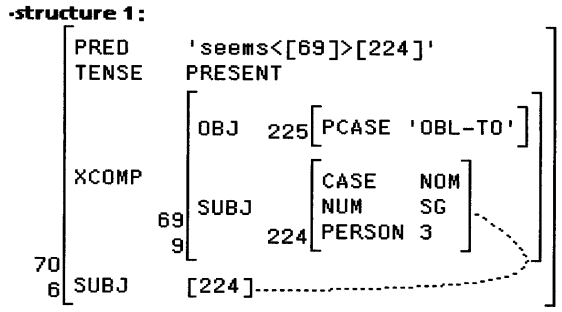


Abbildung 20: f-Struktur mit Kontrollverbindung aus dem GWB

Diese Verbindung zweier f-Strukturen zur Subjekts-, bzw. zur Objektskontrolle nennen wir funktionale Kontrolle.

persuade V S-ED (↑ PRED)='PERSUADE<(↑ XCOMP) (↑ OBJ) (↑`SUBJ)>'
((↑ XCOMP SUBJ)= (↑ OBJ)).

Beim Lexikoneintrag von "persuade" wird in der Kontroll-Gleichung das Subjekt des offenen Komplements gleichgesetzt mit dem Objekt des Hauptprädikats. Hier findet also eine Objektkontrolle statt.

Die Subjektkontrolle und die Objektkontrolle sind weitere Mechanismen, an denen wir die kontrollierende und einschränkende Wirkung des Lexikons in LFG erkennen können.

Das Subjekt des Komplements ist in der c-Struktur nicht präsent. Es erscheint einzig durch die Prädikat-Argument-Struktur vom Verb und deren Kontroll-Gleichung im Lexikoneintrag.

Schlussbemerkungen

Ich hoffe mit dieser Arbeit zum besseren Verständnis der LFG beitragen zu können. Ein sehr wertvoller Link zur weiteren Lektüre über LFG ist sicher die LFG-Homepage. Dort sind neben einführenden Texte auch weiterführende Artikel zu finden. Man kann da unter anderem nachlesen wie in LFG das Passiv implementiert werden kann. LFG entwickelt sich immer weiter. Durch das erhältliche GWB sind alle herzlich eingeladen an dieser Entwicklung aktiv teilzunehmen.

7 Glossar

CHART (KETTE)
Eine Chart besteht aus Knoten und deren Verbindungen. Der Chart-Parser versucht mit den Regeln die Verbindungen zwischen den Knoten zu Konstituenten zusammenzufassen. Beim GWB ist auch ein Chart-Parser eingebaut. Er arbeitet wahlweise Top down oder Bottom up.

CONSTITUENTE STRUCTURE (KONSTITUENTEN STRUKTUR)
Die c-Struktur ist die Baumstruktur in LFG. Sie besteht aus Knoten und Verbindungen und ist dadurch hierarchisch. In LFG enthält die Baumstruktur aber noch mehr Informationen, die funktionalen Schemata.

CONTROL EQUATION (KONTROLL-GLEICHUNG)
Die Kontroll-Gleichung steht im Lexikoneintrag bei bestimmten Verben. Kontroll-Gleichungen stellen sicher, dass innerhalb eines Satzes die grammatischen Funktionen SUBJ oder OBJ die gleiche semantische Bedeutung erhalten. (Subjektkontrolle oder Objektkontrolle)

FUNCTIONAL DESCRIPTION (FUNKTIONALE BESCHREIBUNG)
Das ist die Menge aller funktionalen Gleichungen nach dem Koindizieren der funktionalen Schemata einer c-Struktur.

FUNCTIONAL EQUATION (FUNKTIONALE GLEICHUNG)
Funktionale Gleichungen entstehen aus den funktionalen Schemata nachdem letztere koindiziert wurden. Aus ($\uparrow$ SUBJ) = $\downarrow$ wird (f_1 SUBJ) = f_2 .
Wir unterscheiden verschiedene Typen funktionaler Gleichungen:

1. (f_n A) = B schaffen Merkmalstrukturen der Form f_n [A B]
2. f_n = f_m setzen zwei Merkmalstrukturen gleich, bzw. ergeben Doppelnamen für eine Merkmalstruktur.
3. ((f_n A) B) = C schaffen verschachtelte Merkmalstrukturen der Form f_n [A [B C]]
4. (f_n A) = f_m ordnen einem Attribut A den Wert f_m (eine weitere f-Struktur) zu.

