

Notes on the Milca English Resource Grammar

Kordula De Kuthy, Vanessa Metcalf, Detmar Meurers

Columbus, June 3, 2003

Overview

- The LinGO English Resource Grammar
- The LKB (Linguistic Knowledge Building) System
- Implementing a grammar in LKB
- The Trale (Type Resolution Attribute Logic Engine) System
- Implementing a grammar in Trale
- Sample analyses from the Trale grammar

The LinGO English Resource Grammar

- a large-scale grammar for English based on the HPSG framework
- developed as part of the Linguistics Grammars Online (LinGO) project at the Center for the Study of Language and Information (CSLI) at Stanford University
- implemented in the LKB grammar development system

Coverage of the English Resource Grammar

Basic declarative sentences

- intransitive verbs:
 - (1) Abrams works.

- transitive verbs:
 - (2) Abrams hired Browne.

- ditransitive verbs:
 - (3) a. Abrams showed the office to Browne.
b. Abrams showed Browne the office.

- sentential complements:

(4) Abrams bet Browne five dollars that Chiang hired Devito.

- predicatives:

(5) a. Abrams became competent.
b. Abrams became a manager.

- prepositional complements:

(6) a. Abrams works for Browne.
b. * Abrams works of Browne.
c. Abrams approves of Browne.
d. * Abrams approves for Browne.

- impersonal constructions

- (7) a. It is time for an interview.
- b. It is true that Abrams hired Browne.
- c. There is a bookcase in the office.
- d. There are programmers.
- e. There are programmers interviewing Devito.
- f. There are programmers older than Devito.
- g. There stands in the office a bookcase.
- h. There is Abrams, Browne, and Chiang.

- auxiliaries/modal verbs

- (8) a. Abrams may hire Browne.
- b. Abrams can hire Browne.
- c. Abrams did hire Browne.

- d. Abrams has hired Browne.
- e. Abrams had better hire Browne.
- f. Abrams better hire Browne.
- g. Abrams could have hired Browne.
- h. Abrams could be hiring Browne.
- i. Abrams could have been hiring Browne.

- control verbs

- (9) a. Abrams promised Browne to hire Chiang.
- b. He promised us to evaluate himself.
- c. *Abrams promised there to be a bookcase in the office.
- d. Abrams promised Browne to be interviewed by Chiang.
- e. Abrams urged Browne to hire Chiang.
- f. Browne was urged to hire Chiang.

- g. Abrams urged Browne to be interviewed.
- h. Abrams was urged to be interviewed.
- i. Abrams appealed to Browne to hire Chiang.
- j. Abrams appealed to Browne to be interviewed.

- tough constructions

- (10) a. It was hard to show an office to Chiang.
- b. Chiang was hard to show an office to.
- c. Offices are hard to show to Chiang.
- d. Chiang was hard for Abrams to show an office to.
- e. Offices are hard for Abrams to show to Chiang.
- f. Chiang was hard to show an office.

- passive

- (11)
- a. Abrams was hired (by Browne).
 - b. An office was shown to Abrams (by Chiang).
 - c. Chiang was shown an office by Abrams.
 - d. Abrams was urged to hire Browne by Chiang.
 - e. Abrams was urged by Chiang to hire Browne.
 - f. Abrams was known (by Chiang) to be interviewing Browne.
 - g. Abrams was made to interview Browne.
 - h. Abrams got hired (by Browne).
 - i. Abrams had Browne hired (by Chiang).

- inversion

- (12)
- a. That office, the consultants work in.
 - b. To Browne, Abrams showed an office.
 - c. Competent, Abrams is.
 - d. Hire Browne, Abrams did.
 - e. Not one programmer did Abrams hire.
 - f. At no time did Abrams hire a programmer.
 - g. No more competent was Abrams.
 - h. A trustworthy employee is Abrams.
 - i. In the office is the bookcase.
 - j. Competent is the manager who hired Abrams.
 - k. Standing in the office is the bookcase.
 - l. Never does Abrams work with Browne.
 - m. In the office is a good location for the bookcase.

- unbounded dependencies

- (13) a. Which manager did Abrams know was interviewing programmers.
b. Which managers did Abrams know were interviewing programmers.
c. Which department is Abrams the manager of?
d. Whose department does Abrams work in?

- comparatives

- (14) a. Chiang is (two days) older than Browne.
b. Abrams is (two days) older than thirty years.
c. Abrams is more competent than Browne.
d. Abrams is as competent as Browne.
e. Abrams is the oldest manager.

- f. Abrams is the most competent manager.
- g. Abrams manages more programmers than Browne manages.
- h. Abrams manages as many programmers as Browne manages.
- i. Abrams interviewed more programmers than were hired.

- conjunction

- (15)
 - a. Chiang is a manager and Devito is a programmer.
 - b. Chiang and Devito work.
 - c. Browne, Chiang, and Devito work.
 - d. Browne and Chiang and Devito work.
 - e. Both Chiang and Devito work.
 - f. Either Chiang or Devito works.
 - g. Neither Chiang nor Devito works.
 - h. Chiang hired Devito and manages Browne.

- ellipsis

- (16)
- a. Abrams was interviewing programmers, and Browne was, too.
 - b. Abrams was interviewing programmers, and so was Browne.
 - c. Abrams was interviewing programmers, or Browne was.
 - d. Abrams wasn't interviewing programmers, nor was Browne.
 - e. Abrams wasn't interviewing programmers, and neither was Browne.
 - f. Browne wasn't interviewed by Devito, but he was by Chiang.
 - g. Abrams doesn't consult for Browne, but he does for Chiang.

- binding

- (17)
- a. Devito knew that he had hired a programmer.
 - b. The manager evaluated her staff.
 - c. Her manager interviewed Browne.

- d. The person who manages him knows that Browne is competent.
- e. Devito interviewed programmers. He hired Browne.
- f. Browne evaluated himself.
- g. The managers evaluated each other.

- adverb placement

- (18)
- a. Evidently Chiang works.
 - b. Chiang evidently works.
 - c. Chiang works, evidently
 - d. Chiang has evidently been leaving.
 - e. Evidently Chiang showed Devito an office.
 - f. Chiang evidently showed Devito an office.
 - g. * Chiang showed evidently Devito an office.
 - h. * Chiang showed Devito evidently an office.

- quantifier

- (19)
- a. Every manager who interviewed a programmer hired him.
 - b. If a programmer was interviewed by every manager, he was hired.
 - c. Every engineer who has a bookcase arrived.
 - d. The engineer who has a bookcase arrived.
 - e. Every manager interviewed one of the programmers.
 - f. Two programmers were interviewed by every manager.

- negation

- (20)
- a. Devito has not hired Abrams.
 - b. Devito could not hire Abrams.
 - c. Devito could not have hired Abrams.
 - d. Devito could not be hiring Abrams.

- e. Devito did not hire Abrams.
- f. * Devito hired not Abrams
- g. Devito is not a manager.

Interrogative sentences

- direct questions

- (21) a. She works for whom?
- b. She works for who?
- c. She manages whom?
- d. She manages who?
- e. She showed whom an office?
- f. She showed who an office?
- g. Whom does she work for?

- h. Who does she work for?
- i. For whom does she work?
- j. * For who does she work?
- k. Who hired Browne?

- embedded interrogatives

- (22) a. Abrams does not know who hired Browne.
- b. Abrams does not know which managers interviewed Browne.
- c. Abrams does not know who Browne hired.
- d. Abrams does not know who showed Browne an office.
- e. Abrams does not know who(m) Browne was hired by.
- f. Abrams does not know by whom Browne was hired.
- g. Abrams does not know which programmers Browne hired.
- h. Abrams does not know where Browne works.

