Chart-based Generation Using
Typed Feature Structures

Stephan Oepen
Dan Flickinger

Center for the Study of Language and Information
Stanford University – Spring Quarter 2003

(oe|danf)@csli.stanford.edu

http://lingo.stanford.edu/courses/03/pg
Overview of Generation

- Parsing: start with a string (sequence of tokens) and compute semantics
- Generating: start with semantics and compute strings with that meaning
- The semantics of a sign (a word or phrase) can be represented as a set of relations, each consisting of a predicate and one or more arguments (semantic roles)
- Each sign identifies one of those argument values as its index
- The generator builds all phrases which are consistent with the input semantics and with the rules of the grammar
Naive “Shake ’n Bake” generation

• Lexical entries are retrieved from lexicon by semantic relations of the input
• All combinations of all words and phrases are tried
• All phrases are returned which use up all of the input semantics
• This approach can be expensive
The grammar enriched with semantics

- **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.
  
  \[ \text{this}(x) \land \text{fierce}(x) \land \text{dog}(x) \land \text{chased}(e,x,y) \land \text{that}(y) \land \text{angry}(y) \land \text{cat}(y) \]

- **Compositionality**
  The meaning of a phrase is composed of the meanings of its parts.

- **Existing machinery**
  Unification is the only mechanism used for constructing semantics in the grammar.
Semantics in feature structures

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

```
| HEAD   | pos |
| SPR    | *list* |
| COMPS  | *list* |
| SEM    | cont[RELS *dlist*] |
```

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

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

- Semantic relations are introduced by lexical entries, and are appended when words are combined with other words or phrases to form larger phrases.
A simple example

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

/          \
/          \

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

/          \
/          \\n
Det          N
[RELS <! [ PRED "the_q" ] [RELS <! [ PRED "dog_n"
ARG0 x1 ] !> ]
ARG0 x1 ] !> ]
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.

\[
\begin{align*}
\text{A} & \quad \begin{bmatrix}
\text{LIST} & 1 \\
\text{*ne-list*} & \text{FIRST } "\text{item-1}" \\
\text{REST} & 2 \\
\text{*list*} &
\end{bmatrix} \\
\text{*dlist*} & \quad \begin{bmatrix}
\text{LAST} & 2 \\
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\text{B} & \quad \begin{bmatrix}
\text{LIST} & 3 \\
\text{*ne-list*} & \text{FIRST } "\text{item-2}" \\
\text{REST} & 4 \\
\text{*list*} &
\end{bmatrix} \\
\text{*dlist*} & \quad \begin{bmatrix}
\text{LAST} & 4 \\
\end{bmatrix}
\end{align*}
\]

- 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.
Linking semantic arguments

- Each word and phrase also has an INDEX attribute in SEM
- When heads select a complement or specifier, they also constrain its INDEX value – an instance variable for nouns, or an event variable for verbs.

```
trans-verb-lxm
\[\begin{array}{|c|}
\hline
\text{HEAD} & \text{verb} \\
\text{SPR.FIRST} & \text{syn-struc} \left[ \text{SEM.INDEX} \ [1] \right] \\
\text{COMPS.FIRST} & \text{syn-struc} \left[ \text{SEM.INDEX} \ [2] \right] \\
\text{SEM} & \left[ \begin{array}{c}
\text{INDEX} \ [0] \text{ event} \\
\text{REL.S} \ < ! \end{array} \right] \\
\text{cont} & \left[ \begin{array}{c}
\text{PRED} \ string \\
\text{ARG0} \ [0] \\
\text{ARG1} \ [1] \\
\text{ARG2} \ [2] \\
\end{array} \right] \\
\text{arg12-v-rel} & \left[ \begin{array}{c}
\end{array} \right] \\
\hline
\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 whole sign 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 constrains the semantic index of the head daughter it will modify, and then the rules unify the whole sign of the head daughter with the MOD’s value in the modifier.
Our simple example again
Using a modified chart for generation

• The chart can be organized by semantic index values rather than by string position
• Each passive edge introduces one of the indices found in the input semantics
• Each active edge is looking for a passive edge with the right index
• A successful result is a passive edge that “uses up” all of the input semantics
Chart-based generation

Given an input semantics consisting of a bag of relations R-1 ... R-n

• for each R-i, retrieve from the lexicon all lexical entries whose RELS list contains R-i
• initialize the agenda with passive edges for these lexical entries anchored by their INDEX values
• initialize the chart with one cell for each unique index in the input semantics, and add active edges for the grammar rules with each anchored by the INDEX of the next daughter
• for each edge in the agenda, apply a variant of the fundamental rule based on matching indices (adding the popped edge to the chart and putting the newly constructed edges on the agenda)
• stop generating when the agenda is empty
• return as successful results all edges whose semantics subsumes (covers) the input semantics, and whose feature structure unifies with the root conditions
Background reading
