PTQ, LF, TFL
— zum theoretischen Status einer "Übersetzung in eine formale Semantiksprache"

Frank Richter   Manfred Sailer
{fr,mf}@sfs.nphil.uni-tuebingen.de

20.12.1996

1 Elemente einer Grammatik in getypter Merkmalslogik (TFL-Grammatik)

1.1 Grundbausteine

Die folgenden zwei Grundkomponenten sind im Prinzip ausreichend, um die gesamte Grammatik aufzuschreiben ((King 1989), (King 1994)). Um eine leichtere und intuitivere Notation zu ermöglichen, werden jedoch Erweiterungen eingeführt.

Die Signatur
Die Signatur legt fest, wie die Objekte im linguistischen Universum strukturiert sind und was die syntaktischen Zeichen sind, mit denen Aussagen über sie gemacht werden können.
Sie beinhaltet:

1. die Sorten (d.h. die Arten von Objekten)
2. die Attribute (d.h. die relevanten Merkmale)
3. die *appropriateness conditions* (d.h. welche Merkmale sind bei Objekten welcher Sorte vorhanden)

Die Theorie
Die Theorie enthält Beschreibungen. In einem Modell soll jedes Objekt alle Beschreibungen der Theorie erfüllen.
Eine Beschreibung ist

1. eine Sortenbeschreibung (:$\sim$*phrase* oder :NONHEAD-DTR$\simeq$:HEAD-DTR)
2. eine Pfadgleichung (:HEAD $\approx$:HEAD-DTR HEAD)
3. die Negation, Konjunktion, Disjunktion oder Implikation von Beschreibungen.

1.2 Erweiterungen

Sortenhierarchie
Zusätzlich zu Sorten, die das Universum partitionieren, werden oft Übersorten verwendet, die mehrere minimale Sorten zusammenfassen. (vergl. (Richter and Sailer 1995))
(z.B. zu den minimalen Sorten *word* und *phrase* wird eine Übersorte *sign* angenommen.)

Kovariationsregeln
Kovariationsregeln werden in der Regel in der Form von *Lexikalischen Regeln* verwendet. Eine Regel $A \mapsto B$ bedeutet vereinfacht: wenn es im Modell ein Objekt gibt, das die Beschreibung A erfüllt, dann gibt es auch eines, das die Beschreibung B erfüllt. (vergl. (Meurers 1994))
Daraus ergibt sich ein TFL-Lexikon etwa als:

$$
\begin{array}{ll}
:\sim \mathit{word} \quad \rightarrow & \\
& LE_1 \\
& \vee \ldots \vee LE_n \\
& \vee \left(\boxed{1} \left[\text{LEX-RULE-IN}\ \boxed{2}\ \mathit{word}\right] \wedge \left(\texttt{lexical-rule}_1(\boxed{2},\boxed{1})\right.\right. \\
& \qquad\qquad\qquad\qquad\qquad\qquad \vee \ldots \\
& \qquad\qquad\qquad\qquad\qquad\qquad \left.\left.\vee\ \texttt{lexical-rule}_m(\boxed{2},\boxed{1})\right)\right)
\end{array}
$$

wobei $LE_1, \ldots, LE_n$ die Basiseinträge sind und $\texttt{lexical-rule}_1$, $\ldots\texttt{lexical-rule}_m$ die Lexikalischen Regeln.

Relationen
In einer traditionellen TFL-Grammatik können Relationen über Sorten emuliert werden (vergl. (Ait-Kaci 1984), (Carpenter 1992)). Um sie direkt als Relationen hinschreiben zu können, bedarf es einigem definitorischen Aufwand (vergl. (Richter 1996a, Kapitel 2)).
Bsp.:

$$
\begin{array}{l}
\texttt{append}(x, y, z) := \\
\qquad x \sim \mathit{elist} \\
\qquad \wedge\ y \approx z \\
\texttt{append}(x, y, z) := \\
\qquad x\text{FIRST} \approx z\text{FIRST} \\
\qquad \wedge\ \texttt{append}(x\text{REST},\ y,\ z\text{REST})
\end{array}
$$

2 Die Syntax von IL

2.1 Typen

(1) *Typen bei Montague:*

t is a type.

e is a type.

if a and b are types, then $\langle a, b\rangle$ is a type.

if a is a type, then $\langle s, a\rangle$ is a type.