- i. Abrams does not know when Browne was hired.
- j. Abrams does not know how many consultants Browne has hired.
- k. Abrams does not know how competent Browne is.

- selection of interrogatives

- (23)
- a. Kim knows whether Sandy sleeps.
 - b. Kim knows whether to sleep.
 - c. * Kim knows whether Sandy to sleep.
 - d. Kim knows when to sleep.
 - e. Kim knows what to think Sandy likes.
 - f. Kim knows who to think Sandy was hired by.
 - g. Kim knows if Sandy sleeps.
 - h. Kim wonders whether Sandy sleeps.
 - i. Kim wonders whether to sleep.

- tag questions

- (24)
- a. It is raining, isn't it?
 - b. There is a meeting, isn't there?
 - c. * It is raining, isn't there?
 - d. Sara is sleeping, isn't she?
 - e. * Sara is sleeping, isn't there?
 - f. Sara will sleep, won't she?
 - g. * Sara will sleep, can't she?
 - h. Sara sleeps, doesn't she?
 - i. Everyone slept, didn't they?

Imperative sentences

- (25)
- a. Hire a programmer!
 - b. Be trustworthy!
 - c. *Are trustworthy!
 - d. Do not hire a programmer!
 - e. Don't hire a programmer!
 - f. You hire a programmer!
 - g. You be trustworthy!
 - h. Don't you hire a programmer!
 - i. Don't you be trustworthy!
 - j. Everyone hire a programmer!
 - k. Everyone be trustworthy!
 - l. Don't anyone hire a programmer!
 - m. Don't anyone be trustworthy!

Noun phrases in the ERGO grammar

Pronouns

- **Personal pronouns: Case**

He hired her.

*She hired he.

- **Coordination**

She and I interviewed Abrams.

I and she interviewed Abrams.

Me and her interviewed Abrams.

Her and me interviewed Abrams.

*Her and I interviewed Abrams.

*She and me interviewed Abrams.

- **Reflexives**

The woman evaluated herself.

- **Generic pronoun**

Anyone with an office works.

Head-Specifier constructions

The programmer was hired.

This programmer was hired.

These programmers were hired.

That programmer was hired.

Those programmers were hired.

Some programmer was hired.

Some programmers were hired.

*Any programmer was hired.

*Any programmers were hired.

Most programmers were hired.

Few programmers were hired.

No programmers were hired.

Many programmers were hired.

A dozen programmers were hired. (NP)

One programmer was hired. (mine)

More managers were interviewed.

Only the more competent programmers were interviewed.

Every manager who interviewed a programmer hired him.

Not one programmer did Abrams hire. (NP)

Head-Adjunct Constructions

- **Premodifiers**

Abrams sees Browne as a competent manager.

Abrams bet Browne five dollars that Chiang hired Devito.

- **Postmodifiers**

- **Prepositional phrases**

It is time for an interview.

Abrams has an office with a bookcase.

- **Relative clauses**

Abrams has an office that Browne showed Chiang.

Abrams has an office which Browne showed Chiang.

Abrams has an office Browne showed Chiang.

Abrams hired a woman that Browne interviewed.

Abrams hired a woman Browne approved of.

– **Verb phrases**

Abrams hired a woman interviewed by Chiang.

Abrams hired a woman working for Chiang.

Other kinds of noun phrases

- **Partitive constructions**

None of the consultants work for Abrams.

Five of the seven consultants work for Abrams.

Any five of the seven consultants can work for Abrams.

Most of the staff is competent.

Most of the program works.

Almost all of the program works.

Five teams of programmers were hired.

Half of the programmers were interviewed by Abrams.

Half the programmers were interviewed by Abrams.

Fewer than half the programmers were hired. (NP)

Fewer than half of the programmers were hired. (NP)

An number of programmers were interviewed. (NP)

An number of programmers was interviewed.

- **Head-complement NPs**

All but seven programmers were hired.

Titles

Browne is the manager.

Lee Browne is the manager.

Mr. Browne is the manager.

Mr. Lee Browne is the manager.

A Mr. Lee Browne is the manager.

A Lee Browne is the manager.

Lee Browne, Esq is the manager.

Dates

Browne was hired on 1/3/84.

Browne was hired on the 1st of January, 1984.

Browne was hired on January 1st, 1984.

The LKB (Linguistic Knowledge Building) System

- A grammar and lexicon development environment for use with constraint-based formalisms
- intended to be used for NLP research involving unification-based linguistic formalisms for parsing and/or generation
- implemented in Common Lisp
- designed to be efficient with large grammars

Implementing a grammar in LKB

The Type Hierarchy

- There must be a single hierarchy containing all types with a unique top type
- There is no distinction between the declaration of a type, its appropriateness conditions and constraints on that type.
- Unique greatest lower bounds are inserted automatically by the LKB system.
- No recursive type declarations.

```
sign := basic_sign &  
  [ SYNSEM synsem ].
```

```
phrase := sign &  
  [ SYNSEM canonical_synsem &  
    [ LOCAL.CONT [ TOP #hand,  
                  INDEX #index,  
                  E-INDEX #event ] ],  
    C-CONT mrs_min & [ TOP #hand,  
                      INDEX #index,  
                      E-INDEX #event ] ].
```

invalid type declaration:

```
list := top &  
  [ FIRST top,  
    REST list ].
```

Implementing a grammar in LKB

Phrase Structure Rules

- are specified using the same syntax as for type declarations.
- The daughters must be specified as elements on the `ARGS` list.
- This need not be part of the rule declaration itself, but can also be part of constraints on subtypes of the mother of the rule.

```
hcomp := hcomp_rule &  
  [ RULE-NAME 'hcomp '].
```

```
hcomp_rule := binary_rule_left_to_right & head_comp_phrase.
```

```
binary_rule_left_to_right := rule &  
  [ ARGS < [ KEY-ARG + ] , [ KEY-ARG bool ] > ].
```

Implementing a grammar in LKB

Lexical entries

- are specified using the same syntax as for type declarations.
- The orthography of the lexical entry must be specified with the help of an attribute.

```
sleep_v1 := v_unerg_le &  
  [ STEM < "sleep" >,  
    SYNSEM.LOCAL.KEYS.KEY _sleep_rel ].
```

```
v_unerg_le := main_verb &  
  [ SYNSEM unerg_verb ].
```

```
main_verb := main_verb_sans_key & topkey & mcna & nonmsg &  
  [ INFLECTED -,  
    SYNSEM.LOCAL [ KEYS.KEY #key,  
                   CONT.LISZT <! #key !> ] ].
```

Implementing a grammar in LKB

Lexical Rules

- are again specified using the same syntax as for type declarations.
- The input of the lexical rule is specified as the element on the `ARGS` list.

```
third_sg_fin_verb_infl_rule :=  
%suffix (!s !ss) (!ss !ssses) (ss sses) (!ty !ties) (ch ches) (sh shes) (x xes) (z zes)  
lex_rule_infl_affixed &  
[ NEEDS-AFFIX +,  
  SYNSEM.LOCAL third_sg_fin_verb ].
```

```
lex_rule_infl_affixed := lex_rule_compos &  
[ INFLECTED +,  
  KEY-ARG #keyarg,  
  SYNSEM #synsem,  
  ARGS < #dtr >,  
  DTR #dtr & [ INFLECTED -,  
               KEY-ARG #keyarg,  
               SYNSEM #synsem ].
```

Implementing a grammar in LKB

Macros

- The system does not provide any means of abbreviating descriptions, i.e. macros.
- Instead, types are very often used to abbreviate certain descriptions.

