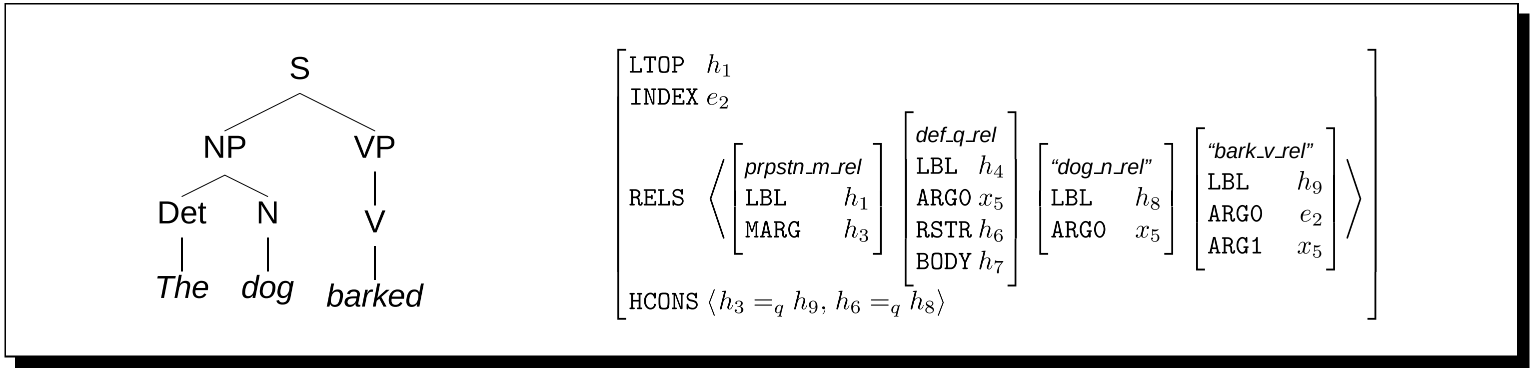

Computational Linguistics (INF2820 — Semantics)

$\{$*this(x)* $\wedge$ *fierce(x)* $\wedge$ *dog(x)* $\wedge$ *bark(e,x)* $\}$

Stephan Oepen

Universitetet i Oslo

`oe@ifi.uio.no`

Adding Semantics to Unification Grammars

- **Logical Form**

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

  This fierce dog chased that angry cat.

  $$this(x) \wedge fierce(x) \wedge dog(x) \wedge chase(e,x,y) \wedge past(e) \wedge that(y) \wedge angry(y) \wedge cat(y)$$

- **Compositionality**

  The meaning of each phrase is composed of the meanings of its parts.

- **Existing Machinery**

  Unification is the only means for constructing semantics in the grammar.

(Elementary) Semantics in Typed Feature Structures

- Semantic content in the SEM attribute of every word and phrase

$$\mathit{expression}\begin{bmatrix} \mathrm{HEAD} & \mathit{pos} \\ \mathrm{SPR} & \mathit{*list*} \\ \mathrm{COMPS} & \mathit{*list*} \\ \mathrm{SEM} & \mathit{semantics}\begin{bmatrix} \mathrm{RELS} & \mathit{*dlist*} \end{bmatrix} \end{bmatrix}$$

- The value of SEM for a sentence is simply a list of relations in the attribute RELS, with the arguments in those relations 'linked up' appropriately:

$$\begin{bmatrix} \mathrm{RELS} & \left\langle \begin{bmatrix} \mathrm{PRED} & \mathit{"the\_rel"} \\ \mathrm{ARG0} & \boxed{1}\,\mathit{entity} \end{bmatrix}, \begin{bmatrix} \mathrm{PRED} & \mathit{"dog\_rel"} \\ \mathrm{ARG0} & \boxed{1} \end{bmatrix}, \begin{bmatrix} \mathrm{PRED} & \mathit{"bark\_rel"} \\ \mathrm{ARG0} & \mathit{event} \\ \mathrm{ARG1} & \boxed{1} \end{bmatrix} \right\rangle \end{bmatrix}$$

- Semantic relations are introduced by lexical entries, and are appended when grammar rules combine words with other words or phrases.

