A UNIFICATION–BASED ID/LP PARSING SCHEMA

Frank Morawietz
SFB 340, A8
Universität Tübingen
Wilhelmstr. 113
72074 Tübingen
Germany
frank@sfs.nphil.uni-tuebingen.de

September 21, 1995

- GPSG (Gazdar et al. 1985) and HPSG (Pollard and Sag 1994) want ID/LP.

- HPSG does not provide succinct definitions.

- We do not incorporate LP rules into the theory, but we show a way of using ID and LP rules in a parser for Unification ID/LP grammars.

- It follows a very short motivation of ID/LP.

Context–Free vs. ID/LP Grammars

$$
\begin{aligned}
S &\rightarrow A\ B\ C \\
S &\rightarrow A\ C\ B \\
S &\rightarrow B\ A\ C \\
A &\rightarrow a \\
B &\rightarrow b \\
C &\rightarrow c
\end{aligned}
$$

Context–Free version

$$
\begin{aligned}
S &\rightarrow A,\ B,\ C \\
A &\rightarrow a \\
B &\rightarrow b \\
C &\rightarrow c
\end{aligned}
\qquad A \prec C
$$

ID/LP version

ID/LP allows:

- separation of dominance relations, as for example constituency, and the correlated precedence phenomena, i.e. syntactic properties of the categories

- difference between fixed vs. free word order

- generalizations

- The ID/LP version expresses the same language with fewer rules and the generalization that Cs must not precede As

- psychological reasons (free word order is easier)

Indirect ID/LP parsing

Direct ID/LP Parsing

- Shieber's parser (Shieber 1984)
  - Earley–based
  - Productions split in ID and LP rules, RHSs of rules are now treated as multisets
  - Completer and Predictor test for LP acceptability

- Seiffert's parser (Seiffert 1991)
  - Feature structures as nonterminals
  - Application of LP rules
  - LP acceptability
  - Nonlocal flow of information