List Operations

- The LKB system does not provide any kind of list operations.
- List operations, like append, have to be encoded with the help of difference lists.

```

basic_two_arg := lex_synsem &
  [ LOCAL.ARG-S < [ NON-LOCAL [ SLASH [ LIST #smiddle,
                                     LAST #slast ],
                                     REL [ LIST #rmiddle,
                                          LAST #rlast ],
                                     QUE [ LIST #qmiddle,
                                          LAST #qlast ] ] ],
    LOCAL.CONT.INDEX individual ],
  [ NON-LOCAL [ SLASH [ LIST #sfirst,
                       LAST #smiddle ],
               REL [ LIST #rfirst,
                    LAST #rmiddle ],
               QUE [ LIST #qfirst,
                    LAST #qmiddle ] ] ],
    LOCAL.CONT.INDEX individual ] >,
NON-LOCAL [ SLASH [ LIST #sfirst,
                  LAST #slast ],
           REL [ LIST #rfirst,
                LAST #rlast ],
           QUE [ LIST #qfirst,
                LAST #qlast ] ] ] ].

```

The Trale (Type Resolution Attribute Logic Engine) System

- Combination of the two systems ALE and Control:
 - Interpretation of the signature (closed-world reasoning) are taken from Control
 - Universal principles with complex antecedents
 - The parser and description language are based on ALE.

Implementing a grammar in Trale

The Type Hierarchy

- All types and their appropriateness conditions must be declared in the signature.
- Constraints on any type must be declared separately

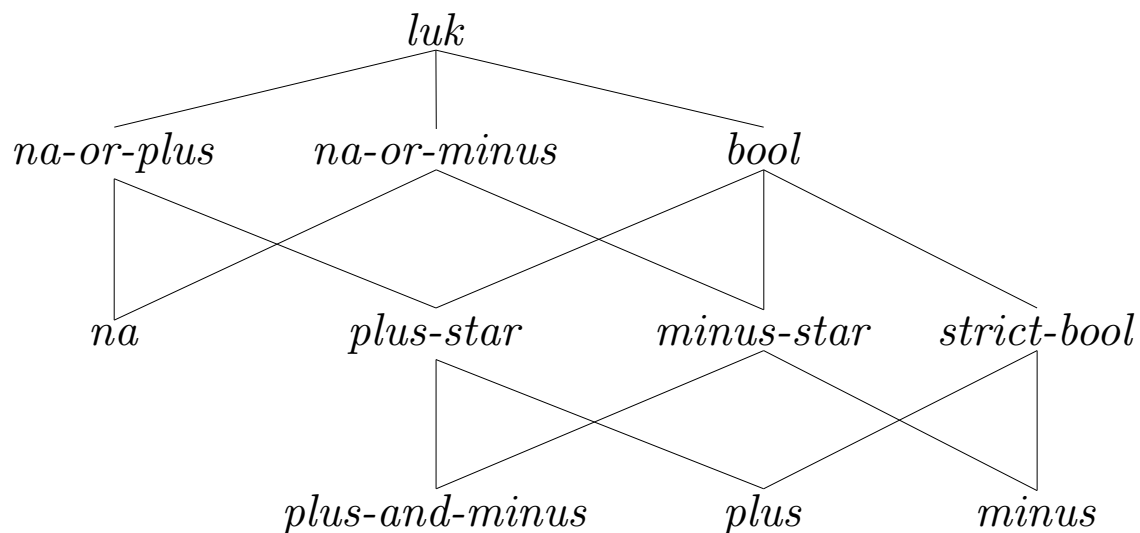
```
sign synsem:canonical_synsem
  phrase dtrs:list c_cont:mrs_min
    binary_phrase
      basic_head_filler_phrase
      basic_binary_headed_phrase
    unary_phrase
      basic_head_opt_comp_phrase
```

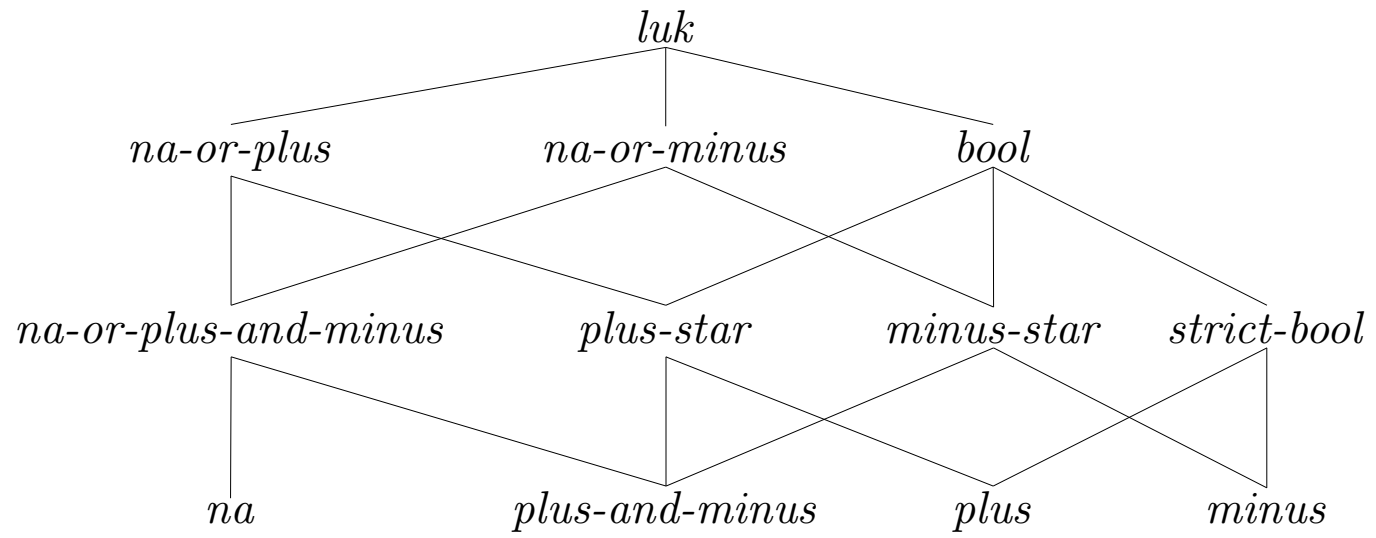
```
phrase *>
  ( synsem:( canonical_synsem,
             local:cont:( top:Hand,
                          index:Index,
                          e_index:Event)),
    c_cont:( top:Hand,
             index:Index,
             e_index:Event)).
```

Implementing a grammar in Trale

Unique greatest lower bounds

The Trale system requires the specification of all unique greatest lower bounds in the signature.