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 at the top level:

$$\boxed{A}\ {}_{\textit{*dlist*}}\begin{bmatrix} \texttt{LIST}\ \boxed{1}\ {}_{\textit{*ne-list*}}\begin{bmatrix} \texttt{FIRST}\ X \\ \texttt{REST}\ \ \boxed{2}\ \textit{*list*} \end{bmatrix} \\ \texttt{LAST}\ \boxed{2} \end{bmatrix} \qquad \boxed{B}\ {}_{\textit{*dlist*}}\begin{bmatrix} \texttt{LIST}\ \boxed{3}\ {}_{\textit{*ne-list*}}\begin{bmatrix} \texttt{FIRST}\ Y \\ \texttt{REST}\ \ \boxed{4}\ \textit{*list*} \end{bmatrix} \\ \texttt{LAST}\ \boxed{4} \end{bmatrix}$$

- Using the LAST pointer of difference list $\boxed{A}$ we can append $\boxed{A}$ and $\boxed{B}$ by

  (i) unifying the front of $\boxed{B}$ (i.e. the value of its LIST feature) into the tail of $\boxed{A}$ (i.e. the value of its LAST feature); and

  (ii) using the tail of $\boxed{B}$ as the new tail for the result of the concatenation.

An Example: Concatenation of Orthography

$$\left[\texttt{ORTH}\left[\begin{array}{ll}\texttt{LIST} & \boxed{1}\\ \texttt{LAST} & \boxed{3}\end{array}\right]\right] \longrightarrow \left[\texttt{ORTH}\left[\begin{array}{ll}\texttt{LIST} & \boxed{1}\\ \texttt{LAST} & \boxed{2}\end{array}\right]\right], \left[\texttt{ORTH}\left[\begin{array}{ll}\texttt{LIST} & \boxed{2}\\ \texttt{LAST} & \boxed{3}\end{array}\right]\right]$$

List Concatenation in TDL

$$\left[\text{ORTH}\left[\begin{array}{l}\text{LIST } \boxed{1}\\ \text{LAST } \boxed{3}\end{array}\right]\right] \longrightarrow \left[\text{ORTH}\left[\begin{array}{l}\text{LIST } \boxed{1}\\ \text{LAST } \boxed{2}\end{array}\right]\right], \left[\text{ORTH}\left[\begin{array}{l}\text{LIST } \boxed{2}\\ \text{LAST } \boxed{3}\end{array}\right]\right]$$

```
binary-rule := phrase &
[ ORTH [ LIST #front, LAST #tail ],
  ARGS < [ ORTH [ LIST #front, LAST #middle ] ],
         [ ORTH [ LIST #middle, LAST #tail ] ] > ].

binary-head-initial := head-initial & binary-rule.

binary-head-final := head-final & binary-rule.
```

Linking Semantic Arguments

- Each word or phrase carries an associated variable: its `INDEX` in `SEM`;
- When heads select a complement or specifier, they constrain its `INDEX` value: an *entity* variable for nouns, an *event* variable for verbs;
- Each lexeme also specifies a `KEY` relation (to allow complex semantics).

```
[ HEAD          verb
  SPR.FIRST     [ SEM.INDEX [1] ]
  COMPS.FIRST   [ SEM.INDEX [2] ]
  SEM           [ INDEX [0] event
                  KEY   [3] [ PRED *string*
                              ARG0 [0]
                              ARG1 [1]
                              ARG2 [2] ]
                  RELS  <! [3] !> ]
]
transitive-verb-lxm
```

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) (difference list concatenation);
- Every phrase identifies its semantic `INDEX` value with the `INDEX` value of exactly *one* of its daughters (which we will call the *semantic head*);
- As we unify the whole TFS of a complement or specifier with the constraints in the syntactic head, unification takes care of semantic linking.
- Head – modifier structures are analogous: the modifier lexically constrains the `INDEX` of the head daughter it will modify; the rules unify the whole TFS of the head daughter with the `MOD` value in the modifier.

A Linking Example Involving Modification