(2) *Typen:*

type *truth*

entity

c(omplex)-type IN *type*
OUT *type*

i(ntensional)-type TYPE *type*

2.2 Syntaxregeln

(3) *Hilfskonstrukt:*

number NUM *list-of-plus*

Hilfssorte:

list-of-plus

elist-of-plus

nelist-of-plus FIRST *plus*
REST *list-of-plus*

Variablen und Konstanten sind durchnummeriert. ($c_{a,n}$, $v_{a,n}$, für jeden Typ a und für jede natürliche Zahl n.)

(4) *Die Syntaxregeln bei Montague:*

The set of meaningfull expressions of type a, called ME_a, is defined recursively as follows:

1. Every variable of type a is in ME_a.
2. Every constant of type a is in ME_a.
3. If $\alpha \in ME_a$ and u is a variable of type β, then $\lambda u \alpha \in ME_{\langle b,\, a\rangle}$.
4. If $\alpha \in ME_{\langle a,\, b\rangle}$ and $\beta \in ME_a$, then $\alpha(\beta) \in ME_a$.
5. If α and β are both in ME_a, then $\alpha = \beta \in ME_t$.

6. – 10. If ϕ and ψ are in ME_t, then the following are also in ME_t:

6. $\neg\phi$
7. $[\phi \wedge \psi]$
8. $[\phi \vee \psi]$
9. $[\phi \rightarrow \psi]$
10. $[\phi \leftrightarrow \psi]$

11. If $\phi \in ME_t$ and u is a variable of any type, then $\forall u \phi \in ME_t$.
12. If $\phi \in ME_t$ and u is a variable of any type, then $\exists u \phi \in ME_t$.
13. If $\phi \in ME_t$, then $\Box\phi \in ME_t$.
14. If $\phi \in ME_t$, then $\mathbf{W}\phi \in ME_t$.
15. If $\phi \in ME_t$, then $\mathbf{H}\phi \in ME_t$.
16. If $\alpha \in ME_a$, then $\hat{}\alpha \in ME_{\langle s,\, a\rangle}$.
17. IF $\alpha \in ME_{\langle s,\, a\rangle}$, then $\check{}\alpha \in ME_a$.

(5) *Syntaxregeln von IL*

m(eaningfull-)e(xpression) TYPE *type*

var(iable) NUMBER *number*
const(ant) NUMBER *number*
abstr(action) VAR *var*
ARG *me*
appl(ication) FUNC *me*
ARG *me*
equ(ation) ARG1 *me*
ARG2 *me*
l(ogical)-const1 ARG *me*

negation
necessary
future
past

l-const2 ARG1 *me*
ARG2 *me*

disjunction
conjunction
implication
equivalence

quantifier VAR *var*
ARG *me*

universal
existential

up ARG *me*
down ARG *me*

(6) *Typbeschränkungen:*

1. :∼*abstr* →
   :TYPE IN≈ :VAR TYPE
   ∧ :TYPE OUT ≈ :ARG TYPE
2. :∼*appl* →
   :FUNC TYPE IN ≈ :ARG TYPE
   ∧ :FUNC TYPE OUT ≈ :TYPE
3. :∼*equ* →
   :ARG1 TYPE ≈ :ARG2 TYPE
4. :∼*l-const1* →
   :ARG TYPE∼*truth*
5. :∼*l-const2* →
   :ARG1 TYPE∼*truth*
   ∧ :ARG2 TYPE∼*truth*
6. :∼*quantifier* →
   :ARG TYPE∼*truth*

7. $:\sim up \quad \rightarrow$

$:\text{TYPE}\sim \textit{i-type}$
$:\text{TYPE TYPE} \approx :\text{ARG TYPE}$

8. $:\sim down \quad \rightarrow$

$:\text{ARG TYPE}\sim \textit{i-type}$
$\wedge :\text{ARG TYPE TYPE} \approx :\text{TYPE}$

3 λ-Konversion

3.1 Das Content-Prinzip

Für Objekte der Sorte *content* nehmen wir die folgende Struktur an:

(7) *content* TERM *me*
AUX *me*

Die Werte der Attribute TERM und AUX eines *content*-Objekts sollen in jedem relevanten semantischen Modell äquivalent sein. Sie können sich aber durch ihre syntaktische Form unterscheiden. So kann der AUX-Wert einem IL-Term mit einem beta-Redex entsprechen und der TERM-Wert einem IL-Term, der beta-reduziert ist. Solche syntaktischen Umformungen können mit Hilfe von Kovariationsregeln emuliert werden. Als Beispiel wird weiter unten die Kovariationsregel für λ-Konversion angegeben. Das Content-Prinzip lautet wie folgt:

(8) $:\sim content \quad \rightarrow$

$$\begin{bmatrix}\text{TERM}\ \boxed{1}\\ \text{AUX}\ \boxed{1}\end{bmatrix} \vee \left(\boxed{2}\ \begin{bmatrix}\text{L-CR-IN}\ \boxed{3}\end{bmatrix} \wedge \lambda\texttt{-CR}(\boxed{3}, \boxed{2})\right)$$

Wir fordern, daß der CONTENT TERM-Wert eines *sign* immer vollständig beta-reduziert ist.
Ein (endliches, azyklisches) *me*-Objekt ist redex-frei genau dann, wenn es keine Komponente u hat, so daß u von der Sorte *appl* ist und der FUNC-Wert von u von der Sorte *abstr*.

(9) $:\sim sign \quad \rightarrow$

$$\neg\ \texttt{component}(x, y;\ x \approx :\text{CONTENT TERM} \wedge y\sim \textit{appl} \wedge y\text{FUNC}\sim \textit{abstr})$$

(10) *Die Relation* `component`

$\texttt{component}(x, y) :=$
δ

y tritt als Komponente von x auf.

3.2 λ-Konversion als Kovariationsregel

(11) *Die λ-Konversionskovariationsregel (λ-CR):*

$$\begin{bmatrix}\textit{content}\\ \text{TERM}\ \boxed{2}\\ \text{AUX}\ \boxed{1}\end{bmatrix} \mapsto \begin{bmatrix}\textit{content}\\ \text{TERM}\ \boxed{3}\\ \text{AUX}\ \boxed{1}\end{bmatrix} \wedge \texttt{find-redex}(x, y, v, w;\ \delta)$$

wobei

$\delta = x \approx \boxed{2}$
$\wedge\ v \approx \boxed{3}$
$\wedge\ \texttt{replace}(y\text{FUNC ARG},\ y\text{FUNC VAR},\ w,\ y\text{ARG})$
$\wedge\ \texttt{replace}(x, y, v, w)$

(12) *Die Relation* `find-redex`*:*

`find-redex`$(x, y, v, w) :=$
$x{\sim}abstr$
$\wedge\ x \simeq v$
$\wedge\ x\text{TYPE} \approx y\text{TYPE}$
$\wedge\ x\text{VAR} \approx v\text{VAR}$
$\wedge$ `find-redex`$(x\text{ARG},y,v\text{ARG},\text{W})$

`find-redex`$(x, y, v, w) :=$
$x{\sim}appl$
$\wedge\ x \simeq v$
$\wedge\ x\text{TYPE} \approx y\text{TYPE}$
$\wedge$ `find-redex`$(x\text{FUNC},y,v\text{FUNC},w)$
$\wedge\ x\text{ARG} \approx v\text{ARG}$

`find-redex`$(x, y, v, w) :=$
$x{\sim}appl$
$\wedge\ x \simeq v$
$\wedge\ x\text{TYPE} \approx y\text{TYPE}$
$\wedge\ x\text{FUNC} \approx v\text{FUNC}$
$\wedge$ `find-redex`$(x\text{ARG},y,v\text{ARG},w)$

`find-redex`$(x, y, v, w) :=$
$x{\sim}appl$
$\wedge\ x\text{FUNC}{\sim}abstr$
$\wedge\ x \approx y$
$\wedge\ v \approx w$

`find-redex`$(x, y, v, w) :=$
$x{\sim}equ$
$\wedge\ x \simeq v$
$\wedge\ x\text{TYPE} \approx y\text{TYPE}$
$\wedge$ `find-redex`$(x\text{ARG1},y,v\text{ARG1},w)$
$\wedge\ x\text{ARG2} \approx v\text{ARG2}$

`find-redex`$(x, y, v, w) :=$
$x{\sim}equ$
$\wedge\ x \simeq v$
$\wedge\ x\text{TYPE} \approx y\text{TYPE}$
$\wedge\ x\text{ARG1} \approx v\text{ARG1}$
$\wedge$ `find-redex`$(x\text{ARG2},y,v\text{ARG2},w)$