Implementing a grammar in Trale

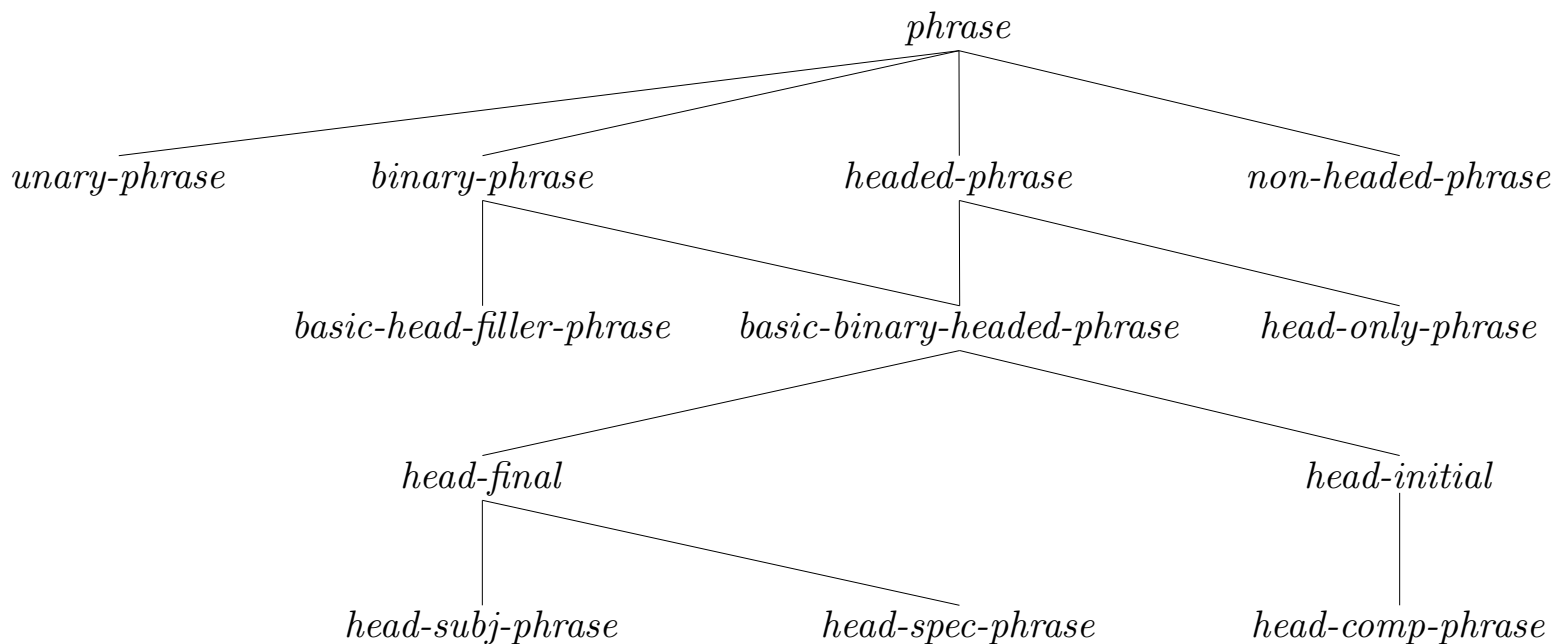
Phrase Structure Rules

- are specified using a special syntax.
- Each of the daughters must be explicitly mentioned in the declaration of the rule.

```
hcomp ## @head_comp_phrase(HeadDtr,NonheadDtr)
  ==>
cat> HeadDtr,
cat> NonheadDtr.
```

```
head_comp_phrase(HeadDtr,NonheadDtr) macro (head_comp_phrase,
                                             @head_comp_or_marker_phrase(NonheadDtr),
                                             synsem: (local:( cat:posthead:Ph,
                                                             conj:cnil),
                                                         lex:Lex),
                                             head_dtr: (HeadDtr, synsem:local:cat:( posthead:Ph,
                                                                                   hc_lex:Lex )) ).
```

Phrasal types



Implementing a grammar in Trale

Lexical entries

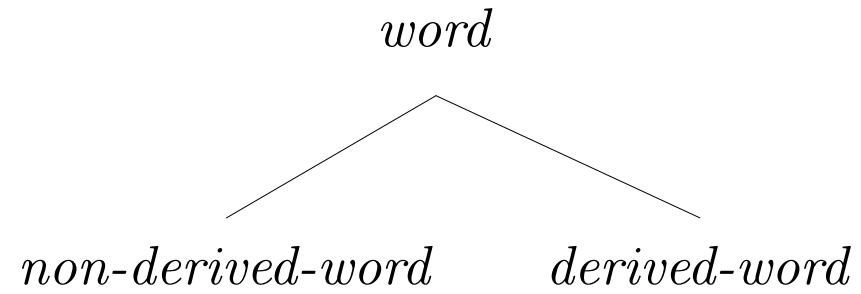
- A special syntax is used for lexical entries.
- The left hand side of the lexical entry corresponds to the orthography of the word.
- Macros can be used as part of the description.

```
sleep  ~~> @v_unerg_le.
```

```
v_unerg_le macro (@main_verb,  
                 synsem: @unerg_verb).
```

```
main_verb macro (@main_verb_sans_key, @mcna, @nonmsg, @topkey,  
                inflected:minus,  
                synsem:local:( keys:key:Key,  
                               cont:liszt:[Key]))).
```

Lexical types



Implementing a grammar in Trale

Lexical Rules

```
third_sg_fin_verb_infl_rule ##
```

```
  Dtr
```

```
**>
```

```
( lr:lex_rule_infl_affixed,  
  needs_affix:plus,  
  synsem:local: @third_sg_fin_verb,  
  @lex_rule_infl_affixed_macro(Dtr))
```

```
morphs
```

```
  X      becomes Y when irregpres(X,Y),  
  (X,S,s) becomes (X,S,ssess) when letter_set_s(S),  
  (X,ss)  becomes (X,sses),  
  (X,T,y) becomes (X,T,ies) when letter_set_t(T),  
  (X,ch)  becomes (X,ches),  
  (X,sh)  becomes (X,shes),  
  (X,x)   becomes (X,xes),  
  (X,z)   becomes (X,zes),  
  (X,C)   becomes (X,C,s) when letter_set_s(C).
```

