Computational Linguistics II: Parsing

Definite Clause Grammars

Frank Richter & Jan-Philipp Söhn

fr@sfs.uni-tuebingen.de, jp.soehn@uni-tuebingen.de

December 20th, 2006

Definite Clause Grammars

```
# Grammar Rules
s --> np, vp.          # query: s([john,walks],[]).
np --> n.
vp --> vint.
vp --> vtra, np.

# Lexicon
n --> [john].
n --> [mary].
vint --> [walks].
vint --> [sings].
vtra --> [loves].
```

Definite Clause Grammars – Prolog internal rules

```
# Grammar Rules
s(S1,S2)     :- np(S1,Sr), vp(Sr,S2).      # s --> np, vp.
np(Np1,Np2)  :- n(Np1,Np2).                # np --> n.
vp(Vp1,Vp2)  :- vint(Vp1,Vp2).             # vp --> vint.
vp(Vp1,Vp2)  :- vtra(Vp1,Vr), np(Vr,Vp2). # vp --> vtra, np.

# Lexicon
n([john|R],R).      # n --> [john].
n([mary|R],R).      # n --> [mary].
vint([walks|R],R). # vint --> [walks].
vint([sings|R],R). # vint --> [sings].
vtra([loves|R],R). # vtra --> [loves].
```

Definite Clause Grammars – Giving the structure

```
# Grammar Rules
s([s,[NP,VP]]) --> np(NP), vp(VP).
np([np,[N]]) --> n(N).
vp([vp,[V]]) --> vint(V).
vp([vp,[V,Np]]) --> vtra(V), np(Np).

# Lexicon
n([n,john]) --> [john].
n([n,mary]) --> [mary].
vint([v,walks]) --> [walks].
vint([v,sings]) --> [sings].
vtra([v,loves]) --> [loves].
```

Definite Clause Grammars – Giving the structure

Advantages:

- The user sees the structure of the parse
- A different grammar can be used for parsing than being shown

Disadvantage:

- You can cheat!

Definite Clause Grammars with Features

Example: French Grammar Fragment

Lexical Functional Grammar

Desires:

- easily parseable
- linguistically adequate

→ Joan Bresnan and Ronald Kaplan invent LFG

- distinction between constituent-structure and feature-structure
- unification of feature-structures

Head-Driven Phrase Structure Grammar

- feature-structures contain all information
- mathematically fully fledged-out formalism behind
- model-theoretic interpretation:
  feature-structures represent objects and thus have to be well-typed and sort-resolved