`find-redex`$(x, y, v, w) :=$
$x{\sim}$*l-const1*
$\wedge\ x \simeq v$
$\wedge\ x\text{TYPE} \approx y\text{TYPE}$
$\wedge$ `find-redex`$(x\text{ARG},y,v\text{ARG},w)$

`find-redex`$(x, y, v, w) :=$
$x{\sim}$*l-const2*
$\wedge\ x \simeq v$
$\wedge\ x\text{TYPE} \approx y\text{TYPE}$
$\wedge$ `find-redex`$(x\text{ARG1},y,v\text{ARG1},w)$
$\wedge\ x\text{ARG2} \approx v\text{ARG2}$

$\texttt{find-redex}(x, y, v, w) :=$

$x{\sim}\textit{l-const2}$
$\wedge\ x {\simeq}\ v$
$\wedge\ x\text{TYPE} \approx y\text{TYPE}$
$\wedge\ x\text{ARG1} \approx v\text{ARG1}$
$\wedge\ \texttt{find-redex}(x\text{ARG2},y,v\text{ARG2},w)$

$\texttt{find-redex}(x, y, v, w) :=$

$x{\sim}\textit{quant}$
$\wedge\ x {\simeq}\ v$
$\wedge\ x\text{TYPE} \approx y\text{TYPE}$
$\wedge\ x\text{VAR} \approx v\text{VAR}$
$\wedge\ \texttt{find-redex}(x\text{ARG},y,v\text{ARG},w)$

$\texttt{find-redex}(x, y, v, w) :=$

$x{\sim}\textit{up}$
$\wedge\ x {\simeq}\ v$
$\wedge\ x\text{TYPE} \approx y\text{TYPE}$
$\wedge\ \texttt{find-redex}(x\text{ARG},y,v\text{ARG},w)$

$\texttt{find-redex}(x, y, v, w) :=$

$x{\sim}\textit{down}$
$\wedge\ x {\simeq}\ v$
$\wedge\ x\text{TYPE} \approx y\text{TYPE}$
$\wedge\ \texttt{find-redex}(x\text{ARG},y,v\text{ARG},w)$

(13) *Die Relation* `replace`*:*

$\texttt{replace}(x, y, v, w) :=$

$x \approx y$
$\wedge\ v \approx w$

$\texttt{replace}(x, y, v, w) :=$

$\neg\ x \approx y$
$\wedge\ x{\sim}\textit{var}$
$\wedge\ x \approx v$

$\texttt{replace}(x, y, v, w) :=$

$\neg\ x \approx y$
$\wedge\ x{\sim}\textit{const}$
$\wedge\ x \approx v$

$\texttt{replace}(x, y, v, w) :=$

$\neg\ x \approx y$
$\wedge\ x{\sim}\textit{abstr}$
$\wedge\ x {\simeq}\ v$
$\wedge\ x\text{TYPE} \approx v\text{TYPE}$
$\wedge\ x\text{VAR} \approx v\text{VAR}$
$\wedge\ \texttt{replace}(x\text{ARG},y,v\text{ARG},w)$

$\texttt{replace}(x, y, v, w) :=$
$\neg\ x \approx y$
$\wedge\ x{\sim}appl$
$\wedge\ x{\simeq}\ v$
$\wedge\ x\text{TYPE} \approx v\text{TYPE}$
$\wedge\ \texttt{replace}(x\text{FUNC},y,v\text{FUNC},w)$
$\wedge\ \texttt{replace}(x\text{ARG},y,v\text{ARG},w)$

$\texttt{replace}(x, y, v, w) :=$
$\neg\ x \approx y$
$\wedge\ x{\sim}equ$
$\wedge\ x{\simeq}\ v$
$\wedge\ x\text{TYPE} \approx v\text{TYPE}$
$\wedge\ \texttt{replace}(x\text{ARG1},y,v\text{ARG1},w)$
$\wedge\ \texttt{replace}(x\text{ARG2},y,v\text{ARG2},w)$

$\texttt{replace}(x, y, v, w) :=$
$\neg\ x \approx y$
$\wedge\ x{\sim}\textit{l-const1}$
$\wedge\ x{\simeq}\ v$
$\wedge\ x\text{TYPE} \approx v\text{TYPE}$
$\wedge\ \texttt{replace}(x\text{ARG},y,v\text{ARG},w)$

