Logical Grammars

- Introduction
- Logic grammars
- Feature grammars
Logical Grammars

- In Phrase structured grammars $\Sigma$ and $V$ were finite sets (simple collections of labels, tagsets)
  - $A \rightarrow \alpha$ $\alpha \in (V \cup \Sigma)^*$
- In Logic Grammars (LG) elements of $V$ and $\Sigma$ can be complex categories, owning internal structure and possibly infinite
  - functors, Prolog terms, feature vectors and matrices, ...

Basic idea
Logical Grammars

• Most known system:
  • Definite clause grammars (DCG)
Feature structures

• Features and complex categories
  • FS (Feature Structures)

• Unification Formalisms (Shieber)
  • Declarative
  • Based on complex categories (FS)
  • Lexicalized (usually based on nuclear CFG).
  • Unification as basic mechanism combination of FS.
  • Constraints over FS as way of representing the well formation rules of the Language.
Feature structures

- Two families:
  - Open Feature Structures
  - Typed Feature Structures
    - every FS belongs to a type
    - The values allowed for assigning to a feature belong to a type.
Feature structures 3

\[
\begin{align*}
\text{he} &: \begin{cases}
\text{synt}: \begin{cases}
\text{pos: det} & \begin{cases}
\text{gen : masc} \end{cases} \\
\text{agreement :} & \begin{cases}
\text{num : sing} \\
\text{person : 3}
\end{cases}
\end{cases} \\
\text{sem} & = \ldots \\
\text{mor} & = \ldots
\end{cases}
\end{align*}
\]
Feature structures 4

FS as DAG (Directed Acyclic Graphs)
Feature structures

- **Path**: sequence of features (a branch of the tree).

- **Constraints** can be expressed as:
  - `<path_1> = <path_2>` (reentrancy)
  - `<path_1> = <value>`
  - `<synt agreement gen> = fem`
Reentrancy

\[
\begin{bmatrix}
  f : [h : a] \\
  g : [h : a]
\end{bmatrix}
\]

\[
\begin{bmatrix}
  f : <1> [h : a] \\
  g : <1>
\end{bmatrix}
\]
Subsumption

• **$FS_1$ subsumes $FS_2$** (is more general, or contains less information) iff:
  • Each feature in $FS_1$ exists in $FS_2$.
  • Whatever value of a feature of $FS_1$ subsumes the corresponding value of $FS_2$.
  • If two values are reentrant in $FS_1$ they are also reentrant in $FS_2$.

\[ FS_1 \sqsubseteq FS_2 \]
Feature structures

FS1 ⊆ FS

\[
\begin{align*}
\text{syt: [cat: n]} & \quad \subseteq \quad \text{syt: [cat: n, agreement: [num: sing]]} \\
\text{[synt: [gen: fem]]} & \quad \subseteq \quad \text{[synt: [gen: fem, agreement: [num: sing]]]}
\end{align*}
\]
The basic operation is **unification**. This operation allows combining the information of two FSs if they compatible (unificable). The unification of $FS_1$ and $FS_2$ is the more general FS subsumed by both $FS = FS_1 \parallel FS_2$.
Feature structures

\[
\begin{align*}
\text{Sint: } & \text{cat: n} \\
\text{Concord: } & \text{gen: fem}
\end{align*}
\]

\[
\begin{align*}
\text{Sint: } & \text{cat: n} \\
\text{Concord: } & \text{num: sing}
\end{align*}
\]
Feature structures

- Prolog terms vs (untyped) FS (DAGs)
  - Prolog terms are a restricted class of DAGs where reentrancy (subgraph sharing) is only allowed for leaves assigned to variables.
  - Both formalisms can represent partial information.
  - Both formalisms allow incremental addition of information.
  - DAGs allow omitting non essential information.
  - DAGs allow, at the same embedding level an order free declaration of features.
- The most known system is PATR II (Shieber)
Feature structures

Parsing with open FS

• Prolog terms vs FS.
  • naive FS representation
    • Gazdar, Mellish, 1989
  • Boyer, 1988
  • P-Patr, Hirsh, 1988
  • Schöter, 1993

