Implementing HPSG grammars
Part II: The TRALE system

Detmar Meurers
OSU, LING795K, Spring 2002



## The components of a Trale grammar

- Signature
- Descriptions
- Phrase Structure
- Implicational Constraints
- A second look at descriptions: abbreviations
- Relations (called using relational or functional notation)
- Lexical Rules



## Signature: declaring the basic vocabulary

The signature declares the basic set of *type* and *feature* names for use in descriptions.

These names must start with a lower case letter and continue with `a-z`, `A-Z`, `0-9`, or `_`.



## Signature: Example

```
type_hierarchy
bot
   sign
      phrase dtr:list
      word cat:cat
   cat
      noun
      verb
   list
     ne_list hd:bot tl:list
     e_list
.
```

## Signature: Example with multiple inheritance

```
type_hierarchy
bot
   sign
      phrase dtr:list
      word cat:cat
   cat
      nounish
        noun
        gerund
      verbish
        verb
        &gerund
   list
     ne_list hd:bot tl:list
     e_list
.
```



## Descriptions

A description consists of

- a type specification: word starting with lower letters
- a structure sharing: word starting with a capital letter
- a compound description: *path*:*description*
  with *path* consisting of *feature* or *feature*:*path*



## Phrase Structure (I)

```
% I. Lexicon
john ---> (word,phon:[(a_ john)],cat:noun).
left ---> (word,phon:[(a_ left)],cat:verb).

% II. Phrase Structure Rule:
subj_head_rule rule
   (phrase,phon:[SubjPhon,HeadPhon],dtrs:[Subj,Head]) ===>
cat> (Subj,word,phon:[SubjPhon],cat:noun),
cat> (Head,word,phon:[HeadPhon],cat:verb).
```



## Phrase Structure (II)

```
:- tree_extensions.
:- multifile if/2.

% I. Lexicon
john ~~> (word,cat:noun).
left ~~> (word,cat:verb).

% II. Phrase Structure Rule:
subj_head_rule ##
   phrase
===>
    cat> (word,cat:noun),
    cat> (word,cat:verb).
```

## Implicational Constraints

HFP as specified in Pollard and Sag (1994)

$$\begin{bmatrix} \textit{phrase} \\ \text{DTRS}\ \textit{headed-structure} \end{bmatrix} \rightarrow \begin{bmatrix} \text{SYNSEM|LOC|CAT|HEAD} & \boxed{1} \\ \text{DTRS|HEAD-DTR|SYNSEM|LOC|CAT|HEAD} & \boxed{1} \end{bmatrix}$$

can be specified in Trale as:

```
(phrase, dtrs:headed_structure)  *>
                    (synsem:loc:cat:head:H,
                     dtrs:head_dtr:synsem:loc:cat:head:H).
```



## A second look at descriptions
### Example from HPSG: Abbreviation used in Pollard and Sag (1994)

Abbreviation Abbreviated AVM

NP$_{\boxed{1}}$

$$\begin{bmatrix} \textit{synsem} \\ \text{LOCAL} \begin{bmatrix} \text{CATEGORY} & \begin{bmatrix} \text{HEAD} & \textit{noun} \\ \text{SUBCAT} & \langle\rangle \end{bmatrix} \\ \text{CONTENT|INDEX} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

VP:$\boxed{1}$

$$\begin{bmatrix} \textit{synsem} \\ \text{LOCAL} \begin{bmatrix} \text{CATEGORY} & \begin{bmatrix} \text{HEAD} & \textit{verb} \\ \text{SUBCAT} & \left\langle \textit{synsem} \right\rangle \end{bmatrix} \\ \text{CONTENT} & \boxed{1} \end{bmatrix} \end{bmatrix}$$



## Another look at descriptions
### Realization in Trale using logical variable macros

**Definition:**

```
np(Index-index):=  (local:(category:(head:noun,
                                     subcat:[])
                           content:index:Index)).
vp(Cont-cont):=  (local:(category:(head:verb,
                                   subcat:[synsem])
                           content:Cont)).
```