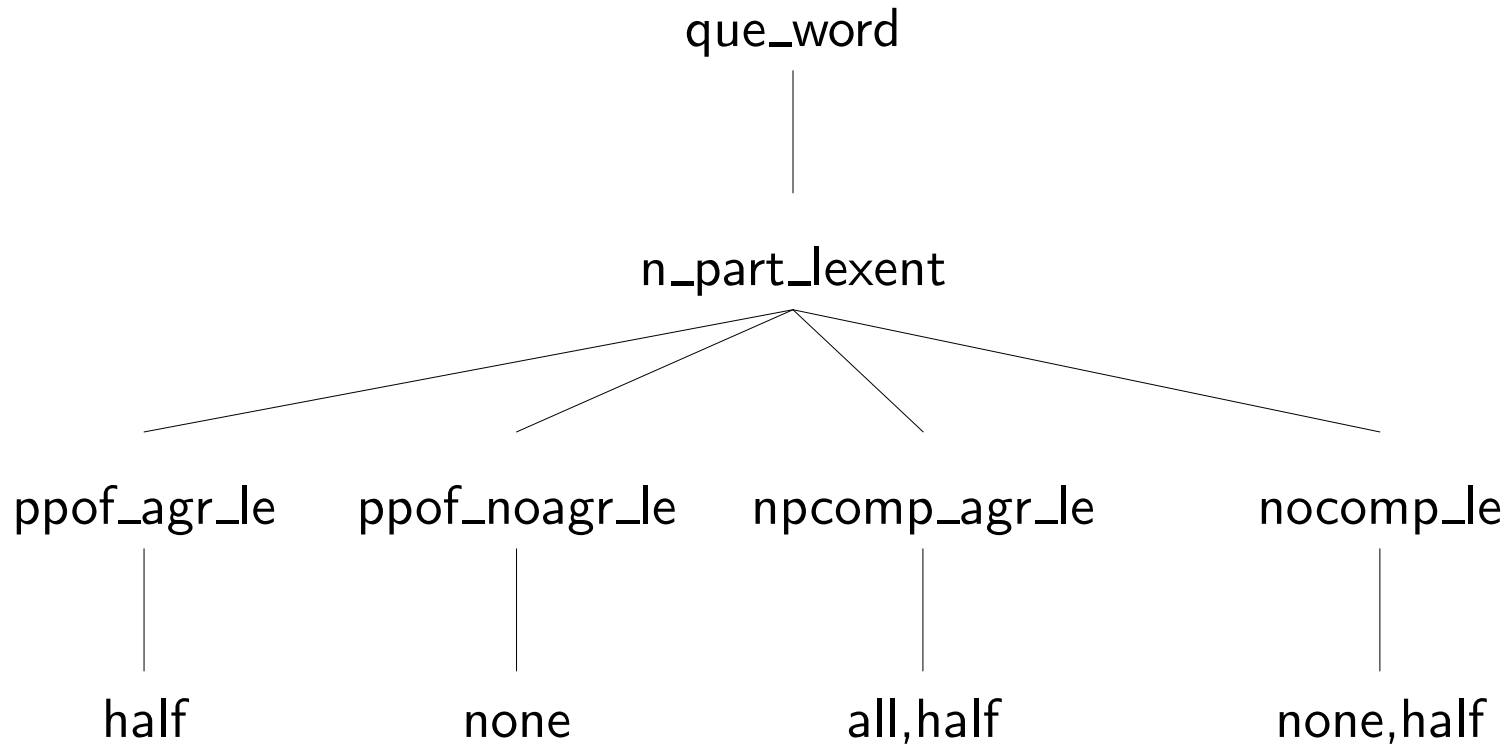
Implementing a grammar in Trale

List Operations

- List operations can be specified by defining relations over feature descriptions.
- These relations can be called by phrase structure rules and principles.

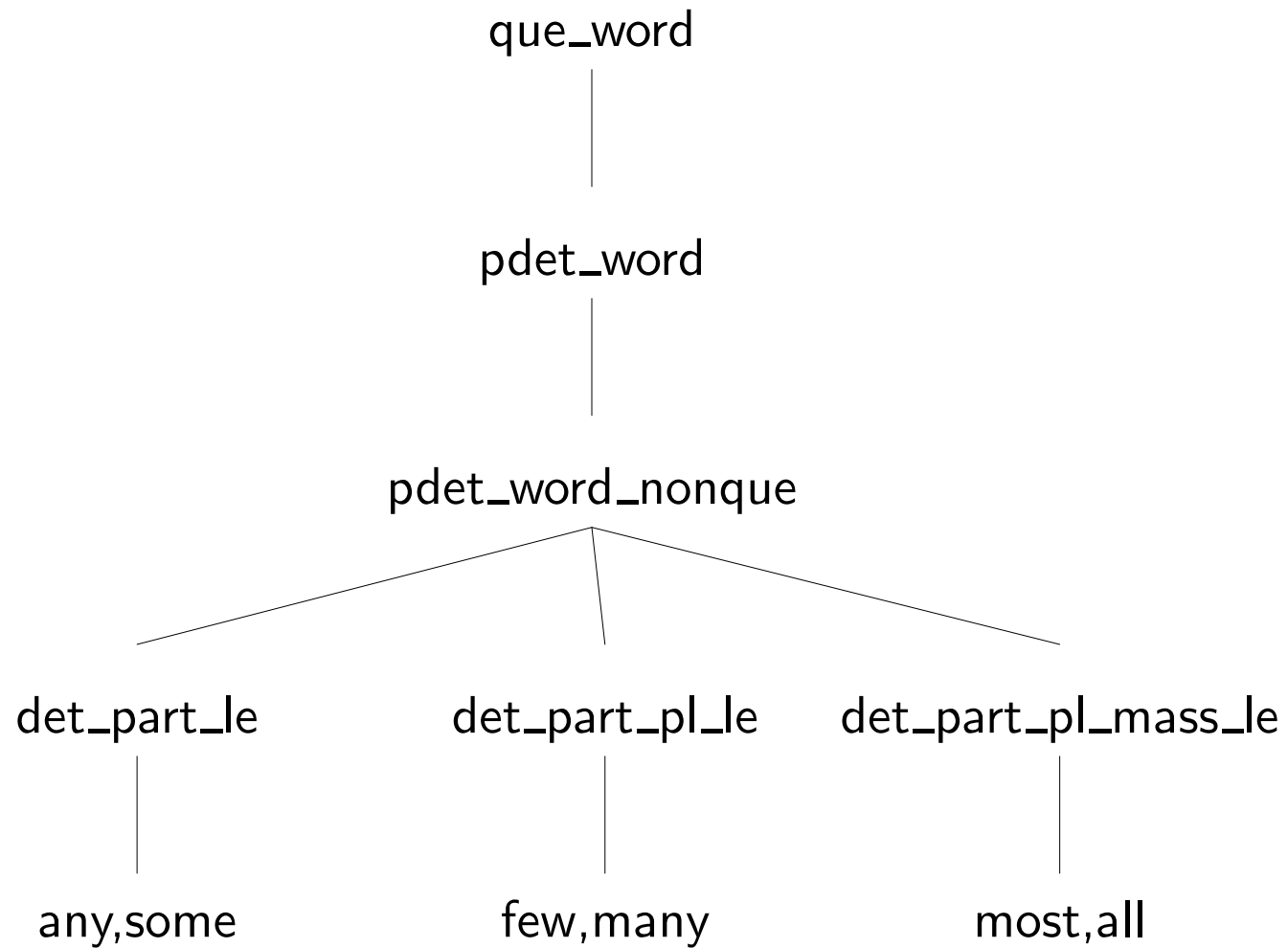
```
(non_derived_word,
  synsem:lex:plus)
*>
(synsem:(local:arg_s:X,
         nonlocal:(slash:Slash,
                   rel:Rel,
                   que:Que)))
goal
  (collect_slashes(X,Slash),
   collect_rel(X,Rel),
   collect_que(X,Que)).
collect_slashes([],[]) if true.
collect_slashes([(nonlocal:slash:X)|Rest_y], All_1) if
  collect_slashes(Rest_y,Rest_1),
  append(X,Rest_1,All_1).
```

Partitive nouns: Macros



```
t(1,"all the dogs sleep",_,1,n_part_npcomp_le).
t(2,"all the dog sleeps",_,1,n_part_npcomp_le).
t(3,"all the dogs sleeps",_,0,"*").
t(4,"half of the dogs sleep",_,1,n_part_ppof_agr_le).
t(5,"half of the dog sleeps",_,1,n_part_ppof_agr_le).
t(6,"half of the dogs sleeps",_,0,"*").
t(7,"half the dogs sleep",_,1,n_part_npcomp_le).
t(8,"half the dog sleeps",_,1,n_part_npcomp_le).
t(9,"half the dogs sleeps",_,0,"*").
t(10,"half sleep",_,2,nocomp_or_opt_npcomp).
t(11,"half sleeps",_,2,nocomp_or_opt_npcomp).
t(12,"none of the dogs sleep",_,1,n_part_ppof_noagr_le).
t(13,"none of the dogs sleeps",_,1,n_part_ppof_noagr_le).
t(14,"none sleep",_,1,n_part_nocomp_le).
t(15,"none sleeps",_,1,n_part_nocomp_le).
t(16,"all dog",_,0,_).
```