  $\leadsto$ second step for the algorithm

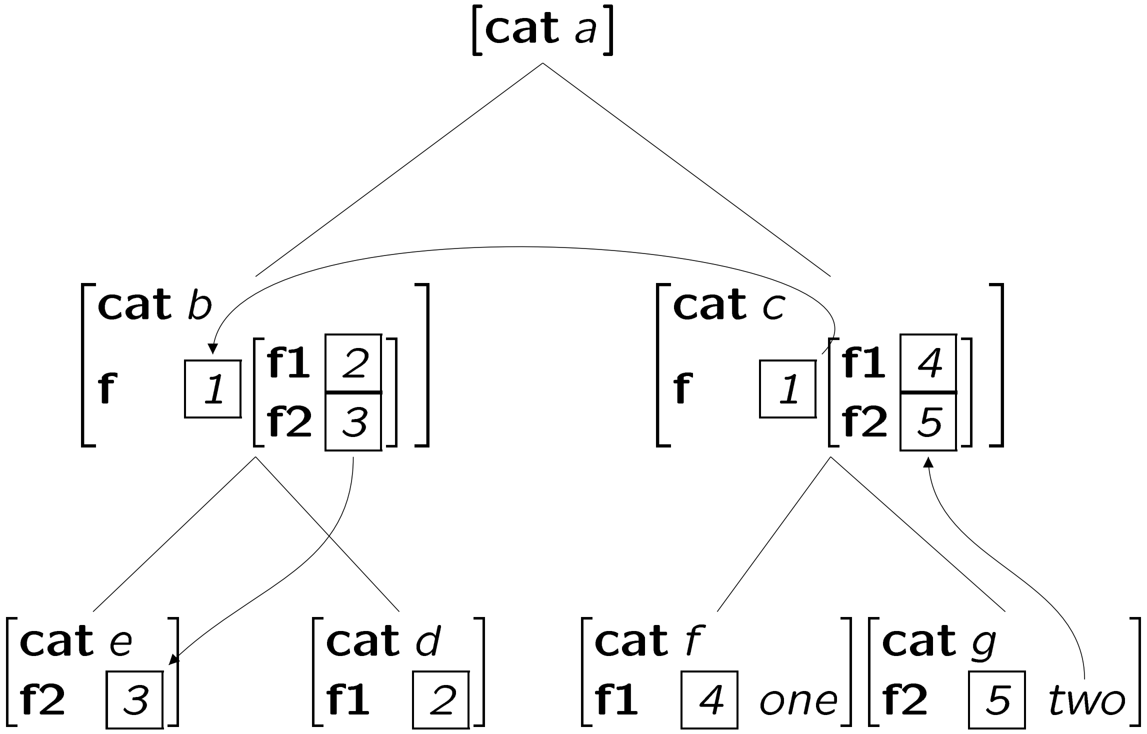
<u>Seiffert's Grammar</u>*

$$
\begin{array}{lll}
\text{Lexicon} & : \begin{bmatrix}\textbf{cat}\ \textit{d}\end{bmatrix} & \rightarrow h \\
 & \begin{bmatrix}\textbf{cat}\ \textit{e}\end{bmatrix} & \rightarrow i \\
 & \begin{bmatrix}\textbf{cat} & \textit{f} \\ \textbf{f1} & \textit{one}\end{bmatrix} & \rightarrow j \\
 & \begin{bmatrix}\textbf{cat} & \textit{g} \\ \textbf{f2} & \textit{two}\end{bmatrix} & \rightarrow k \\
\text{ID–Rules} & : \begin{bmatrix}\textbf{cat}\ \textit{a}\end{bmatrix} & \rightarrow \begin{bmatrix}\textbf{cat} & \textit{b} \\ \textbf{f} & \boxed{1}\end{bmatrix} , \begin{bmatrix}\textbf{cat} & \textit{c} \\ \textbf{f} & \boxed{1}\end{bmatrix} \\
 & \begin{bmatrix}\textbf{cat} & \textit{b} \\ \textbf{f} & \begin{bmatrix}\textbf{f1} & \boxed{2} \\ \textbf{f2} & \boxed{3}\end{bmatrix}\end{bmatrix} & \rightarrow \begin{bmatrix}\textbf{cat} & \textit{d} \\ \textbf{f1} & \boxed{2}\end{bmatrix} , \begin{bmatrix}\textbf{cat} & \textit{e} \\ \textbf{f2} & \boxed{3}\end{bmatrix} \\
 & \begin{bmatrix}\textbf{cat} & \textit{c} \\ \textbf{f} & \begin{bmatrix}\textbf{f1} & \boxed{4} \\ \textbf{f2} & \boxed{5}\end{bmatrix}\end{bmatrix} & \rightarrow \begin{bmatrix}\textbf{cat} & \textit{f} \\ \textbf{f1} & \boxed{4}\end{bmatrix} , \begin{bmatrix}\textbf{cat} & \textit{g} \\ \textbf{f2} & \boxed{5}\end{bmatrix} \\
\text{LP–Rules} & : \begin{bmatrix}\textbf{f1}\ \textit{one}\end{bmatrix} & \prec \begin{bmatrix}\textbf{f2}\ \textit{two}\end{bmatrix} \\
 & \begin{bmatrix}\textbf{cat}\ \textit{b}\end{bmatrix} & \prec \begin{bmatrix}\textbf{cat}\ \textit{c}\end{bmatrix} \\
\text{Start Symbol} & : \begin{bmatrix}\textbf{cat}\ \textit{a}\end{bmatrix} &
\end{array}
$$

$$\mathrm{L}(G) = \{hijk\}$$

*Taken from Seiffert (1991).

Nonlocal Feature Passing



On input *ihjk* – which is not well formed – Seiffert's parser can not determine on the first pass whether the local tree (cat b, (cat e, cat d)) is well formed since the information that is relevant for the LP rule, namely that *one* has to precede *two*, is not available yet. Only after the local trees (cat c, (cat f, cat g)) and (cat a, (cat b, cat c)) have been constructed, the information becomes accessible through structure sharing. The nonlocal flow can be seen in the parse tree given in figure 2.* The information *one* and *two* comes from the lexical entries for *f* and *g* and is passed to *e* and *d* via *c* and *b*. Only then is the information that the value for f2 at *e* is *two* and that the value for f1 at *d* is *one* available and the local tree (cat b, (cat e, cat d)) has to be ruled out since the LP rule $\left[\textbf{f1}\ \textit{one}\right] \prec \left[\textbf{f2}\ \textit{two}\right]$ is violated.

This was purely technical, but there is a linguistic motivation as well.

*For readability reasons the arrows are drawn just for one value, namely *two*.

- Formalization similar to the one in Sikkel (1993)
  - minimal approach
  - $Feat \cap Const = \emptyset$
  - DAGs as feature structures $\rightsquigarrow$ feature lattice
  - composite feature structures
  - assumes context–free backbone, but nothing hinges on that.

1 Definition (Unification ID/LP Grammar)

A Unification ID/LP grammar is defined as a tuple $\mathcal{G} = (G, \Phi, \varphi_0, \varsigma_0, Lex)$ *such that:*

$G = \langle N, \Sigma, ID, LP, S \rangle$	*is a (context–free) ID/LP grammar.*
$\Phi = \phi(Feat, Const)$	*is a lattice of feature structures.*
$\varphi_0 : ID \to \Phi$	*is a function that assigns a composite feature structure to each rule in ID.*
$\varsigma_0 : LP \to \Phi$	*is a function that assigns a composite feature structure to each rule in LP.*
$Lex : \Sigma \to \wp(\Phi)$	*is a function that assigns a set of feature structures to each lexical entry.*

Use your favorite feature structure definition, for example:

- automata
- graphs
- constraint based
- typed or non typed
- with or without appropriateness specifications
- ...

Composite feature structures are multiply rooted feature structures, i.e. there is a set of root nodes corresponding to the number of the formal parameters* Note that roots may have incoming edges.

*In $\Gamma(X_1, \ldots, X_k)$ $X_1, \ldots, X_k$ are the formal parameters.