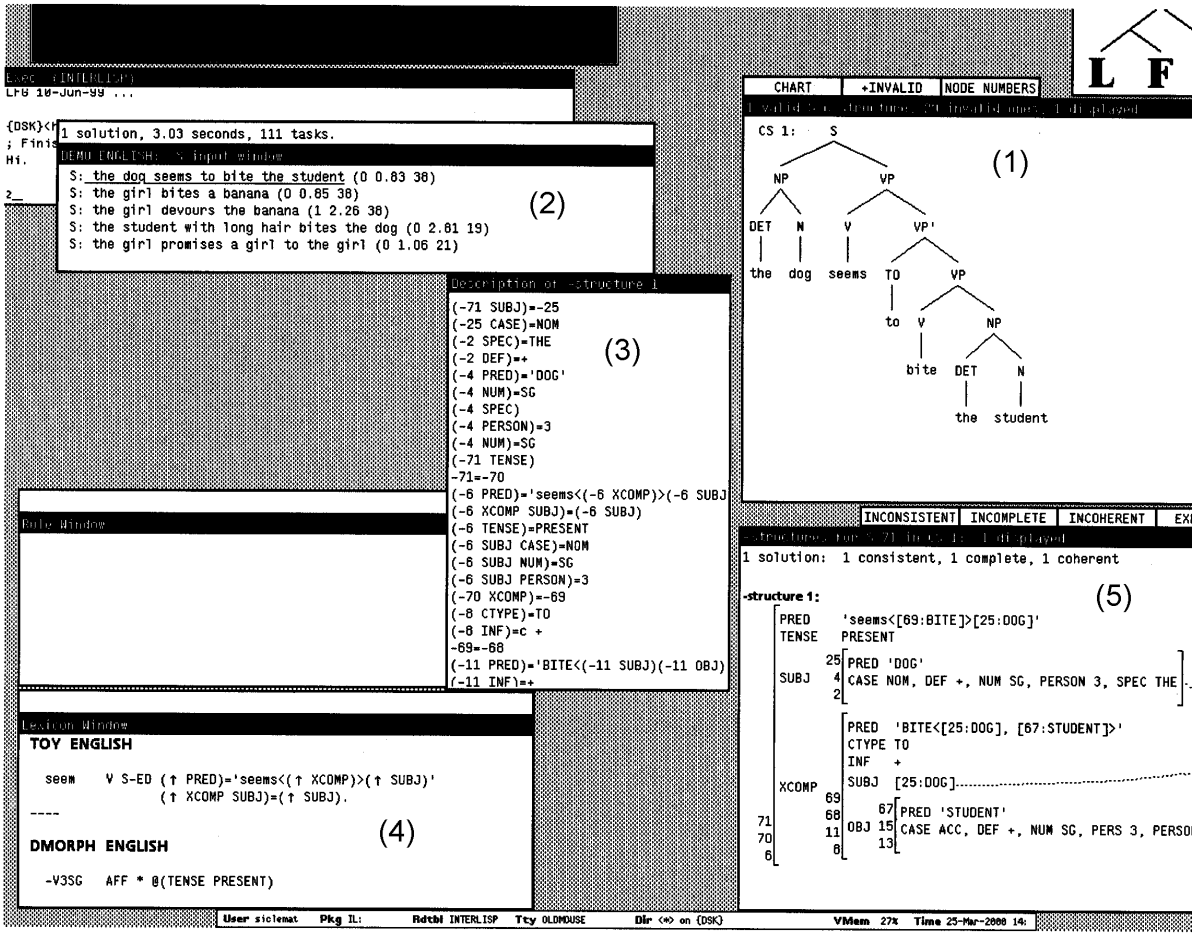
FUNCTIONAL SCHEMATA (FUNKTIONALES SCHEMA)
In den grammatischen Regeln enthalten diese funktionalen Schemata die Metavariablen $\uparrow$ und $\downarrow$. In diesem Sinne sind die funktionalen Schemata noch keine fertigen Merkmalstrukturen. Sie bilden die Ausgangspunkte zur Instanzierung einer c-Struktur und sind deshalb ein verbindendes Element zwischen c-Struktur und f-Struktur.

FUNCTIONAL STRUCTURE (FUNKTIONALE STRUKTUR)
Funktionale Strukturen (f-Strukturen) sind Merkmalstrukturen im Sinne der Unifikationsgrammatik. Sie bestehen aus Attributen und deren Werten. Zu den Attributen gehören die grammatischen Funktionen wie SUBJ, OBJ und PRED sowie auch morphosyntaktische Merkmale wie CASE, NUM und TENSE.
Ein Attribut kann als Wert ein Symbol, eine semantische Form (die Prädikat-Argument-Struktur) zwischen 'Anführungszeichen' , oder wieder eine f-Struktur haben.

F-Strukturen müssen drei Kriterien (Konsistenz, Kohärenz, Vollständigkeit) genügen, damit sie als konsistent bezeichnet werden können.

GWB

GWB oder Grammar Writer's Workbench ist eine Programmumgebung, die entwickelt wurde zum Schreiben und Testen von LFG basierten Grammatiken. Das System wurde in den 80er Jahren geschrieben und ist bis heute weiterentwickelt worden. Es erlaubt es Syntaxregeln, Lexikoneinträge (4) und einfache morphologische Regeln zu schreiben und zu prüfen. Zur Analyse der eingegebenen Sätze oder Strings gibt das System Informationen in vier verschiedenen Fenstern an: c-Struktur (1), Chart (ist hier nicht zu sehen), f-Struktur (5) und f-Beschreibung (3). Das Fenster (2) ist das Satzeingabe-Fenster. Hier werden die zu analysierenden Sätze eingegeben.



