Grammar Engineering for Deep Linguistic Processing SS2010

Lecture 5: Minimal Recursion Semantics (MRS)

Dan Flickinger, Yi Zhang, Valia Kordoni

Department of Computational Linguistics & Phonetics
Saarland University

Language Technology Lab
German Research Center for Artificial Intelligence

May 2010

Outline

Logical form

- For each sentence admitted by the grammar, we want to produce a meaning representation suitable for applying rules of inference.

  `This fierce dog chased that angry cat.`

- $this(x) \wedge fierce(x) \wedge dog(x) \wedge chased(e,x,y) \wedge that(y) \wedge angry(y) \wedge cat(y)$

Desiderata

- **Compositionality**
  The meaning of a phrase is composed of the meanings of its parts.
- **Existing machinery**
  Unification is the only mechanism we use for constructing semantics in the grammar.

Semantics in feature structures

Semantic content in the CONT attribute of every word and phrase

$$\begin{bmatrix} \textit{sign} \\ \text{SYNSEM} \begin{bmatrix} \text{LOCAL} \begin{bmatrix} \text{CAT} \begin{bmatrix} \text{HEAD} \begin{bmatrix} \textit{head} \\ \begin{bmatrix} \text{MOD} & \langle \ldots \rangle \end{bmatrix} \end{bmatrix} \\ \text{VAL} \begin{bmatrix} \text{SUBJ} & \langle \ldots \rangle \\ \text{SPR} & \langle \ldots \rangle \\ \text{COMPS} & \langle \ldots \rangle \end{bmatrix} \end{bmatrix} \\ \textbf{CONT} \begin{bmatrix} \text{HOOK} & [\,] \\ \text{RELS} & \langle \ldots \rangle \\ \text{HCONS} & \langle \ldots \rangle \end{bmatrix} \end{bmatrix} \\ \text{NON-LOCAL} \begin{bmatrix} \text{SLASH} & \langle \ldots \rangle \end{bmatrix} \end{bmatrix} \\ \text{ARGS} \quad \langle \quad \rangle \end{bmatrix}$$

Semantics in feature structures

- We use Minimal Recursion Semantics [Copestake et al., 2005]
- The value of `CONT` for a sentence is essentially a list of relations in the attribute `RELS`, with the arguments in those relations appropriately linked:
- Semantic relations are introduced by lexical entries, and are appended when words are combined with other words or phrases.