Notation

- $V \simeq N \cup \Sigma$

- $\varphi_0(X_i) \simeq \varphi_0(X_0 \rightarrow X_1, \ldots, X_k)|_{X_i}$

- $\varsigma_0(A) \simeq \varsigma_0(A \prec B)|_A$

- $a, b, \ldots$ are elements of Σ

- $\varsigma(LP)$ is the set of all the $\varsigma_0(X_i)$

- $\mathcal{G}_{\mathcal{ID}/\mathcal{LP}}$ is the class of unification ID/LP grammars

Assumptions

- $Feat \cap Const = \emptyset$

- $\textbf{cat} \in Feat$

- $V \subseteq Const$

- ID and LP are multisets of rules

- $\langle \varsigma(LP), \prec \rangle$ is a transitive, asymmetric, irreflexive relation.

- $\forall_{0 \leq i \leq k}\ \varphi_0(X_i).\textbf{cat} = X_i$ holds for each ID rule $X_0 \rightarrow\ X_1, \ldots, X_k$ and likewise for the LP rules.

- $\varphi(a).\textbf{cat} \in \Sigma$

- Transitivity of the LP rules

- We just assume for expository purposes that we are dealing with the preterminals only.

We need the concepts to deal with ID/LP, i.e. application of an LP rule and LP acceptability.

2 Definition (applies)

An LP rule $A \prec B$ *applies to a sequence* $\sigma = \sigma_1 \ldots \sigma_n$*,* $\sigma_k \in V$ *iff there exist* i, j*,* $i \neq j$*,* $1 \leq i, j \leq n$ *such that*

$\varsigma_0(A) \sqsubseteq \varphi(\sigma_i)$ *and*
$\varsigma_0(B) \sqsubseteq \varphi(\sigma_j)$*.*

3 Definition (LP acceptable)

A sequence σ *is LP acceptable iff no LP rule applies to it with* $j < i$*.*

- application: are the categories informative enough?

- acceptability: If no LP rule applies to a structure it has to be LP acceptable. The same is true if no rule applies to it causing a violation of the order.

- Overloading of LP acceptability: we use it on sets of sequences of feature structures

4 Definition (parse tree)

A grammar $\mathcal{G} = (G, \Phi, \varphi_0, \varsigma_0, Lex)$ *generates a parse tree* $\langle \tau, \varphi(\tau) \rangle$ *for the sentence* $a_1 \ldots a_n$ *iff*

- *the root is labeled with the initial symbol* S *of* G *and* $\varphi_0(S) \sqsubseteq \varphi(S)$*;*

- $\forall_{1 \leq i \leq n}\ a_i \in \Sigma$ *and* $\varphi'(a_i) \in Lex(a_i)$ *such that* $\varphi'(a_i) \sqsubseteq \varphi(a_i)$*;*

- *for all subtrees* $\tau_i \in \tau$*, where* T_0 *immediately dominates* $T_1 \ldots T_n$*, there is a rule* $T_0 \rightarrow \alpha$ *in* ID *such that*

  $T_1 \ldots T_n \in permute(\alpha)$ *and*
  $\forall_{1 \leq i \leq n}\ \varphi_0(T_i) \sqsubseteq \varphi(T_i)$ *and*
  the yield of the local tree τ_i *is LP acceptable.*

Assume tree formalization from Partee et al. (1990). The given formalization does not enforce adequate decoration. We have to assume that parse trees are minimal in this sense (the formalization can be found in Sikkel (1993)).

Note that structure sharing is indicated with a plain $=$, whereas equality is denoted with $\doteq$.

5 Definition ($permute$)

Let $\mathcal{G}$ be a unification ID/LP grammar, $A \in V$, then the function $permute : \wp_m(V) \rightarrow \wp_m(V^*)$ is defined as

$$permute(\emptyset) = \emptyset$$
$$permute(\alpha) = \left\{ A\beta \;\middle|\; \begin{array}{l} A \in \alpha \text{ and} \\ \beta \in permute(\alpha \backslash \{A\}) \end{array} \right\}$$

6 Definition (Language of the Grammar)

The language generated by $\mathcal{G}$ is defined as

$$L(\mathcal{G}) = \left\{ w = w_1 \ldots w_n \;\middle|\; \begin{array}{l} \textit{there exists a parse} \\ \textit{tree } \langle \tau, \varphi(\tau) \rangle \textit{ with} \\ \textit{yield } w_1 \ldots w_n \textit{ \&} \\ \varphi(\tau) \not\doteq \bot \end{array} \right\}$$

7 The Parsing Problem

For a given sentence $a_1 \ldots a_n \in \Sigma^*$ and a grammar $\mathcal{G} \in \mathcal{G}_{\mathcal{ID}/\mathcal{LP}}$, find all the parse trees with result $\varphi(S)$.

This completes the formalization of Unification ID/LP grammars. For the parsing schema we need some additional definitions:

- a sequence of feature structures may not be specific enough to determine applicability of LP rules $\rightsquigarrow$ *weak application* is a test unification

- *possibly LP violated*: Complication because subsumption includes unification, so the structure has to be LP acceptable, but . . .

- simple recursive definition of *store union* for the percolation of the store since the logic does not support set–valued feature structures

  – store contains the pending structures and restricts further use of the item

  – We need the store in the definition of the feature structure which is assigned to a rule for structure sharing.