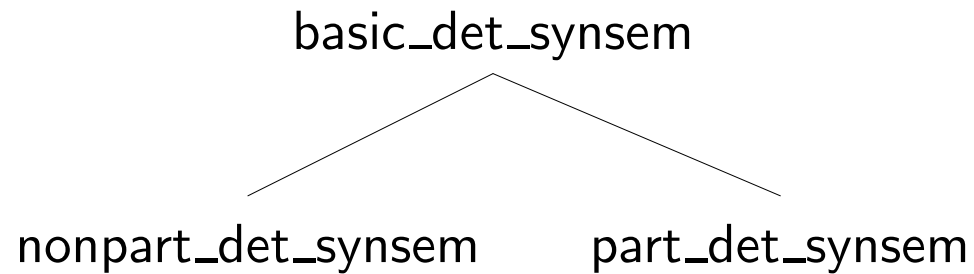
Macros for partitive determiners



```
t(17,"any dog",_,1,det_part_le).
t(18,"any dogs",_,1,det_part_le).
t(19,"any time",_,2,det_part_le).
t(20,"few dog",_,0,det_part_pl_le).
t(21,"few dogs",_,1,det_part_pl_le).
t(22,"few time",_,0,det_part_pl_le).
t(23,"most dog",_,0,det_part_pl_mass_le).
t(24,"most dogs",_,1,det_part_pl_mass_le).
t(25,"most time",_,1,det_part_pl_mass_le).
```

t(26,"many sleep",_,1,part_nocomp_lexrule).
t(27,"many sleeps",_,1,part_nocomp_lexrule).
t(28,"some of the dog sleeps",_,1,part_ppof_agr_lexrule).
t(29,"some of the dog sleep",_,0,part_ppof_agr_lexrule).
t(30,"some of the dogs sleeps",_,0,part_ppof_agr_lexrule).
t(31,"some of the dogs sleep",_,1,part_ppof_agr_lexrule).
t(33,"many of the dogs sleeps",_,0,part_ppof_noagr_lexrule).
t(34,"many of the dogs sleep",_,1,part_ppof_noagr_lexrule).
t(35,"many of the dog sleeps",_,0,part_ppof_noagr_lexrule).
t(36,"many of the dog sleep",_,1,part_ppof_noagr_lexrule).

Partitive determiner synsem values



```
nonpart_det_synsem macro (@basic_det_synsem,  
  local:cat:head:nonpart_det).
```

```
part_det_synsem macro (@basic_det_synsem,  
  local:cat:head:part_det).
```

The Optionality Data

Head-Complement Structures:

- (26) a. Kim hired Tom.
b. Kim showed Tom the office.
c. Kim will sleep.
d. Kim ought to hire Tom.
e. Kim has been hired by Tom.

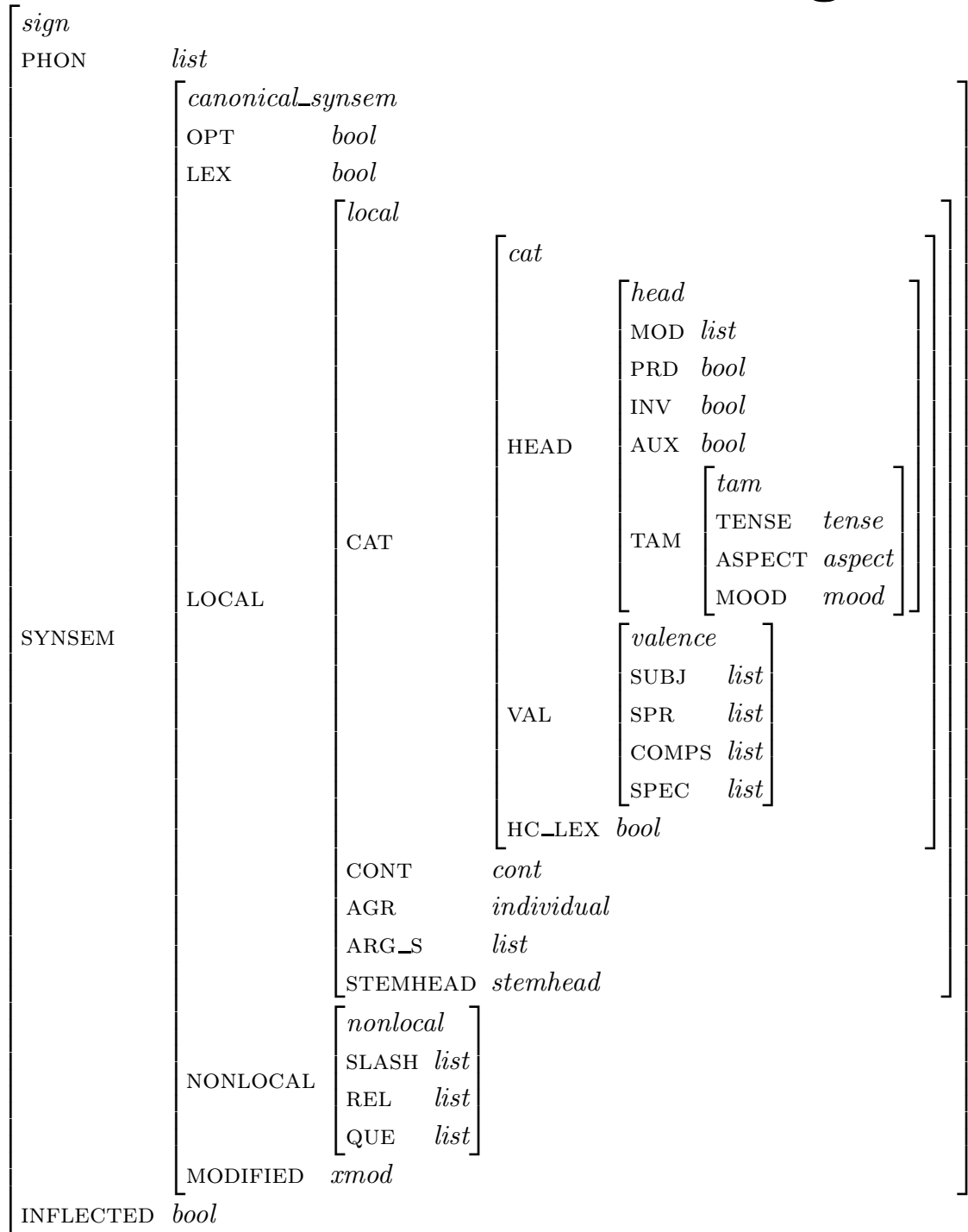
Optional Complements:

- (27)
- a. Kim bet Tom five dollars that they hired Cindy.
 - b. Kim bet Tom five dollars.
 - c. Kim bet Tom that they hired Cindy.
 - d. Kim bet five dollars that they hired Cindy.
 - e. Kim bet five dollars.
 - f. Kim bet that they hired Cindy.
 - g. Kim bet Tom.
 - h. Kim bet.

