Computational Linguistics: Syntax-Semantics

Raffaella Bernardi

University of Trento

Contents

1. The Three Tasks Revised

Task 1 Specify a reasonable **syntax** for the natural language fragment of interest. **We can do this using CFG**.

Task 2 Specify semantic representations for the **lexical items**. **We know what this involves**

Task 3 Specify the **translation** of an item $\mathcal{R}$ whose parts are $\mathcal{F}$ and $\mathcal{A}$ with the help of functional application. That is, we need to specify which part is to be thought of as functor (here it's $\mathcal{F}$), which as argument (here it's $\mathcal{A}$) and then let the resultant translation $\mathcal{R}'$ be $\mathcal{F}'(\mathcal{A}')$. **We know that β-conversion (with the help of α-conversion), gives us the tools needed to actually construct the representation built by this process.**

2. Lambda terms and CFG

We will look at the compositional approach to the syntax-semantics interface and build the meaning representation in parallel to the syntactic tree. This reduces to have a **rule-to-rule** system, i.e. each syntactic rule correspond to a semantic rule.

Syntactic Rule 1 $S \rightarrow NP\ VP$

Semantic Rule 1 If the logical form of the NP is α and the logical form of the VP is β then the logical form for the S is $\beta(\alpha)$.

Syntactic Rule 2 $VP \rightarrow TV\ \ NP$

Semantic Rule 2 If the logical form of the TV is α and the logical form of the NP is β then the logical form for the VP is $\alpha(\beta)$.

2.1. Augumenting CFG with terms

That can also be abbreviated as below where γ, α and β are the meaning representations of S, NP and VP, respectively.

$$S(\gamma) \to NP(\alpha)\ VP(\beta) \qquad \gamma = \beta(\alpha)$$

This implies that lexical entries must now include semantic information. For instance, a way of writing this information is as below.

$$TV(\lambda x.\lambda y.wrote(y, x)) \to wrote$$

2.1.1. Exercise (a) Write the semantic rules for the following syntactic rules:

```
s --> np vp
vp --> iv
vp --> tv np
np --> pn
pn --> John | Mia
tv --> knows
iv --> left
```

(b) apply these labeled rules to build the sentences “John knows Mia” and “John left”

3. CG: syntax-semantics interface

Summing up, CG specifies a language by describing the **combinatorial possibilities of its lexical items** directly, without the mediation of phrase-structure rules. Consequently, two grammars in the same system differ only in the lexicon.

The **close relation between the syntax and semantics** comes from the fact that the two syntactic rules are application of a functor category to its argument that corresponds to functional application of the lambda calculus.

We have to make sure that the lexical items are associated with **semantic terms** which correspond to the **syntactic categories**.

3.1. Mapping: types-categories

To set up the form-meaning correspondence, it is useful to build a language of semantic types in parallel to the syntactic type language.

Definition 3.1 (Types) Given a non-empty set of basic types Base, the set of types TYPE is the smallest set such that

i. $\mathsf{Base} \subseteq \mathsf{TYPE}$;
ii. $(a \rightarrow b) \in \mathsf{TYPE}$, if a and $b \in \mathsf{TYPE}$.

Note that this definition closely resembles the one of the syntactic categories of CG. The only difference is the lack of directionality of the functional type $(a \rightarrow b)$. A function mapping the syntactic categories into TYPE can be given as follows.

Definition 3.2 (Categories and Types) *Let us define a function* $\mathtt{type} : \mathit{CAT} \rightarrow \mathit{TYPE}$ *which maps syntactic categories to semantic types.*

$$\mathtt{type}(np) = e; \qquad \mathtt{type}(A/B) = (\mathtt{type}(B) \rightarrow \mathtt{type}(A));$$
$$\mathtt{type}(s) = t; \qquad \mathtt{type}(B\backslash A) = (\mathtt{type}(B) \rightarrow \mathtt{type}(A));$$
$$\mathtt{type}(n) = (e \rightarrow t).$$

3.2. CG: categories and terms

Modus ponens corresponds to functional application.

$$\frac{B/A : t \quad A : r}{B : t(r)}\ (\mathrm{MP_r}) \qquad\qquad \frac{A : r \quad A\backslash B : t}{B : t(r)}\ (\mathrm{MP_l})$$

Example

$$\frac{np : \mathtt{john} \quad np\backslash s : \mathtt{walk}}{s : \mathtt{walk}(\mathtt{john})}\ (\mathrm{MP_l})$$

$$np\backslash s : \lambda x.\mathtt{walk}(x) \qquad (\lambda x.\mathtt{walk}(x))(\mathtt{john}) \leadsto_{\lambda-\mathrm{conv.}} \mathtt{walk}(\mathtt{john})$$

$$\frac{np : \mathtt{john} \quad \dfrac{(np\backslash s)/np : \mathtt{know} \quad np : \mathtt{mary}}{np\backslash s : \mathtt{know}(\mathtt{mary})}\ (\mathrm{MP_r})}{s : \mathtt{know}(\mathtt{mary})(\mathtt{john})}\ (\mathrm{MP_l})$$