8 Definition (weakly applies)

An LP rule $A \prec B$ *weakly applies to a sequence* $\sigma = \sigma_1 \ldots \sigma_n$*,* $\sigma_k \in V$ *iff there exist* i, j*,* $i \neq j$*,* $1 \leq i, j \leq n$ *such that*

$\varsigma_0(A) \sqcup \varphi(\sigma_i)$ *and*

$\varsigma_0(B) \sqcup \varphi(\sigma_j)$*.*

9 Definition (possibly LP violated)

A sequence σ *is possibly LP violated iff it is LP acceptable, but an LP rule weakly applies to it with* $j < i$*.*

Why not use unification for application in the first case?

The rule might not be instantiated that way – so why rule it out. Or one would have to store it as kind of pending information the way we do it here. But clearly this would be inefficient.

10 Definition (Store Union ($\bowtie$))

The union of an LP store Θ *with another LP store* Ω *with resulting LP store* Δ $(\varphi(\Theta) \bowtie \varphi(\Omega) \doteq \varphi(\Delta))$ *is recursively defined as*

$$\begin{aligned}&\varphi(\Delta) \doteq \varphi(\Omega) \ \textit{if}\\ &\qquad \varphi(\Theta).\mathbf{hd} \doteq e\_list\end{aligned}$$

$$\begin{aligned}&\varphi(\Delta) \doteq \varphi(\Theta).\mathbf{tl} \bowtie \varphi(\Omega) \ \textit{if}\\ &\qquad \varphi(\Theta).\mathbf{hd} \doteq \varphi(A) \wedge \varphi(A) \in \varphi(\Omega)\end{aligned}$$

$$\begin{aligned}&\varphi(\Delta).\mathbf{hd} \doteq \varphi(A) \wedge \varphi(\Delta).\mathbf{tl} \doteq \varphi(\Theta).\mathbf{tl} \bowtie \varphi(\Omega) \ \textit{if}\\ &\qquad \varphi(\Theta).\mathbf{hd} \doteq \varphi(A) \wedge \varphi(A) \notin \varphi(\Omega)\end{aligned}$$

Why parsing schemata?

- designed to compare parsing algorithms
- efficient representation because no control information and no data structures are given

$\rightsquigarrow$ allowance of nontrivial noncluttered algorithms

Presentation in three steps:

- Parsing System $\rightsquigarrow$ Deduction System with fixed input string and grammar
- uninstantiated Parsing System $\rightsquigarrow$ abstracts from the fixed input string
- Parsing Schema $\rightsquigarrow$ abstracts further from the grammar

11 Parsing System

A parsing system for some grammar G and input string $a_1, \ldots, a_n$ is a triple $\mathcal{P}(\mathcal{I}, \mathcal{H}, \mathcal{D})$ with

- $\mathcal{I}$ *a set of items;*
- $\mathcal{H}$ *a finite set of items (not necessarily a subset of $\mathcal{I}$), the hypotheses; and*
- $\mathcal{D}$ *a set of deduction steps $\eta_1, \ldots, \eta_m \vdash \xi$ with*
  $\eta_i \in \mathcal{I} \cup \mathcal{H}$ *for* $0 \leq i \leq m$ *and*
  $\xi \in \mathcal{I}$.

Items of an Earley or chart parser:

$[A \rightarrow \alpha \bullet \beta, i, j]$

The set of valid items which contains those and only those items from $\mathcal{I}$ which can be inferred from $\mathcal{H}$ by steps from $\mathcal{D}$.

12 Uninstantiated Parsing System

An uninstantiated parsing system for a grammar G is a triple $\langle \mathcal{I}, \mathcal{H}, \mathcal{D} \rangle$ with $\mathcal{H}$ a function that assigns a set of hypotheses to each string $a_1, \ldots, a_n \in \Sigma^*$ such that $P(\mathcal{I}, \mathcal{H}(a_1, \ldots, a_n), \mathcal{D})$ is a parsing system.

Since $\mathcal{H}$ can be defined for our purposes as follows, we are not going to make a difference between the two kinds of parsing systems.

$$\mathcal{H}(a_1, \ldots, a_n) = \left\{ [a, i-1, i] \;\middle|\; a = a_i, 1 \leq i \leq n \right\}.$$

13 Parsing Schema

A parsing schema for a class of grammars $\mathcal{G}$ is a function that assigns an (uninstantiated) parsing system to every grammar in $\mathcal{G}$.