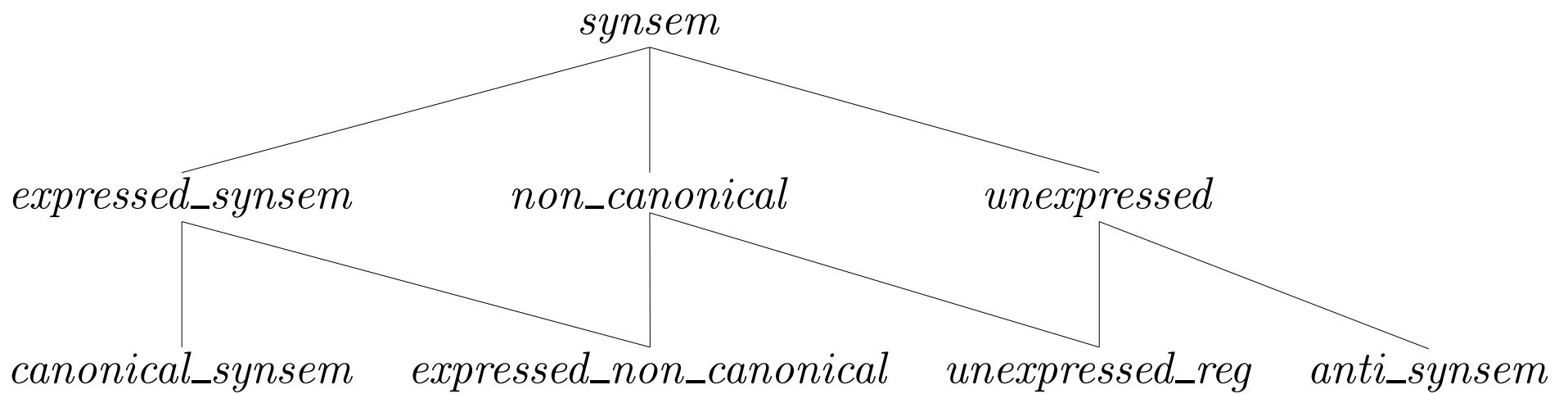
Head-Marker Structures:

- (28)
- a. Kim says that the dog sleeps in the office.
 - b. Kim says that in the office the dog sleeps.