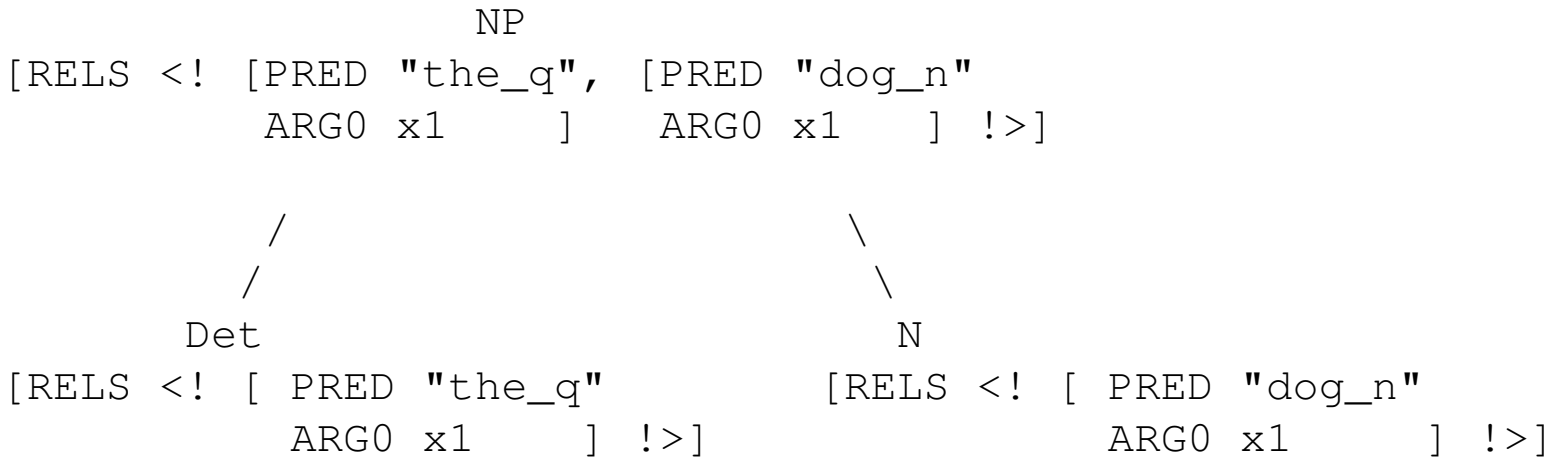
A simple example

The dog barks.

```
[RELS <! [ PRED "the_q" , [ PRED "dog_n" , [ PRED "bark_v"
           ARG0 x1 ]        ARG0 x1 ]        ARG0 e1
                                             ARG1 x1 ] !> ]
```

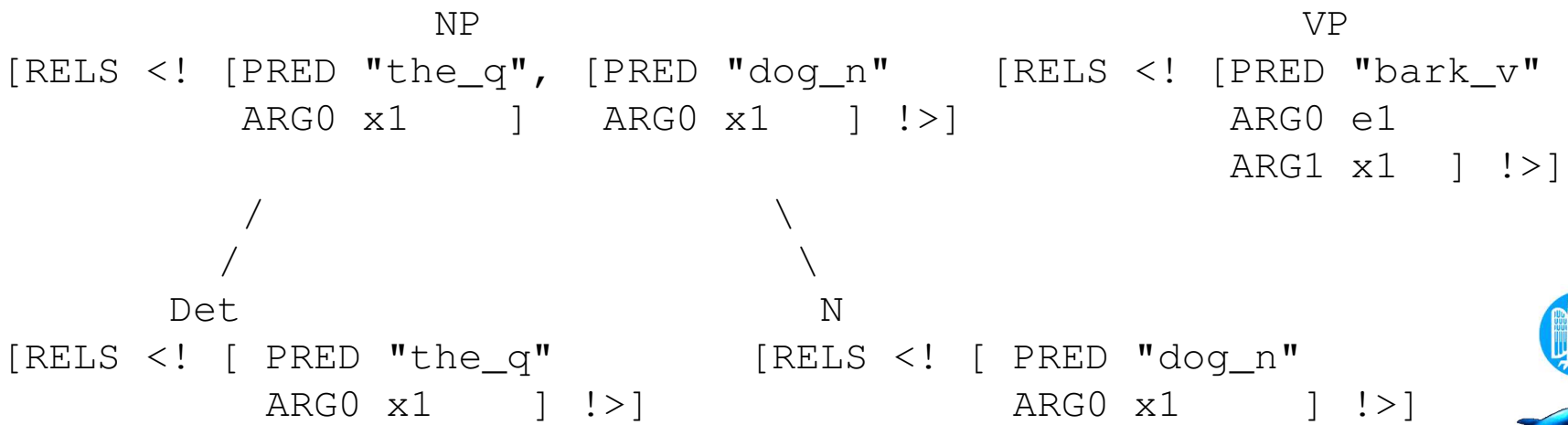
Semantic Composition

The dog barks.