The idea:

- change the deduction steps for the completer
- add sequence to the store in case of *possible LP violation* by structure sharing
- in completion we check the store for LP acceptability
- only completion is forbidden since the edge might be used in another way which does not lead to an LP violation
- percolation of the LP stores
- optimizations for the implementation in **Troll** (Gerdemann and Götz forthcoming)

14 ID/LP Parsing System

A parsing system $\mathcal{P}(\mathcal{I}_{ID/LP}, \mathcal{H}_{ID/LP}, \mathcal{D}_{ID/LP})$ *for a Unification ID/LP grammar* $\mathcal{G} = (G, \Phi, \varphi_0, \varsigma_0, Lex)$ *is defined by*

$$\mathcal{I}_{ID/LP} \;=$$
$$\left\{ [A \to \alpha \bullet \beta, i, j, \Delta]_\xi \;\middle|\; \begin{array}{l} A \to \alpha\beta \in ID \\ 0 \leq i \leq j \\ \varphi_0(A \to \alpha\beta) \sqsubseteq \varphi(\xi) \\ \varphi(\xi) \not\doteq \bot \\ \alpha \text{ and } \Delta \text{ are LP acceptable} \end{array} \right\}$$

$$\mathcal{H}_{ID/LP} \;=$$
$$\{\, [a, j-1, j, \{\}] \mid \varphi(a) \in Lex(a_j) \,\}$$

Notation

- Items have been augmented by the LP store Δ,
  $$[A \to \alpha \bullet \beta, i, j] \leadsto [A \to \alpha \bullet \beta, i, j, \Delta]$$

- $[A \to \alpha \bullet \beta, i, j, \Delta]_\xi$ is referred to as ξ

- $\varphi(\xi)$ refers to the composite feature structure of the item and Δ_η to the part of the feature structure η covering just Δ

- Note that items always have to be LP acceptable, i.e. both the LP store and the recognized sequence of feature structures

- $\mathcal{H}$ includes an empty LP store now

$$\mathcal{D}_{ID/LP} = \mathcal{D}^{Init} \cup \mathcal{D}^{Scan} \cup \mathcal{D}^{Pred} \cup \mathcal{D}^{Compl1} \cup \mathcal{D}^{Compl2}$$

$$\mathcal{D}^{Init} = \left\{ \frac{}{[S \to \bullet\gamma, 0, 0, \{\}]_\xi} \,\middle|\, \varphi(\xi) \doteq \varphi_0(S \to \gamma) \right\}$$

$$\mathcal{D}^{Scan} = \left\{ \frac{\begin{array}{c}[A \to \alpha \bullet \beta a \delta, i, j, \Delta]_\eta, \\ [a, j, j+1, \{\}]_\zeta\end{array}}{[A \to \alpha a \bullet \beta\delta, i, j+1, \Delta]_\xi} \,\middle|\, \varphi(\xi) \doteq \varphi(\eta) \sqcup \varphi(\zeta) \right\}$$

$$\mathcal{D}^{Pred} = \left\{ \frac{[A \to \alpha \bullet \beta B \delta, i, j, \Delta]_\eta}{[B \to \bullet\gamma, j, j, \{\}]_\xi} \,\middle|\, \varphi(\xi) \doteq \varphi(B_\eta) \angle \Psi_0(B) \sqcup \varphi_0(B \to \gamma) \right\}$$

All the complicated ID/LP processes have been taken out of the predictor because

- it would be much more complicated

- it would not be efficient, i.e. a lot of edges would be *possibly LP violated*

- tests show no gain

15 Definition (default restrictor)

Let $\mathcal{G} = (G, \Phi, \varphi_0, \varsigma_0, Lex) \in \mathcal{G}_{\mathcal{ID}/\mathcal{LP}}$. *A default restrictor* $\Psi_0(B)$ *for each* $B \in N$ *is defined as the unique restrictor that satisfies the following conditions:*

1. *for any production* $A \to \alpha B \beta \in ID$ *it holds that*
$$\varphi_0(A \to \alpha B \beta)|_B = \varphi_0(A \to \alpha B \beta)|_B \angle \Psi_0(B)$$

2. *for any other* $\Psi(B)$ *it holds that* $\Psi_0(B) \sqsubseteq \Psi(B)$.

Hence the default restrictor can be seen as a function $\Psi_0 : N \times \varphi_0 \to \Phi$.

$$\mathcal{D}^{Compl1} =$$

$$\left\{ \frac{\begin{array}{c}[A \to \alpha \bullet \beta B\delta, i, j, \Delta]_\eta,\\ [B \to \gamma\bullet, j, k, \Omega]_\zeta\end{array}}{[A \to \alpha B \bullet \beta\delta, i, k, \Theta]_\xi} \;\middle|\; \begin{array}{l} \varphi(\xi) \doteq \varphi(\eta) \sqcup \varphi(\zeta) \\ \varphi(\Theta_\xi) \doteq \varphi(\Delta_\eta) \bowtie \varphi(\Omega_\zeta) \\ \varphi((\alpha B)_\xi) \text{ is not possibly LP} \\ \qquad\qquad\quad \text{violated} \end{array} \right\}$$

$$\mathcal{D}^{Compl2} =$$

$$\left\{ \frac{\begin{array}{c}[A \to \alpha \bullet \beta B\delta, i, j, \Delta]_\eta,\\ [B \to \gamma\bullet, j, k, \Omega]_\zeta\end{array}}{[A \to \alpha B \bullet \beta\delta, i, k, \Theta]_\xi} \;\middle|\; \begin{array}{l} \varphi(\xi) \doteq \varphi(\eta) \sqcup \varphi(\zeta) \\ \varphi((\alpha B)_\xi) \text{ is possibly LP violated} \\ \varphi(\Theta_\xi) \doteq \varphi((\alpha B)_\xi) \bowtie \varphi(\Delta_\eta) \bowtie \varphi(\Omega_\zeta) \\ \varphi((\alpha B)_\Theta) = \varphi((\alpha B)_\xi) \end{array} \right\}$$