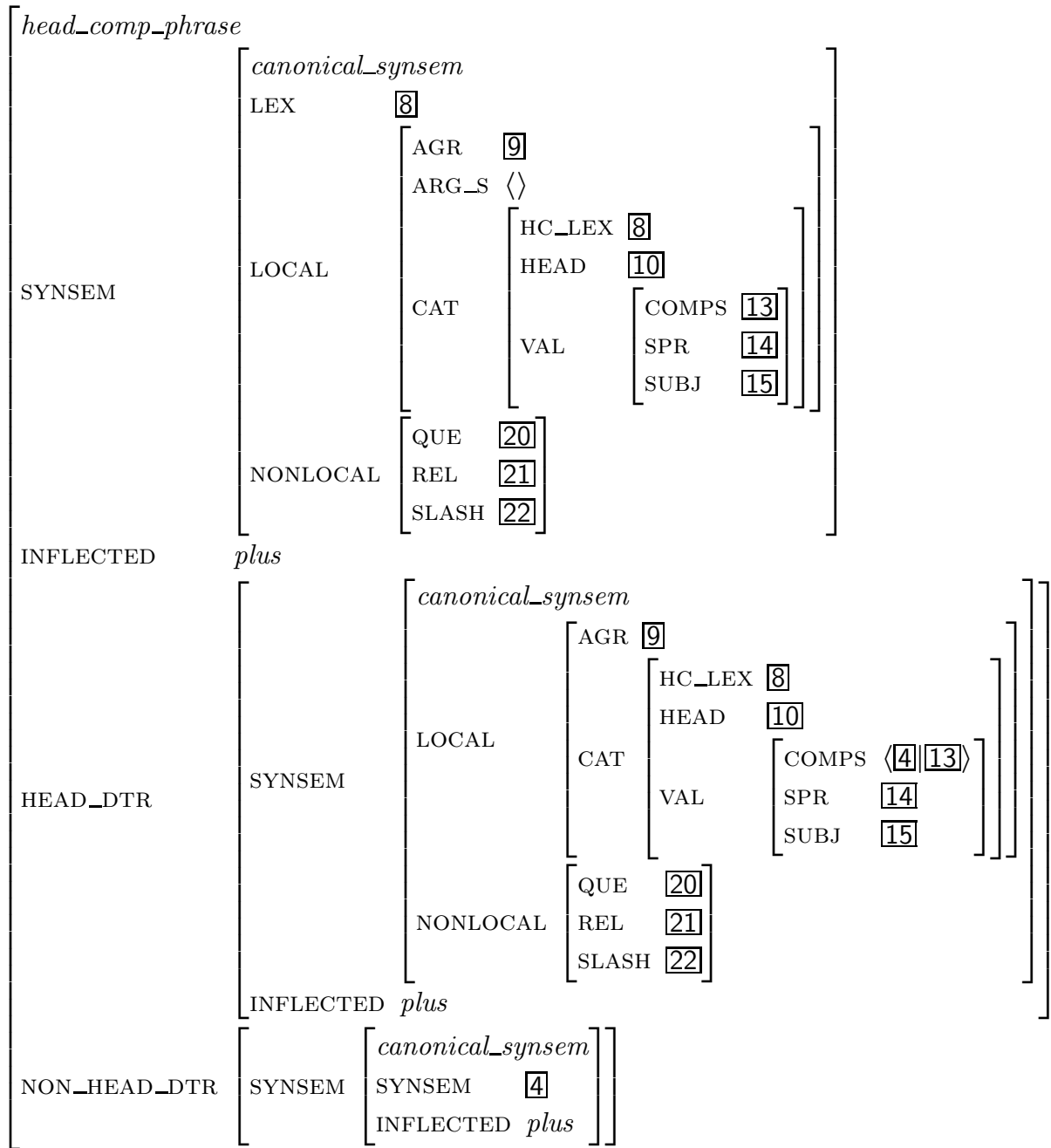
The basic architecture of signs



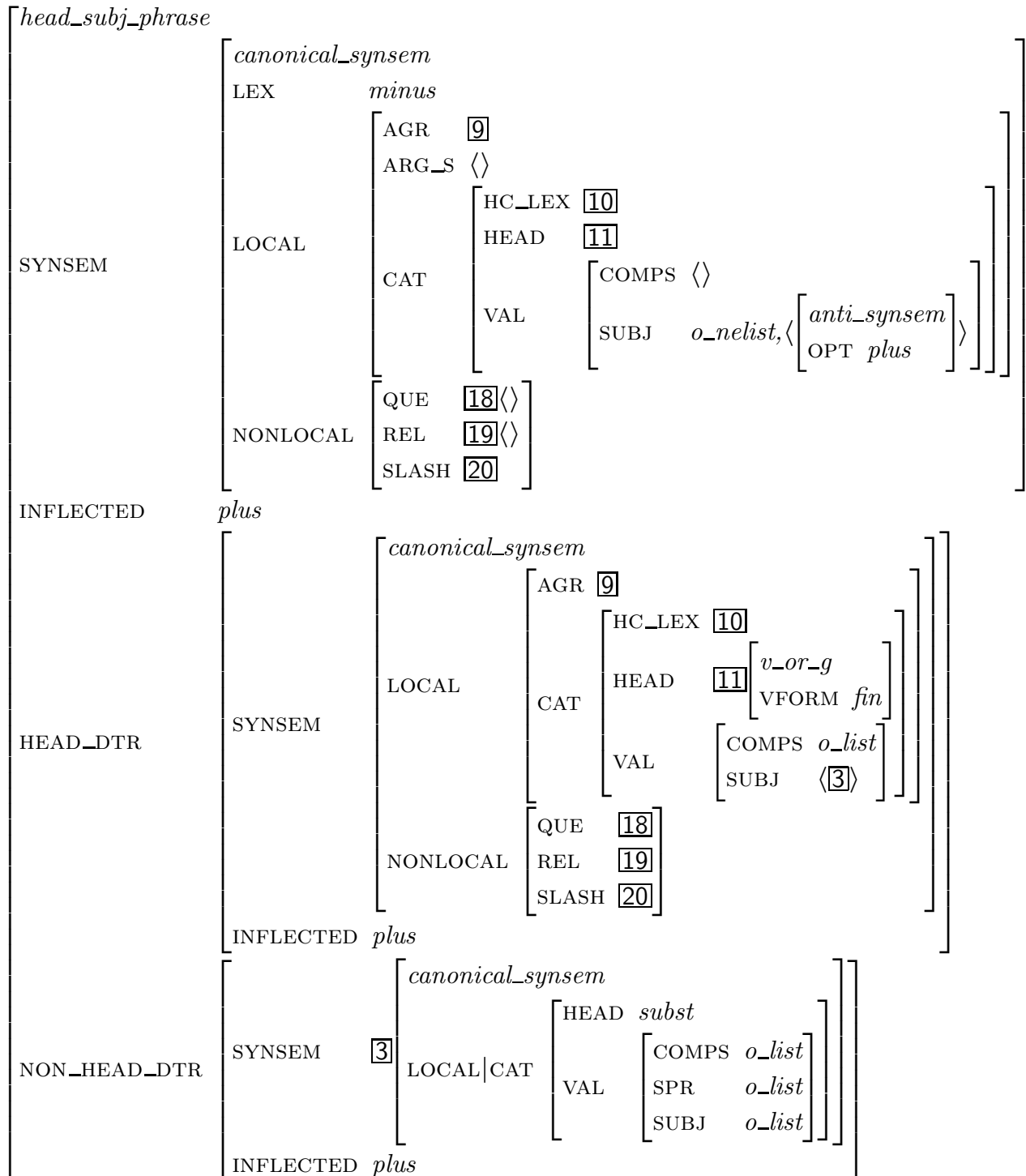
The subtypes of *synsem*



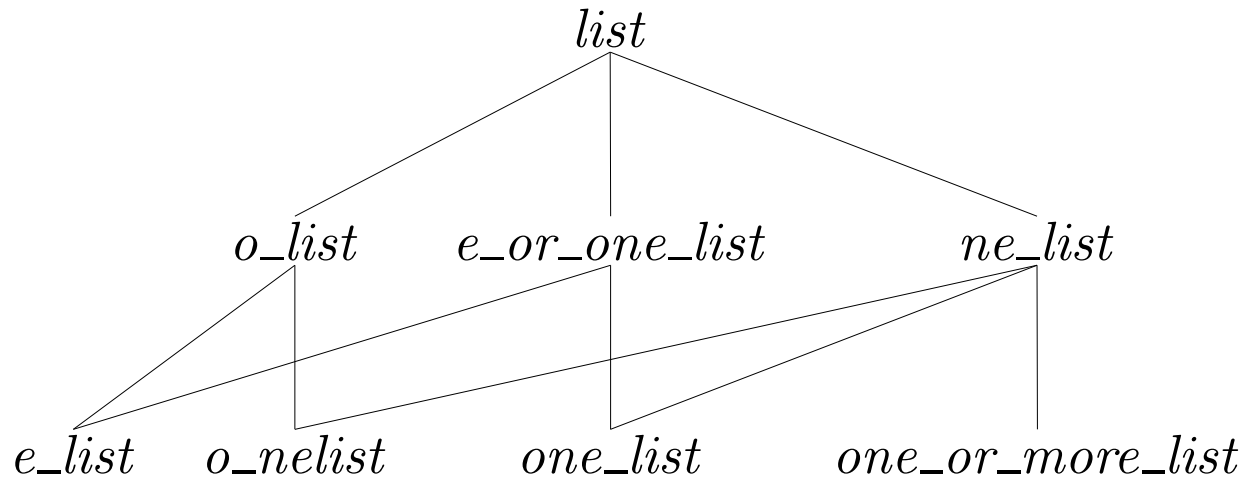
The head-complement rule



The head-subject rule

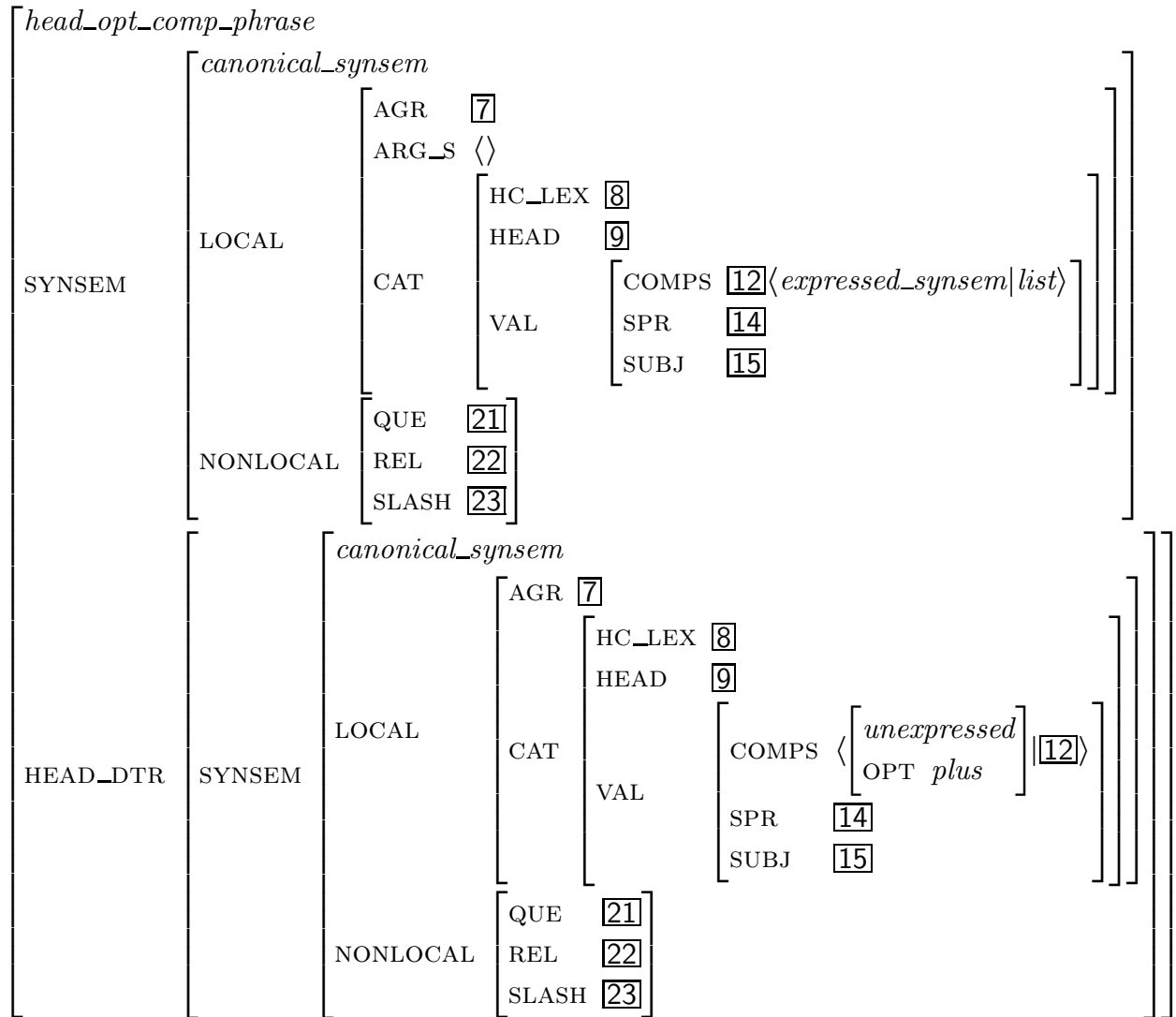


Saturated valence lists



$$o_nelist \rightarrow \left[\begin{array}{l} \text{HD} \left[\begin{array}{l} \textit{unexpressed} \\ \text{OPT plus} \end{array} \right] \\ \text{TL } o_list \end{array} \right]$$

The head-opt-comp rule



The head-opt-two-comp rule