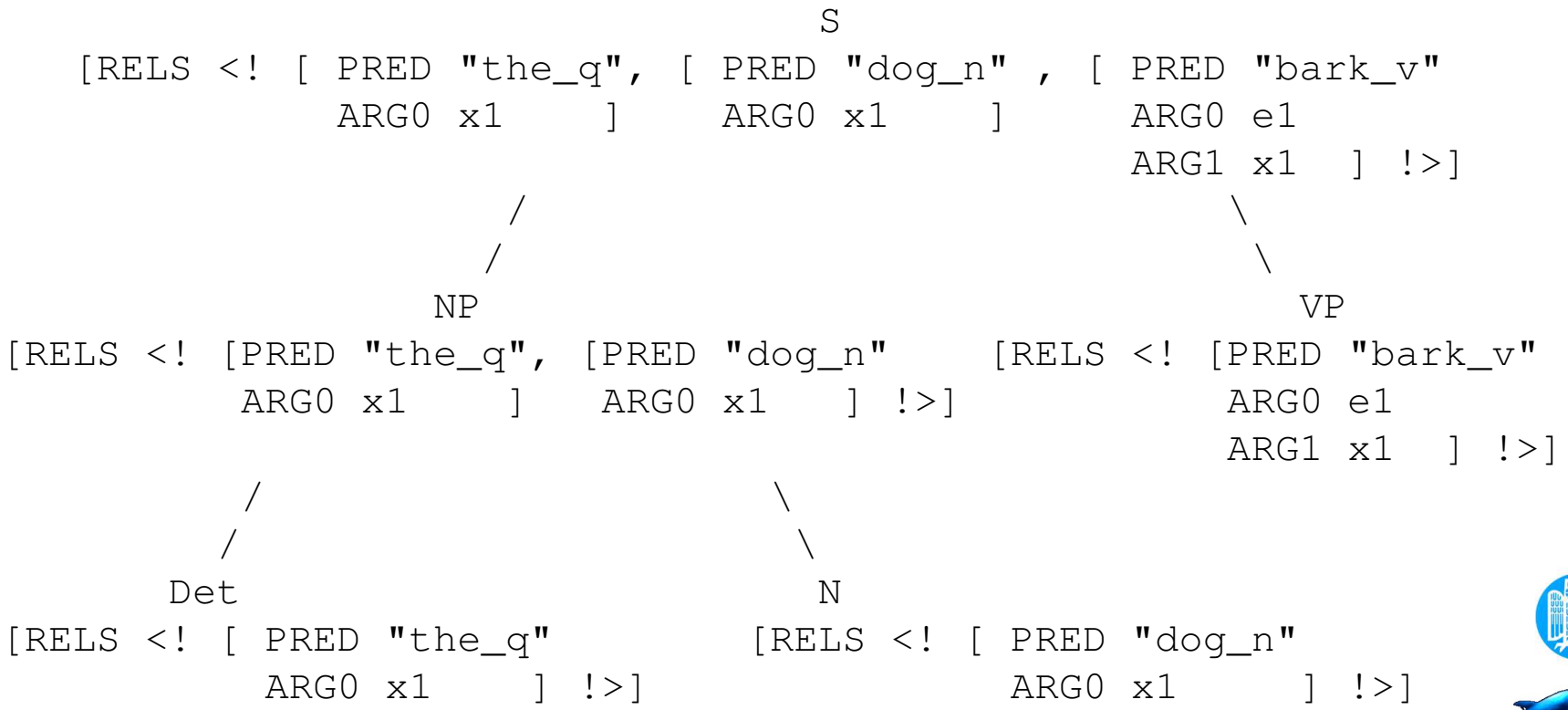
Semantic Composition

The dog barks.



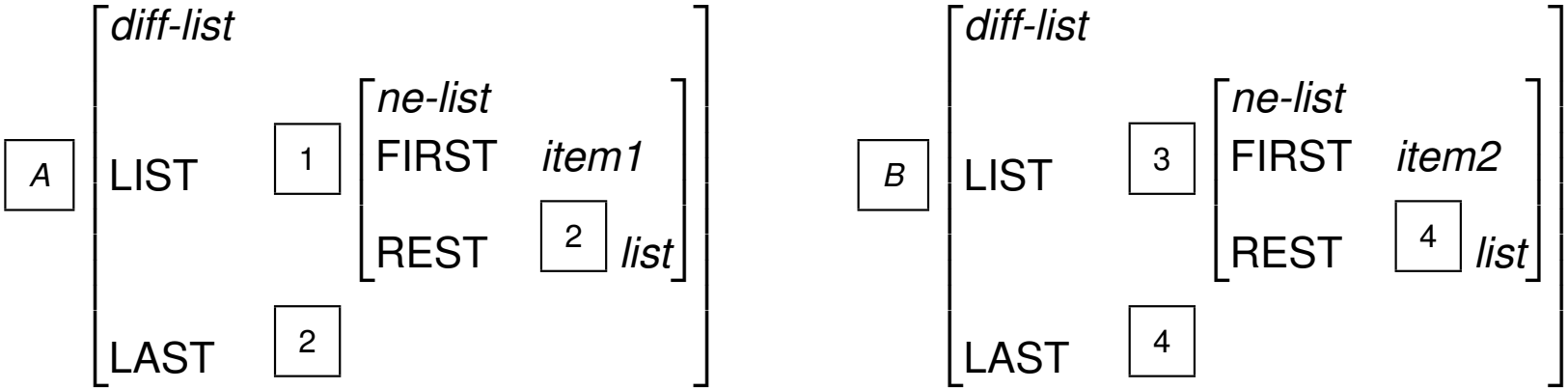
Semantic Composition

The dog barks.