$\texttt{replace}(x, y, v, w) :=$
$\neg\ x \approx y$
$\wedge\ x{\sim}\textit{l-const2}$
$\wedge\ x{\simeq}\ v$
$\wedge\ x\text{TYPE} \approx v\text{TYPE}$
$\wedge\ \texttt{replace}(x\text{ARG1},y,v\text{ARG1},w)$
$\wedge\ \texttt{replace}(x\text{ARG2},y,v\text{ARG2},w)$

$\texttt{replace}(x, y, v, w) :=$
$\neg\ x \approx y$
$\wedge\ x{\sim}quant$
$\wedge\ x{\simeq}\ v$
$\wedge\ x\text{TYPE} \approx v\text{TYPE}$
$\wedge\ x\text{VAR} \approx v\text{VAR}$
$\wedge\ \texttt{replace}(x\text{ARG},y,v\text{ARG},w)$

$\texttt{replace}(x, y, v, w) :=$
$\neg\ x \approx y$
$\wedge\ x{\sim}up$
$\wedge\ x{\simeq}\ v$
$\wedge\ x\text{TYPE} \approx v\text{TYPE}$
$\wedge\ \texttt{replace}(x\text{ARG},y,v\text{ARG},w)$

$\texttt{replace}(x, y, v, w) :=$
$\neg\ x \approx y$
$\wedge\ x{\sim}down$
$\wedge\ x{\simeq}\ v$
$\wedge\ x\text{TYPE} \approx v\text{TYPE}$
$\wedge\ \texttt{replace}(x\text{ARG},y,v\text{ARG},w)$

4 Sätze

4.1 Struktur von *sign*s

Wir gehen von einem Syntaxfragment des Deutschen aus, wie es in (Richter 1996b) definiert wird. Wir vereinfachen die Merkmalsarchitektur unter *sign*:

(14) *sign* PHON *list-of-phons*
HEAD *head*
CONTENT *content*

word

phrase HEAD-DTR *sign*
NONHEAD-DTR *sign*

4.2 Das Semantik-Prinzip

Für Wörter ergibt sich der CONTENT-Wert aus dem Lexikon. Bei *phrase*-Objekten wird der CONTENT-Wert über das Semantik-Prinzip festgelegt. Jeder Übersetzungsregel Montagues entspricht eine Klausel des Semantik-Prinzips:

(15) SEMANTIK-PRINZIP

$$:\sim phrase \quad \rightarrow \quad SP_1 \vee \ldots \vee SP_n$$

(16)z.B.: NP-VP-Regel: ($SP_{\text{NP-VP}}$)

a. IL-Notation: NP′(ˆVP′)

b. Deskription:

$:\sim$*hc-phrase*
$\wedge$ (weitere Bedingungen)
:CONTENT AUX$\sim$*appl*
$\wedge$:CONTENT AUX FUNC $\approx$:NONHEAD-DTR CONTENT TERM
$\wedge$:CONTENT AUX ARG$\sim$*up*
$\wedge$:CONTENT AUX ARG ARG $\approx$:HEAD-DTR CONTENT TERM

c. AVM:

$$\begin{bmatrix} \textit{phrase} & \\ \text{CONTENT AUX} & \begin{bmatrix} \textit{appl} & \\ \text{FUNC} & \boxed{1} \\ \text{ARG} & \begin{bmatrix} \textit{up} & \\ \text{ARG} & \boxed{2} \end{bmatrix} \end{bmatrix} \\ \text{NONHEAD-DTR} & \begin{bmatrix} \textit{sign} & \\ \text{CONTENT TERM} & \boxed{1} \end{bmatrix} \\ \text{HEAD-DTR} & \begin{bmatrix} \textit{sign} & \\ \text{CONTENT TERM} & \boxed{2} \end{bmatrix} \end{bmatrix}$$

5 Semantik-Prinzip für Quantoren

Wir nehmen die folgende Architektur an:

1. Ein Quantor kann entweder an der Stelle in die Semantik eingeführt werden, an der er in der Semantik einer Tochter auftritt, oder es kann statt dessen ein 'he$_i$' eingeführt werden.
2. Ein nicht direkt eingeführter Quantor kann seinen semantischen Beitrag an einer anderen Stelle leisten.
3. Jeder auftretende Quantor leistet genau einmal seinen vollständigen semantischen Beitrag.