- β and δ in the deduction steps can be empty or of any other cardinality $\rightsquigarrow$ RHS is a multiset

- $\mathcal{D}^{Compl2}$: the new LP store is the result of the union of the old stores plus the sequence of categories found

- the schema is the same as before because the parsing system was given for an arbitrary grammar from $\mathcal{G}_{\mathcal{ID}/\mathcal{LP}}$

crucial steps in the trace of a parser solving the problem presented previously, i.e. nonlocal feature passing

Be careful, different notation

- no backbone, just feature structures

- $[LHS \to [REC] \bullet \{REST\}, Start, End, \{LPStore\}]$

An Example Trace

1. $$[\ \begin{bmatrix} \textbf{cat}\ b & \\ \textbf{f} & \begin{bmatrix} \textbf{f1} & \boxed{\mathit{2}} \\ \textbf{f2} & \boxed{\mathit{3}} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \boxed{\mathrm{X}} \begin{bmatrix} \textbf{cat}\ e & \\ \textbf{f2} & \boxed{\mathit{3}} \end{bmatrix} \begin{bmatrix} \textbf{cat}\ d & \\ \textbf{f1} & \boxed{\mathit{2}} \end{bmatrix} \end{bmatrix} \bullet \{\}\ ,\ 0\ ,\ 2\ ,\ \left\{ \boxed{\mathrm{X}} \begin{bmatrix} \textbf{cat}\ e & \\ \textbf{f2} & \boxed{\mathit{3}} \end{bmatrix} \begin{bmatrix} \textbf{cat}\ d & \\ \textbf{f1} & \boxed{\mathit{2}} \end{bmatrix} \right\}\ ]$$

2. $$[\ \begin{bmatrix} \textbf{cat}\ a \end{bmatrix} \rightarrow \begin{bmatrix} \begin{bmatrix} \textbf{cat}\ b & \\ \textbf{f} & \boxed{\mathit{1}} \end{bmatrix} \end{bmatrix} \bullet \left\{ \begin{bmatrix} \textbf{cat}\ c & \\ \textbf{f} & \boxed{\mathit{1}} \begin{bmatrix} \textbf{f1} & \boxed{\mathit{2}} \\ \textbf{f2} & \boxed{\mathit{3}} \end{bmatrix} \end{bmatrix} \right\}\ ,\ 0\ ,\ 2\ ,\ \left\{ \begin{bmatrix} \textbf{cat}\ e & \\ \textbf{f2} & \boxed{\mathit{3}} \end{bmatrix} \begin{bmatrix} \textbf{cat}\ d & \\ \textbf{f1} & \boxed{\mathit{2}} \end{bmatrix} \right\}\ ]$$

3. $$[\ \begin{bmatrix} \textbf{cat}\ c & \\ \textbf{f} & \begin{bmatrix} \textbf{f1} & \boxed{\mathit{6}} \\ \textbf{f2} & \boxed{\mathit{7}} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \begin{bmatrix} \textbf{cat}\ f & & \\ \textbf{f1} & \boxed{\mathit{6}} & \mathit{one} \end{bmatrix} \begin{bmatrix} \textbf{cat}\ g & & \\ \textbf{f2} & \boxed{\mathit{7}} & \mathit{two} \end{bmatrix} \end{bmatrix} \bullet \{\}\ ,\ 2\ ,\ 4\ ,\ \{\}\ ]$$

*4. $$[\ \begin{bmatrix} \textbf{cat}\ a \end{bmatrix} \rightarrow \begin{bmatrix} \begin{bmatrix} \textbf{cat}\ b & \\ \textbf{f} & \boxed{\mathit{1}} \end{bmatrix} \begin{bmatrix} \textbf{cat}\ c & \\ \textbf{f} & \boxed{\mathit{1}} \begin{bmatrix} \textbf{f1} & \boxed{\mathit{2}} & \mathit{one} \\ \textbf{f2} & \boxed{\mathit{3}} & \mathit{two} \end{bmatrix} \end{bmatrix} \end{bmatrix} \bullet \{\}\ ,\ 0\ ,\ 4\ ,\ \left\{ \begin{bmatrix} \textbf{cat}\ e & & \\ \textbf{f2} & \boxed{\mathit{3}} & \mathit{two} \end{bmatrix} \begin{bmatrix} \textbf{cat}\ d & & \\ \textbf{f1} & \boxed{\mathit{2}} & \mathit{one} \end{bmatrix} \right\}\ ]$$

- input *ihjk*

- first edge: created by the parser at some point, *possibly LP violated*, new LP store

- second edge: contains the pending information from the first edge

- third edge: independently constructed, contains the necessary information to determine acceptability of the first one

- forth edge: would not be constructed since it contains a violated LP store, completion from the second and third edges