Appending lists with unification

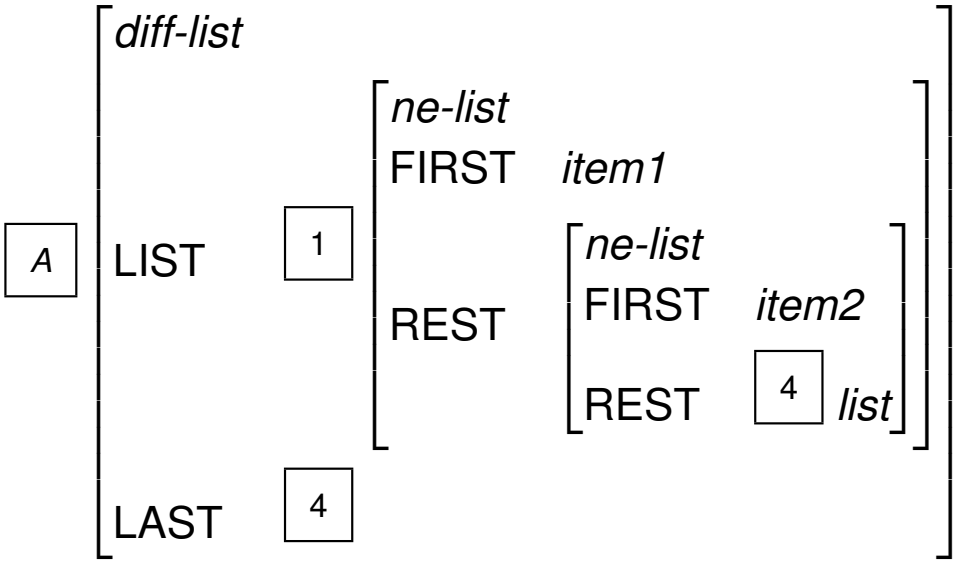
A *difference list* embeds an open-ended list into a container structure that provides a 'pointer' to the end of the ordinary list.



Using the `LAST` pointer of difference list `A` we can append `A` and `B` by

- (i) unifying the front of `B` (i.e. the value of its `LIST` feature) into the tail of `A` (its `LAST` value) and
- (ii) using the tail of difference list `B` as the new tail for the result of the concatenation.

Result of appending lists



Linking semantic arguments

- Each word or phrase also has an `INDEX` attribute in `CONT.HOOK`
- When heads select a complement or specifier, they constrain its `INDEX` value – an instance variable for nouns, an event variable for verbs.
- Each lexeme also specifies a `KEY` relation (to allow complex semantics)