**Use:**

```
john ~~> (word, synsem: @np((per:third,num:sing))).
left ~~> (word, synsem: (@vp((leave-rel,leaver:Index)),
                         local:category:subcat:hd:@np(Index))).
```



## Expressing relational dependencies
### on lexical entries, phrase structure rules or implicational constraints

**Definition:**

```
append([],ne_list,L) if true.
append([H|T1],L,[H|T2]) if append(T1,L,T2).

synsem2sign(e_list,e_list) if true.
synsem2sign([H|T],[synsem:H|NewT]) if synsem2sign(T,NewT).
```

**Use:**

```
phrase *> (synsem:category:subcat:PhrSubcat,
           dtrs: (head_dtr:synsem:category:subcat:HeadSubcat,
                  comp_dtrs:CompDtrs)
goal
   synsem2sign(CompSynsems,CompDtrs),
   append(CompSynsems,PhrSubcat,HeadSubcat).
```

## Functional notation for relations
### Example from HPSG: The Subcat Principle

$$\left[\text{DTRS } \textit{headed-structure}\right] \rightarrow \left[\begin{array}{l}\text{SYNSEM|LOC|CAT|SUBCAT } \boxed{1} \\ \text{DTRS } \left[\begin{array}{l}\text{HEAD-DTR|SYNSEM|LOC|CAT|SUBCAT } \textit{append}\left(\boxed{1},\boxed{2}\right) \\ \text{COMP-DTRS } \textit{synsem2sign}\left(\boxed{2}\right)\end{array}\right]\end{array}\right]$$

with the relations append/3 and synsem2sign/2 defined as follows

$$\begin{array}{lcl}\textit{append}\left(\langle\rangle,\boxed{1}\right) & := & \boxed{1}. \\ \textit{append}\left(\left\langle\boxed{1}|\boxed{2}\right\rangle,\boxed{3}\right) & := & \left\langle\boxed{1}|\textit{append}\left(\boxed{2},\boxed{3}\right)\right\rangle.\end{array}$$

$$\begin{array}{lcl}\textit{synsem2sign}\left(\langle\rangle\right) & := & \langle\rangle. \\ \textit{synsem2sign}\left(\left\langle\boxed{1}|\boxed{2}\right\rangle\right) & := & \left\langle\left[\text{SYNSEM } \boxed{1}\right]|\textit{synsem2sign}\left(\boxed{2}\right)\right\rangle.\end{array}$$



## Functional notation for relations
### A Trale encoding of the Subcat Principle

```
(dtrs:headed-structure) *>
   (synsem:loc:cat:subcat:MotherSubcat,
    dtrs: (head_dtr:synsem:loc:cat:subcat:
                              append(MotherSubcat,CompSynsems)
           comp_dtrs: synsem2sign(CompSynsems)))

fun append(+,+,-).
append([],ne_list,L) if true.
append([H|T1],L,[H|T2]) if
  append(T1,L,T2).

fun synsem2sign(+,-).
synsem2sign(e_list,e_list) if true.
synsem2sign([H|T],[synsem:H|synsem2sign(T)]) if true.
```



## Lexical rules

```
psp_lex_rule ##
   (synsem:loc:(cat:head:(vform:base,
                          aux:Aux),
                cont:Cont))
**>
   (synsem:loc:(cat:head:(vform:psp,
                          aux:Aux),
                cont:(perfect_rel,
                      soa_arg:Cont)))
morphs
    be becomes been,
    give becomes given,
    (X,[e]) becomes (X,ed),
     X becomes (X,ed).
```



## Some basic practical things

- A basic grammar consists of two files: `theory.pl` and `signature`
- To start Trale with the Grisu interface, `cd` to the directory containing your grammar, then type `trale -g`
- Compile a grammar using: `compile_gram(theory).` or short `c.`
- Sentences are parsed using: `rec [put,your,sentence,here].`

Consult the TRALE manual (Part 1: ALE, Part 2: TRALE), available from the course web page, for more information.

The web page also contains links to some Trale example grammars, numbered from simple to more complex.

# References

Pollard, Carl and Ivan A. Sag (1994). *Head-Driven Phrase Structure Grammar*. Chicago, IL: University of Chicago Press.