Feature structures and PATR

Detmar Meurers: Intro to Computational Linguistics I
OSU, LING 684.01, 5. March 2003

Overview

- Feature structures
- Combining feature structures: unification
- Descriptions of feature structures
- Rules with feature descriptions: PATR
- How they are interpreted in terms of feature structures
- How to parse with PATR rules

Representing feature structures in Prolog

- The operator ":" is defined to separate `feature:value`
  ```
  ?- op(500,xfy,':').
  ```
- Feature names and atomic values represented by Prolog atoms
- Prolog variables encode structure sharing.
- Each feature structures is represented as a Prolog lists with an open tail:
  - `[person:third, num:sing|_]`
  - `[num:sing, person:third|_]`
  - `[cat:verb, agr:[num:sing|_]|_]`

Unifying feature structures

```
fs_unify(FS,FS) :- !.

fs_unify([Feat:Val | Rest1], FS) :-
   featval(FS,Feat,Val,Rest2),
   fs_unify(Rest1,Rest2)


featval([Feat:Val1 | Rest], Feat, Val2, Rest) :-
   !, fs_unify(Val1,Val2).

featval([FS | Rest], Feat, Val, [FS | Rest2]) :-
   featval(Rest,Feat,Val,Rest2).
```

Towards defining grammar rules in PATR

```
s --> np(Num,Per),
      vp(Num,Per).
```

```
S ---> [NP,VP] :-
  S~[cat]  === s,
  NP~[cat] === np,
  VP~[cat] === vp,
  NP~[agr,per] === X,
  VP~[agr,per] === X,
  NP~[agr,num] === VP~[agr,num].

:- op(500,xfy,'~').
:- op(500,xfx,--->).
:- op(600,xfy,===).
```

Defining the "===" relation between descriptions in terms of the feature structure denotation

```
X === Y :-
   denotes(X,Z),
   denotes(Y,Z).

denotes(X,X) :-
   (var(X);atomic(X)),!.
denotes(FS~Path,Value) :-
   pathval(Path,FS,Value).

pathval([Feat],FS,Value) :-
   !,
   featval(FS,Feat,Value).
pathval([Feat|RestPath],FS,Value) :-
   featval(FS,Feat,FeatVal),
   pathval(RestPath,FeatVal,Value).
```

Using if-then-else in Prolog

For programs of the form

```
pathval([Feat],FS,Value) :-
   !,
   featval(FS,Feat,Value).
pathval([Feat|RestPath],FS,Value) :-
   featval(FS,Feat,FeatVal),
   pathval(RestPath,FeatVal,Value).
```

a special if-then-else syntax is available in Prolog:

```
pathval([Feat|RestPath],FS,Value) :-
   featval(FS,Feat,FeatVal),
   (  RestPath==[]
   -> Value=FeatVal
   ;  pathval(RestPath,FeatVal,Value) ).
```

Grammar rules in a PATR-like system

```
:- op(500,xfx,--->).
:- op(500,xfy,'/').
:- op(600,xfy,===).
:- op(500,xfy,'~').

W ---> [mary] :-
   W~[cat]      === np,
   W~[agr,per] === third,
   W~[agr,num] === sing.
```

```
S ---> [NP,VP] :-
   S~[cat]   === s,
   NP~[cat]  === np,
   VP~[cat]  === vp,
   NP~[agr]  === VP:agr.

W ---> [left] :-
   W~[cat]      === vp,
   W~[agr,per] === third,
   W~[agr,num] === sing.
```

A left-corner parser for grammars in CNF with atomic categories

```
:- op(1100,xfx,'--->').

% ?- recognize(Start,<list(word)>,[]).
recognize(Phrase,[Word|S1],S) :-
   (Cat ---> [Word]),
   lc(Cat,Phrase,S1,S).

lc(Phrase,Phrase,S,S).

lc(SubPhrase,SuperPhrase,S0,S) :-
    Phrase ---> [SubPhrase,Right],
    recognize(Right,S0,S1),
    lc(Phrase,SuperPhrase,S1,S).
```

A left-corner parser for PATR grammars in CNF
(mini_patr.pl)

```
:- op(1100,xfx,'--->').

% ?- recognize(Start,<list(word)>,[]).
recognize(PhraseFS,[Word|S1],S) :-
   (CatFS ---> [Word]),
   lc(CatFS,PhraseFS,S1,S).

lc(Phrase1FS,Phrase2FS,S,S) :-
   fs_unify(Phrase1FS,Phrase2FS).

lc(SubPhraseFS,SuperPhraseFS,S0,S) :-
    PhraseFS ---> [LeftFS,RightFS],
    fs_unify(LeftFS,SubPhraseFS),
    recognize(RightFS,S0,S1),
    lc(PhraseFS,SuperPhraseFS,S1,S).
```