Conclusions

- Formalization of Unification ID/LP grammars and the corresponding parsing schema
- A local solution to a nonlocal phenomenon
- Detection of the violations as early as possible
- No complex backtracking
- Neutral toward processing direction, i.e. bottom up versus top down
- Neutral toward processing mode, i.e. parsing versus generation (Gerdemann (1991)
- Implementation in **Troll** (Gerdemann and Götz forthcoming)

Open Problems

- Grammars with binary branching only
- Discontinuous constituents
- Set–valued feature structures
- Computational complexity

Complexity

- ID/LP parsing is NP–complete
- Unification grammars are undecidable (for example proven in Rounds and Manaster–Ramer (1987))

Nontermination of the completer

```
append([], L, L) -> [string].
append([H|T1], L, [H|T2]) ->
      append(T1, L, T2).
```

Prediction:

```
append([H|T1], L, [H|T2]) ->
      • append(T1, L, T2)
```

Scan:

```
append([], L, L) ->
     [string]  •
```

Completion:

```
append([H], L, [H|L]) ->
     append([], L, L) •
append([X,Y], L, [X,Y|L]) ->
     append([Y], L, [Y|L]) •
...
```

Linguistic Motivation

- We try to sketch the fact that in a reasonably large fragment for German such nonlocal word order phenomena do exist.

- With acceptance of the claims made, exact knowledge of the linguistic background is not necessary.

- We use:
  - the analysis of partial verb phrases by Hinrichs and Nakazawa (1993)
  - the analysis on word order by Lenerz (1977)
  - the empirical analyses by Richter and Sailer (1995)

Rhematic means:

- Greek: rhema $\rightsquigarrow$ statement
- comment
- the negative of thematic, i.e. not known, not old, not mentioned, not presupposed, not contextual. This is only an approximation. Neither of the mentioned criteria is enough.
- word order and stress are important
- in German: either topicalized or as far to the right as possible

16 Example Sentences

(a)

Karl	wird	dem	Kind	das	Geschenk	geben	wollen.
Karl	will	the	child	the	present	give	want.

Karl will want to give the child the present.

(b)

*Karl	wird	das Geschenk	dem Kind	geben	wollen.
		$\begin{bmatrix}\textbf{rhem} & \textit{plus}\\ \textbf{case} & \textit{acc}\end{bmatrix}$	$\begin{bmatrix}\textbf{rhem} & \textit{minus}\\ \textbf{case} & \textit{dat}\end{bmatrix}$		

(c)

Karl	wird	das	Geschenk	dem	Kind	überlassen	wollen.
Karl	will	the	present	the	child	leave	want.

Karl will want to leave the child the present.

(d)

*Karl	wird	dem Kind	das Geschenk	überlassen	wollen.
		$\begin{bmatrix}\textbf{rhem} & \textit{plus}\\ \textbf{case} & \textit{dat}\end{bmatrix}$	$\begin{bmatrix}\textbf{rhem} & \textit{minus}\\ \textbf{case} & \textit{acc}\end{bmatrix}$		

- Connection between the main verb and the order of the complements

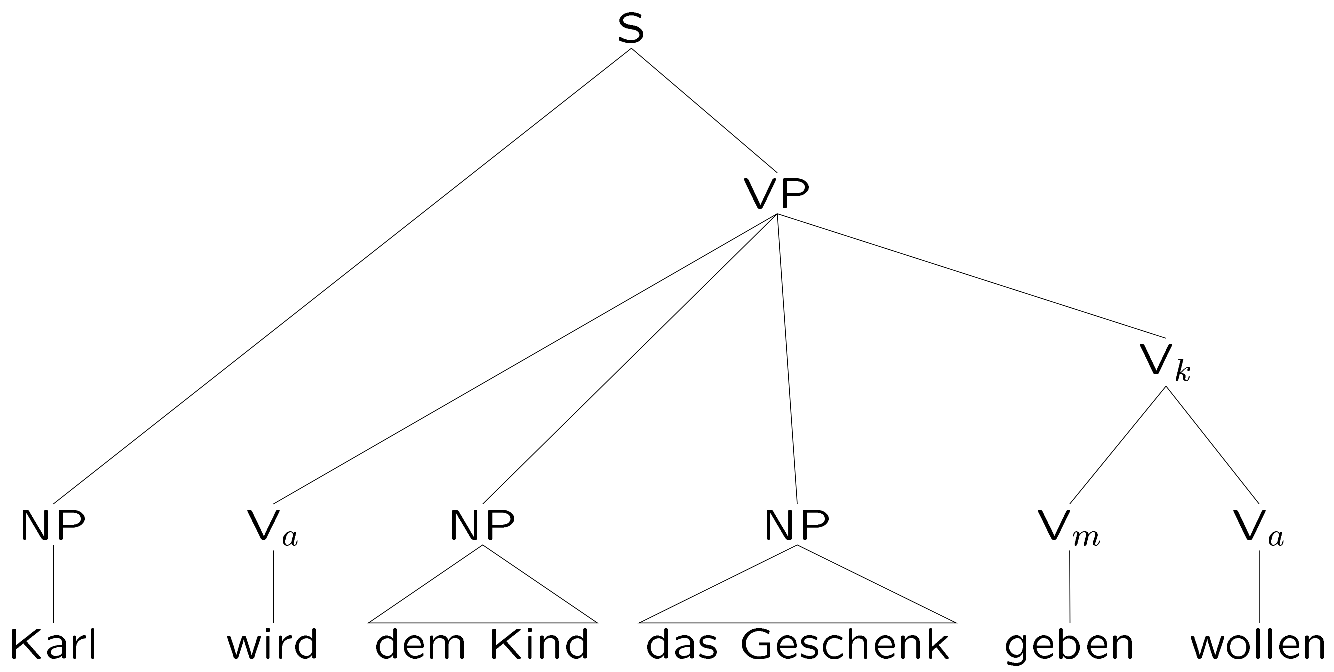
- The *(a)* versions of the sentences do not exhibit any limitation on the order of the complements.