5.1 Einführen von Quantoren

Wir wollen zulassen, daß eine quantifizierte NP nicht direkt im lokalen Baum ihren (vollen) Bedeutungsbeitrag leisten muß. Stattdessen kann auch ein Term der Form (he$_i$: $\lambda P.\check{}P(v_i)$) eingefügt werden.
Um die 'unterdrückten' Quantoren einzuführen, können sie an bestimmten Stellen vor den von der eigentlichen Regel festgelegten Term treten. z.B.:
Regel Funktionalapplikation: $\alpha'(\hat{}\beta')$
Quantoren: $\lambda P \exists x. \ldots \wedge \check{}P(x)(\hat{}\lambda x.\lambda R \exists y. \ldots \wedge \check{}R(y)(\hat{}\lambda y. \ldots (\alpha'(\hat{}\beta')) \ldots)$

(17) *Umformulierte NP-VP-Regel:*

:$\sim$*phrase*
$\wedge$ blabla $\quad\rightarrow$
$\qquad$ `insert-quants`$(x, y;\ \delta)$

mit $\delta =$
$x \sim$*appl*
xFUNC $\approx$:NONHEAD-DTR CONTENT TERM
xARG$\sim$*up*
xARG ARG $\approx$:HEAD-DTR CONTENT TERM
$y \approx$:

(18) *Die Relation* `insert-quants`

`insert-quants`(x, y) :=
$a \approx a$
$\wedge\ b \approx b$
$\wedge$ `dominated-quants`(y,a)
$\wedge$ `sublist`(a,b)
$\wedge$ `apply-quants`$(y$CONTENT AUX$,b,x)$

(19) *Die Relation* `apply-quants`

`apply-quants`(x, y, z) :=
$x \approx z$
$\wedge\ y \sim$*elist*

`apply-quants`(x, y, z) :=
$\wedge\ x \sim$*appl*
$\wedge\ x$FUNC $\approx y$FIRST
$\wedge\ x$ARG$\sim$*up*
$\wedge\ x$ARG ARG$\sim$*abstr*
$\wedge\ x$ARG ARG VAR $\approx y$FIRST ARG VAR
$\wedge$ `apply-quants`$(x$ARG ARG ARG$,y$REST$,\ z)$

(20) a. *Die Relation* `dominated-quants`

$\texttt{dominated-quants}(x, y) \underset{\delta}{:=}$

y ist eine Liste, die alle *me*-Objekte der Gestallt $\lambda P\ Quant\ v_i[\phi]$ enthält, die als CONTENT TERM-Wert einer Tochter von y auftreten.

b. *Die Relation* `sublist`

$\texttt{sublist}(x, y) \underset{\delta}{:=}$

x und y sind Listen, wobei y nur aus Elementen von x besteht. Kein Element aus x steht mehrmals in y

5.2 Einführen von $\lambda P.{}^{\smallsmile}P(v_i)$

(21) *Regel zur Einführung von* $\lambda P.{}^{\smallsmile}P(v_i)$*:* SP_{he_i}

$:\sim hc\text{-}phrase$
$\wedge$ Nichtkopftochter ist quantifizierte NP
$\wedge$ $\texttt{insert-quants}(x, y;\ \delta)$

mit $\delta =$
x hat die Form $\lambda P.{}^{\smallsmile}P(:\text{NONHEAD-DTR CONTENT TERM ARG VAR})$
$\wedge$ $x\text{ARG} \approx :\text{HEAD-DTR CONTENT TERM}$
$y \approx :$

(22) *Beispiel, falls keine zusätzlichen Quantoren eingeführt werden:*

$$\left[\begin{array}{ll}
\multicolumn{2}{l}{hc\text{-}phrase} \\
\text{CONTENT AUX} & \left[\begin{array}{l} appl \\ \text{FUNC } \lambda P.[{}^{\smallsmile}P](\boxed{2}) \\ \text{ARG } \boxed{1} \end{array}\right] \\
\text{HEAD-DTR} & \left[\text{CONTENT TERM } \boxed{1}\right] \\
\text{NONHEAD-DTR} & \left[\text{CONTENT TERM } \left[\begin{array}{ll} \multicolumn{2}{l}{abstr} \\ \text{ARG} & \left[\begin{array}{l} quant \\ \text{VAR } \boxed{2} \end{array}\right] \end{array}\right]\right]
\end{array}\right]$$

5.3 Quantorengrenzen

An bestimmten Stellen (z.B. bei unabhängigen Äußerungen, an Satzgrenzen etc.) wird verlangt, daß alle in die Struktur eingeführten Quantoren ihren semantischen Beitrag geliefert haben.

