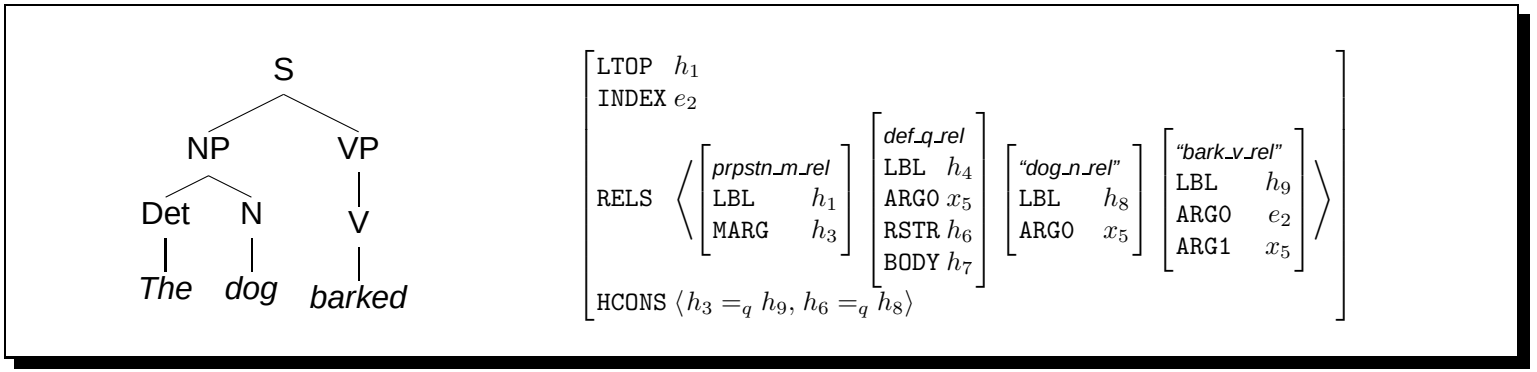

Computational Linguistics (INF2820 — Semantics)

$\{ \textit{the(i)} \wedge \textit{fierce(i)} \wedge \textit{dog(i)} \wedge \textit{bark(s, i)} \}$

Stephan Oepen

Universitetet i Oslo

oe@ifi.uio.no

Some Areas of Descriptive Grammar

Phonetics	***The study of speech signals.***
Phonology	***The study of sound systems.***
Morphology	***The study of word structure.***
Syntax	***The study of sentence structure.***
Semantics	***The study of language meaning.***
Pragmatics	***The study of language use.***

A Tool Towards Understanding: (Formal) Grammar

Wellformedness

- *Kim was happy because ____ passed the exam.*
- *Kim was happy because ____ final grade was an A.*
- *Kim was happy when she saw ____ on television.*

A Tool Towards Understanding: (Formal) Grammar

Wellformedness

- *Kim was happy because ____ passed the exam.*
- *Kim was happy because ____ final grade was an A.*
- *Kim was happy when she saw ____ on television.*

Meaning

- *Kim gave Sandy the book.*
- *Kim gave the book to Sandy.*
- *Sandy was given the book by Kim.*

A Tool Towards Understanding: (Formal) Grammar

Wellformedness

- *Kim was happy because ____ passed the exam.*
- *Kim was happy because ____ final grade was an A.*
- *Kim was happy when she saw ____ on television.*

Meaning

- *Kim gave Sandy the book.*
- *Kim gave the book to Sandy.*
- *Sandy was given the book by Kim.*

Ambiguity

- *Kim saw the astronomer with the telescope.*
- *Have her report on my desk by Friday!*

Thinking Aloud: Candidate Meaning Representations

The dog barked.

The fierce dog barked.

The fierce dog chased that cat.

The fierce dog chased that angry black cat.

The fierce dog chased that angry black cat in the park.

The fierce dog barked loudly.

The dog is fierce and barked.

The dog that barked is fierce.

Thinking Aloud: Candidate Meaning Representations

Semantic *propositions* can be true or false;

(elementary) *predications* as basic building blocks;

(near) *paraphrases* should have equivalent semantics.

The fierce dog chased that angry black cat.

The fierce dog chased that angry black cat in the park.

The fierce dog barked loudly.

The dog is fierce and barked.

The dog that barked is fierce.

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.