- The *(b)* versions reflect the fact that they do not allow the first NP to be rhematic in the sense of Lenerz (1977)

- *geben*: the accusative rhematic NP must not precede the dative not rhematic NP

- *überlassen*: the dative rhematic NP must not precede the accusative not rhematic NP.

- look at the sketched parse tree


S
VP
V$_k$
NP
V$_a$
NP
NP
V$_m$
V$_a$
Karl
wird
dem Kind
das Geschenk
geben
wollen

- argument raising induces that the complements are not in the same local tree as the main verb

- This is not enough to cause the problem since the complements are still contained in the relevant local tree of the mother

- this leads to topicalization:
  - data not as sharp
  - reasonable that the topicalization does not alter the behaviour of the main verb concerning word order

17 Example Sentences

(a) Geben wollen wird Karl dem Kind das Geschenk.

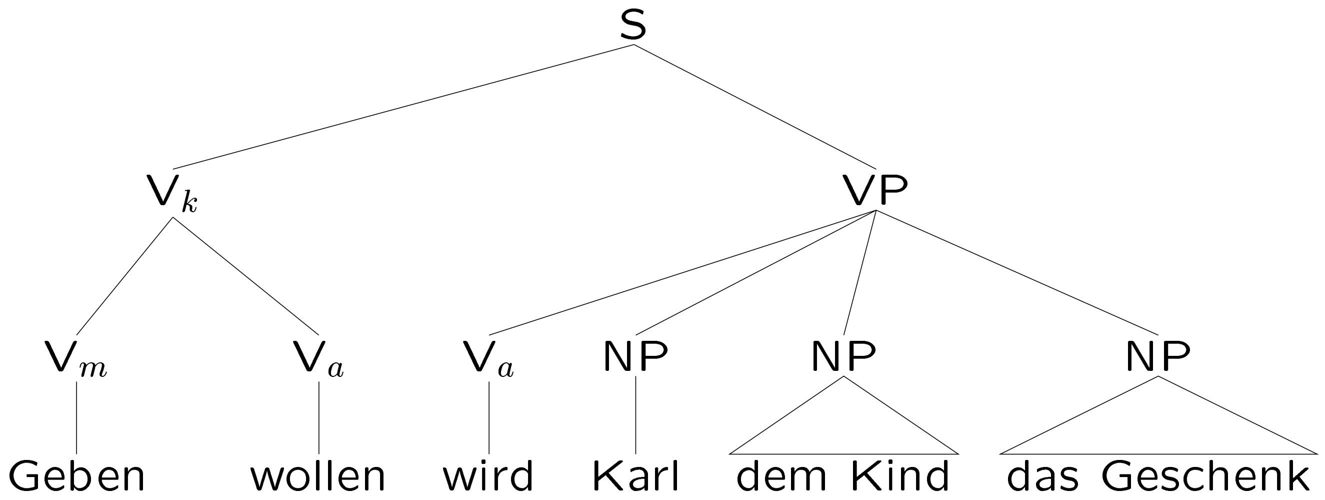
(b) *Geben wollen wird Karl das Geschenk dem Kind.

$$\text{*Geben wollen wird Karl } \underset{\begin{bmatrix}\textbf{rhem} & \textit{plus}\\ \textbf{case} & \textit{acc}\end{bmatrix}}{\text{das Geschenk}} \; \underset{\begin{bmatrix}\textbf{rhem} & \textit{minus}\\ \textbf{case} & \textit{dat}\end{bmatrix}}{\text{dem Kind.}}$$

(c) Überlassen wollen wird Karl das Geschenk dem Kind.

(d) *Überlassen wollen wird Karl dem Kind das Geschenk.

$$\text{*Überlassen wollen wird Karl } \underset{\begin{bmatrix}\textbf{rhem} & \textit{plus}\\ \textbf{case} & \textit{dat}\end{bmatrix}}{\text{dem Kind}} \; \underset{\begin{bmatrix}\textbf{rhem} & \textit{minus}\\ \textbf{case} & \textit{acc}\end{bmatrix}}{\text{das Geschenk.}}$$


S
VP
V_k
V_m
V_a
V_a
NP
NP
NP
Geben
wollen
wird
Karl
dem Kind
das Geschenk