(23) *Prinzip der Quantorengrenzen*

$:\sim unembedded\text{-}sign$
$\vee \ \ldots \quad \rightarrow$
$\qquad \texttt{all-quants-included}(:)$

(24) a. *Die Relation* `all-quants-included`

`all-quants-included`(x) :=

  $a\approx a$

  $\wedge$ `dominated-quants`(x,a)

  $\wedge$ `quants-in`(x,a)

b. *Die Relation* `quants-in`

`quants-in`(x,y) :=

  $y\sim elist$

`quants-in`(x,y) :=

  `dominates`$(u,v;\ u\approx x\ \wedge$ `component`$(v\textsc{content aux},y\textsc{first}))$

  $\wedge$ `quants-in`$(x,y\textsc{rest})$

5.4 Einmaliges Auftreten

Wir müssen noch dafür sorgen, daß jeder Quantor nur einmal in die Struktur eingeführt wird.

(25) *Verbot von mehrmaligem Auftreten*

$:\sim sign \quad \rightarrow$

  $\neg$ `2-path`$(x,y;\ x\approx :\ \wedge$ `is-quant`$(y)\ )$

(26) a. *Die Relation* `2-path`

`2-path`(x,y) :=

  $a\approx a$

  $\wedge\ b\approx b$

  $\wedge$ `encode-path`(x,y,a)

  $\wedge$ `encode-path`(x,y,b)

  $\wedge\ \neg$ `same-path`(a,b)

b. *Die Relation* `encode-path`

`encode-path`(x,y,z) :=

  $x\approx y$

  $\wedge\ z\sim elist$

`encode-path`(x,y,z) :=

  $z\textsc{first}\approx x$

  $\wedge$ [(`is-1`$(z\textsc{first})$ $\wedge$ `encode-path`$(x\textsc{arg},y,z\textsc{rest rest}))$

  $\vee$ (`is-2`$(z\textsc{rest first})$ $\wedge$ `encode-path`$(x\textsc{arg1},y,z\textsc{rest rest}))$

  $\vee$ (`is-3`$(z\textsc{rest first})$ $\wedge$ `encode-path`$(x\textsc{arg2},y,z\textsc{rest rest}))$

  $\vee$ (`is-4`$(z\textsc{rest first})$ $\wedge$ `encode-path`$(x\textsc{func},y,z\textsc{rest rest}))$

  $\vee$ (`is-5`$(z\textsc{rest first})$ $\wedge$ `encode-path`$(x\textsc{var},y,z\textsc{rest rest}))$

  $\vee$ (`is-6`$(z\textsc{rest first})$ $\wedge$ `encode-path`$(x\textsc{arg},y,z\textsc{rest rest}))$

  $\vee$ ...]

c. *Die Relationen* `is-`i

Wir nehmen an, daß die linguistische Theorie nur endlich viele Attributnamen $\alpha_1,\ldots,\alpha_n$ verwendet. Für jedes i mit $1\leq i\leq n$ definieren wir eine Relation `is-`i:

Für $i=1$:

`is-`$i(x)$:=

  $x\sim list\text{-}of\text{-}plus$

  $\wedge\ x\textsc{rest}\sim elist$

Für jedes i mit $1 < i \leq n$:

$\texttt{is-}i(x) :=$
$\quad x{\sim}\textit{list-of-plus}$
$\quad \wedge\ \texttt{is-}(i-1)(x\text{REST})$

d. *Die Relation* `same-path`

$\texttt{same-path}(x, y) :=$
$\quad x{\sim}\textit{elist}$
$\quad \wedge\ y{\sim}\textit{elist}$

$\texttt{same-path}(x, y) :=$
$\quad x\text{FIRST} \approx y\text{FIRST}$
$\quad \wedge\ \texttt{same-length}(x\text{REST FIRST},\ y\text{REST FIRST})$
$\quad \wedge\ \texttt{same-path}(x\text{REST REST},\ y\text{REST REST})$

e. *Die Relation* `same-length`

$\texttt{same-length}(x, y) :=$
$\quad x{\sim}\textit{elist}$
$\quad \wedge\ y{\sim}\textit{elist}$

$\texttt{same-length}(x, y) :=$
$\quad \texttt{same-length}(x\text{REST},\ y\text{REST})$

6 Quasi-syntaktische Beschränkungen auf quasi-semantische Repräsentationen

Beispiele

- *Lokalitätsbedingungen für Quantorenskopus*

  In einem Satz wie *daß Maria nicht glaubt, daß manche Fische fliegen* kann *manche Fische* nicht Skopus über die Matrixnegation haben.