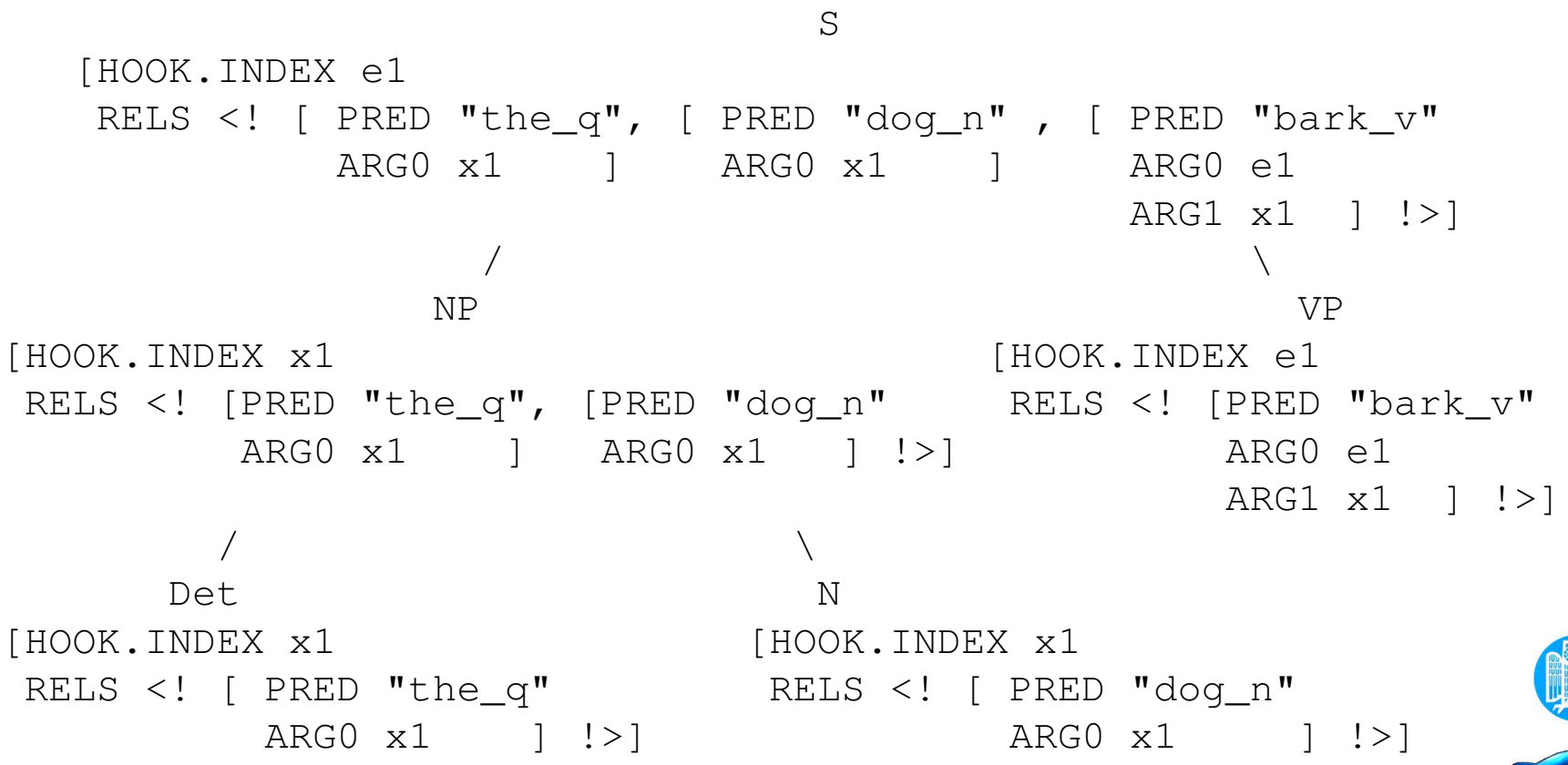
Linking semantic arguments

$$\left[\begin{array}{ll} \textit{synsem} & \\ \text{LOCAL} & \left[\begin{array}{ll} \text{CAT} & \left[\begin{array}{ll} \text{VAL} & \left[\begin{array}{ll} \text{SUBJ} & \langle \left[\text{INDEX } \boxed{1}\right] \rangle \\ \text{COMPS} & \langle \left[\text{INDEX } \boxed{2}\right] \rangle \end{array}\right] \end{array}\right] \\ \text{CONT} & \left[\begin{array}{ll} \text{HOOK} & \left[\text{INDEX } \boxed{3}\right] \\ \text{RELS} & \langle \left[\begin{array}{ll} \text{PRED } \textit{string} & \\ \text{ARG0} & \boxed{3} \\ \text{ARG1} & \boxed{1} \\ \text{ARG2} & \boxed{2} \end{array}\right] \rangle \\ \text{HCONS} & \langle \ldots \rangle \end{array}\right] \end{array}\right] \\ \text{NON-LOCAL} & \left[\text{SLASH } \langle \ldots \rangle\right] \end{array}\right]$$

Semantics of phrases

- Every phrase makes the value of its own RELS attribute be the result of appending the RELS lists of its daughter(s), using unification of difference lists.
- Every phrase identifies its semantic INDEX value with the INDEX value of one of its daughters (the semantic head).
- Since we unify the *synsem* of a complement or specifier with the constraints in the head-daughter, unification also takes care of semantic linking.
- Head-modifier structures work analogously – the modifier lexically constrains the semantic index of the head-daughter it will modify, and then the rules unify the *synsem* of the head-daughter with the MOD's value in the modifier.

Our simple example again



References I

Copestake, A., Flickinger, D., Pollard, C. J., and Sag, I. A. (2005).
Minimal recursion semantics: an introduction.
Research on Language and Computation, 3(4):281–332.