• Unification of PT vs Unification of FS.
  • PT unification is (almost) linear
  • DAG unification is $O(n^2)$
Feature structures

\[ X = \begin{cases} \text{the fish} \\ \text{the fish are colourful} \end{cases} \]

\[ X = \begin{cases} \text{the fish} \\ \text{the fish are colourful} \end{cases} \]
Feature structures 14

Parsing with open FS

\[
X = \begin{bmatrix}
\text{head} : \begin{bmatrix}
\text{cat} : \begin{bmatrix}
\text{n} : +
\end{bmatrix}
\end{bmatrix} \\
\text{v} : - \\
\text{agr} : \begin{bmatrix}
\text{per} : 3
\end{bmatrix}
\end{bmatrix}
\text{bar} : 2
\end{bmatrix}
\]

the fish

\text{cat}(+, -, 3, _, _, 2)

\begin{align*}
X : \text{head} : \text{cat} : \text{n} &\leq \text{'+'} \\
X : \text{head} : \text{cat} : \text{v} &\leq \text{'-'} \\
X : \text{head} : \text{agr} : \text{per} &\leq 3 \\
X : \text{bar} &\leq 2
\end{align*}
Representing FS with open Prolog lists (Gazdar, Mellish)

```
[cat: v
 arg0: [cat: np
   case: nom
   num: sing|_]|_]
```

```
[cat: v, arg0: [cat: np, case: nom, num: sing|_]|_]
```
Feature structures

PatrII notation

```
word pepe:

<syn cat> = n
<syn agreement gen> = male
<syn agreement num> = sing
<syn agreement persona> = 3
<sem> = pepe
...
```

PatrII notation

```
[pepe:

[sem : pepe

[mor : ...

syn:

[agreement :

[gen : male

[num : sing

[person : 3

[cat : n

...]]]]]]
```
**Feature structures**

PatrII notation: some syntactic sugar

```
let noun-male-sing be:
    <syn cat> = n
    < syn agreement gen> = male
    < syn agreement num> = sing
    < syn agreement person> = 3.

word pepe:  noun-male-sing
    <sem> = pepe.
```
PatrII inheritance

let verb be:
  <syn cat> = v
  <syn subj cat> = np
  <syn subj case> = nominative.

let vt be:
  verb
  <syn obj1 cat> = np
  <syn obj1 case> = acusative.

let vdat be:
  vt
  <syn obj2 cat> = pp
  <syn obj2 prep> = a.

word laugh:
  verb
  <sem pred> = laugh
  <sem arg1> = human.
(someone laughs)

word give:
  vdat
  <sem pred> = give
  <sem arg1> = human
  <sem arg2> = thing
  <sem arg3> = human.
(someone gives something to someone).
Feature structures

Syntaxic rules in PatRII

X0 --> X1 X2
  <X0 cat> = NP
  <X1 cat> = det
  <X2 cat> = n
  <X1 agree> = <X2 agree>
  <X0 agree> = <X1 agree>.

NP --> det n
  <det agree> = <n agree>
  <NP agree> = <n agree>.
Typed FS

examples: ALE, CUF, TFS, ALEP

Each FS owns an associated type
Each feature has an associated type for its values
Type structure is usually prescribed

bot sub [list, atom]
list sub [e-list ne-list]
e-list sub []
ne-list sub []
intro [hd:bot, tl:list]
atom sub [a, b]
a sub []
b sub []
Feature structures 21

HPSG using ALE

bot  sub [sign, synsem, loc, cat, head, list, nonloc, pform, bool, cont, context, para, index, psoa, con_struc, pers, num, gen, case, vform, ontologia, morfo, wh_str, marking, xbar].
sign  sub [word, phrase]
    intro [phon:list, synsem:synsem].
phrase  sub []
    intro [dtrs:con_struc].
...
cat  sub [cat_subst, cat_funct]
    intro [head:head, subj:list, comps:list, mark:list, fill:list, adj:list,
marking:bool, xbar:xbar].
...
synsem  sub [synsemsubst, synsemfunct]
    intro [loc:loc, nonloc:nonloc].
synsemsubst sub [synsemnoun, synsemverb, synsemadj, synsemprep]
    intro [loc:locsubst].
synsemfunct  sub [synsemdet, synsemmark].
synsemnoun  sub [synsempropi, synsemcomu]
    intro [loc:locnoun].