Das System ermöglicht Generalisierungen für häufige Konstituenten und funktionale Elemente durch sogenannte Makros (4) und Templates. Dadurch kann die Entwicklung neuer Grammatiken erheblich vereinfacht werden.

In GWB können verschiedene Grammatiken einfach kombiniert werden, bzw. zusammengesetzt werden. So kann man in GWB leicht mit verschiedenen Versionen einer Grammatik experimentieren.

GRAMMATISCHE FUNKTION

In LFG werden grammatische Funktionen wie SUBJ, OBJ oder COMP als lexikalische Funktionen betrachtet. Wir finden grammatische Funktionen in den funktionalen Schemata, in den f-Strukturen (als Attribute) und in den Prädikaten (als semantische Elemente).

GRAMMATISCHE REGELN
Die grammatischen Regeln sind stark beeinflusst von den kontextfreien Phrasenstrukturregeln der TG nach Chomsky. (S $\rightarrow$ NP VP) Es sind auch in LFG Ersetzungsregeln, die eine hierarchische Struktur schaffen und enthalten zusätzlich funktionale Schemata. Sie unterliegen einer Version der X'-Theorie. Die grammatischen Regeln ohne funktionale Schemata sind übergenerierend.

INSTANZIERT
Die Instanzierung (Umwandlung einer c-Struktur in eine f-Struktur) geschieht in mehreren Schritten. Die Knoten einer c-Struktur werden zuerst koindiziert (benannt). Dadurch entstehen die funktionalen Gleichungen. Aus diesen funktionalen Gleichungen wird eine f-Struktur aufgebaut.

CONSISTENCE (KONSISTENZ)
Konsistenz ist eine Wohlgeformtheitsbedingung, die jede f-Struktur in LFG erfüllen muss. Die Konsistenz-Bedingung schreibt vor, dass in einer f-Struktur jedes Attribut genau einen Wert haben muss. Die Werte dürfen sich nicht widersprechen.

COHERENCE (KOHÄRENZ)
Kohärenz ist eine Wohlgeformtheitsbedingung für f-Strukturen in LFG. Alle grammatischen Funktionen wie SUBJ oder OBJ in einer f-Struktur müssen von den grammatischen Funktionen eines lokalen (= in derselben f-Struktur) Prädikates gedeckt sein.

COINDEXING (KOINDIZIERT)
Die Knoten der c-Struktur werden koindiziert, d.h. alle (noch) leeren f-Strukturen erhalten einen Namen. Die Zuteilung ist willkürlich, aber muss eindeutig sein. Jeder Knoten muss eine eigene f-Struktur erhalten.

LFG
Bitte den entsprechenden Eintrag im Bussmann lesen.

LEXICAL ITEM (LEXIKONEINTRAG)
Das Lexikon ist in LFG zentral. Lexikoneinträge dienen der Auflistung aller verfügbaren Wörter einer Sprache. Daneben übernehmen sie in LFG auch eine Kontrollfunktion. Wichtige Sprachphänomene werden in LFG im Lexikon definiert, z.B. das Passiv (siehe dazu auch Sells Seiten 160 ff).

MAPPING (ABBILDUNG)
Mapping ist der Fachausdruck für das Abbilden einer c-Struktur auf eine f-Struktur.

MINIMALE FUNKTIONALE STRUKTUR
Dieser Ausdruck bedeutet, dass jede f-Struktur kohärent sein muss. Es dürfen keine "nicht beweisbaren" Informationen aufgenommen werden.

PRÄDIKAT-ARGUMENT-STRUKTUR
Damit ist hier die Aufzählung aller grammatischen Funktionen zwischen <Klammern> in der Wert-Position beim Attribut PRED gemeint.

COMPLETENESS (VOLLSTÄNDIGKEIT)
Vollständigkeit ist eine Wohlgeformtheitsbedingung für f-Strukturen in LFG, die besagt, dass für alle grammatische Funktionen eines lokalen Prädikates je ein Attribut vorhanden sein muss.

8 Literaturverzeichnis

Schneider, Gerold (1999): Vorlesung "Formale Grammatiken und Syntaxanalyse" Vorlesungen 7 + 8. http://www.ifi.unizh.ch/cl/gschneid/SyntaxVorlesung

Sells, Peter (1985): Lectures on Contemporary Syntactic Theories. Stanford: University of Chicago Press.

Neidle, Carol. (1994): `Lexical-Functional Grammar'. Encyclopedia of Language and Linguistics. New York: Pergamon Press.

Weisweber, Wilhelm (1997): Prolog: Logische Programmierung in der Praxis. (S.317-340) Thomson.

Wescoat, Michael T (19**): Practical Instructions for Working with the Formalism of LFG

Weitere einführende Literatur (auch zum Herunterladen) ist unter folgender Adresse zu erreichen:

http://clwww.essex.ac.uk/LFG/Introductions.html