  Lösungsskizze LF: Die Negation ist im Deutschen eine Barriere für Quantorenanhebung.

  Lösungsskizze TFL: Die Negation legt fest, daß das von ihr modifizierte Verb eine Quantorengrenze ist.

- *Bindungsdaten*

  Gegensatz zwischen Pronomen, die als 'gebundene Variablen' übersetzt werden und solchen, die koreferentiell interpretiert werden.

(27) a. Karl$_i$ behauptet, er$_i$ hat ein Alibi.
b. [Jeder Verdächtige]$_i$ behauptet, er$_i$ hat ein Alibi.
c. Karl$_i$ hat ein Alibi, behauptet er$_i$.
d. * [Jeder Verdächtige]$_i$ hat ein Alibi, behauptet er$_i$.

Lösungsskizze LF: Verb-Zweit-Sätze sind Barrieren für Quantorenanhebung.

Lösungsskizze TFL: Pronomen, die als 'gebundene Variable' übersetzt werden, müssen im Skopus des entsprechenden Quantors enden. Dazu muß der jedoch 'spät' genug in die semantische Struktur eingeführt werden können. Verb-Zweit-Sätze sind aber Grenzen für Quantoren.

Fazit
Die syntaktischen Beschränkungen können in LF- und TFL-Ansätzen gleichermaßen formuliert werden.
PTQ kann in TFL nachgespielt werden mit einer linguistisch-theoretisch motivierten 'Syntax'.
IL-Terme sind in TFL Teil der linguistischen Objekte
Es gibt keine für uns erkennbare Abbildung zwischen einer Syntax und einer Semantik innerhalb des Zeichens, vielmehr: jedes Zeichen hat einen Content, der ein IL-Term ist und interpretiert wird.

7 Beziehung zwischen Syntax und Semantik

In LF-Ansätzen ist nicht klar, welche syntaktischen Teile an welcher Stelle wie zu interpretieren sind. Eine Übersetzung in eine formale Semantiksprache ist, im Idealfall, nicht nötig, wird aber oft geliefert, um dem Leser die genauen Interpretationsregeln für LF-Strukturen zu ersparen.
Im vorgestellten TFL-Ansatz wird der CONTENT TERM-Wert direkt modelltheoretisch interpretiert. Seine Gestalt ist die eines IL-Terms. Es gibt damit keine rein syntaktische Repräsentationsebene, die direkt einer semantischen korrespondiert, sondern unter CONTENT einen IL-Term, der seiner Natur nach frei ist von Information, die nur in der Syntax benötigt wird.

References

Ait-Kaci, H.: 1984, *A Lattice Theoretic Approach to Computation Based on a Calculus of Partially Ordered Type Structures*, PhD thesis, University of Pennsylvania.

Carpenter, B.: 1992, *The Logic of Typed Feature Structures*, Cambridge University Press. Cambridge, Massachusetts, USA.

King, P. J.: 1989, *A logical Formalism for Head-Driven Phrase Structure Grammar*, PhD thesis, University of Manchester.

King, P. J.: 1994, An expanded logical formalism for Head-Driven Phrase Structure Grammar, *Arbeitspapiere des SFB 340 59*, Universität Tübingen.

Meurers, D.: 1994, On implementing an HPSG theory, *in* E. Hinrichs, D. Meurers and T. Nakazawa (eds), *Partial-VP and Split-NP Topicalization in German — A HPSG Analysis and its Implementation*, Arbeitspapiere des SFB 340, Nr. 58, Universität Tübingen, pp. 47–155.

Montague, R.: 1974, The proper treatment of quantification in ordinary English, *in* R. H. Thomason (ed.), *Formal Philosophy. Selected Papers of Richard Montague*, Yale University Press, pp. 247 – 270.

Richter, F.: 1996a, Artikel 1, 1997, Manuskript Universität Tübingen, erscheint in *Arbeitspapiere des SFB-340.*

Richter, F.: 1996b, Unorthodoxe Ruminationen, Manuskript UniversitätTübingen.

Richter, F. and Sailer, M.: 1995, *Remarks on Linearization. Reflections on the Treatment of LP-Rules in HPSG in a Typed Feature Logic*, Master's thesis, Universität Tübingen.