  The fierce dog chased that angry cat.

  $$\textit{the}(i) \wedge \textit{fierce}(i) \wedge \textit{dog}(i) \wedge \textit{chase}(s, i, j) \wedge \textit{past}(s) \wedge \textit{that}(j) \wedge \textit{angry}(j) \wedge \textit{cat}(j)$$

- **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 resides in the SEM attribute of every word and phrase:

$$\textit{expression}\begin{bmatrix}\texttt{HEAD} & \textit{pos}\\ \texttt{SPR} & \textit{*list*}\\ \texttt{COMPS} & \textit{*list*}\\ \texttt{SEM} & \textit{semantics}\begin{bmatrix}\texttt{RSTR}\ \textit{*dlist*}\end{bmatrix}\end{bmatrix}$$

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

$$\begin{bmatrix}\texttt{RSTR}\ \left\langle\begin{bmatrix}\texttt{RELN}\ \textit{"the\_rel"}\\ \texttt{ARG0}\ \boxed{1}\textit{entity}\end{bmatrix},\begin{bmatrix}\texttt{RELN}\ \textit{"dog\_rel"}\\ \texttt{ARG0}\ \boxed{1}\end{bmatrix},\begin{bmatrix}\texttt{RELN}\ \textit{"bark\_rel"}\\ \texttt{ARG0}\ \textit{situation}\\ \texttt{ARG1}\ \boxed{1}\end{bmatrix}\right\rangle\end{bmatrix}$$

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

Composition: 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:

$$\boxed{A}\ \textit{*dlist*}\left[\begin{array}{l}\texttt{LIST}\ \boxed{1}\\ \qquad\textit{*ne-list*}\left[\begin{array}{l}\texttt{FIRST}\ X\\ \texttt{REST}\ \ \boxed{2}\ \textit{*list*}\end{array}\right]\\ \texttt{LAST}\ \boxed{2}\end{array}\right] \qquad \boxed{B}\ \textit{*dlist*}\left[\begin{array}{l}\texttt{LIST}\ \boxed{3}\\ \qquad\textit{*ne-list*}\left[\begin{array}{l}\texttt{FIRST}\ Y\\ \texttt{REST}\ \ \boxed{4}\ \textit{*list*}\end{array}\right]\\ \texttt{LAST}\ \boxed{4}\end{array}\right]$$

- 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 (i.e. the value of its LAST feature); and

  (ii) using the tail of B as the new tail for the result of the concatenation.

An Example: Concatenation of Orthography

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

Notational Conventions

- lists not available as built-in data type; abbreviatory notation in TDL:

  `< a, b >` ≡ `[ FIRST a, REST [ FIRST b, REST *null* ] ]`

- underspecified (variable-length) list:

  `< a, ... >` ≡ `[ FIRST a, REST *list* ]`

- difference (open-ended) lists; allow concatenation by unification:

  `<! a !>` ≡ `[ LIST [ FIRST a, REST #tail ], LAST #tail ]`

- built-in and 'non-linguistic' types pre- and suffixed by asterisk (**top**);
- strings (e.g. *"chased"*) need no declaration; always subtypes of **string**;
- strings cannot have subtypes and are (thus) mutually incompatible.

Composition: 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: a variable of type *entity* for nouns, a *situation* variable for verbs;
- Each lexeme also specifies a KEY predication (for complex semantics).

transitive-verb-lxm

$$\begin{bmatrix} \text{HEAD} & \textit{verb} \\ \text{SPR.FIRST} & \begin{bmatrix} \text{SEM.INDEX} & \boxed{1} \end{bmatrix} \\ \text{COMPS.FIRST} & \begin{bmatrix} \text{SEM.INDEX} & \boxed{2} \end{bmatrix} \\ \text{SEM} & \begin{bmatrix} \text{INDEX} & \boxed{0}\ \textit{situation} \\ \text{KEY} & \boxed{3} \begin{bmatrix} \text{RELN} & \textit{*string*} \\ \text{ARG0} & \boxed{0} \\ \text{ARG1} & \boxed{1} \\ \text{ARG2} & \boxed{2} \end{bmatrix} \\ \text{RSTR} & \langle \boxed{3} \rangle \end{bmatrix} \end{bmatrix}$$

Composition: Semantics of Phrases

- Every phrase makes the value of its own RSTR attribute be the result of appending the RSTR lists (of semantic predications) from all its daughter(s)—by virtue of 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