Computational Linguistics: syntax-semantics

Raffaella Bernardi

University of Trento

1. Seen to be seen

- Syntax:
  - Formal Grammars can be used to assign a structure to NL strings.
  - Of the Chomsky Hierarchy, we need to use at least CFG.
  - Lexicalized FG have been developed: TAG, HPSG, CG etc.
  - Dependency structures vs. Constituency structure
  - Parsing
- Semantics:
  - NL meaning can be captured by sets. (Denotational Semantics)
  - Sets can be translated into functions.
  - Functions can be represented as λ-term.
  - λ-calculus can be used to compose the MR of NL strings.

Today we will look at the question:

What is the relation between Syntax and Semantics of NL?

2. Syntax-Semantics: Parallel vs. Non-parallel

We could build the meaning representation (MR) of an expression either

(a) in parallel with the construction of its syntactic structure, or

(b) after having built the syntactic analysis.

(a) is the method followed by most formal grammar frameworks as Categorial Grammar (CG), Head-Driven Phrase Structure Grammar (HPSG), Lexical Functional Grammar (LFG), Tree-Adjoining Grammar (TAG).

(b) is used by the Government and Binding Theory and the Minimalist Program (both due to Chomsky).

2.0.1. Advantages The reasons for preferring the first approach are the following:

Psycholinguistic works suggest that human processing proceeds incrementally through the simultaneous application of syntactic, semantics, and phonological constraints to resolve syntactic ambiguity. (Though, note that these systems are models of linguistic competence rather than performance. Hence, these results could not provide direct support of either of the approaches.)

Computational approach requires a way to rule out a semantically ill-formed phrase as soon as it is encountered. Therefore, (a) offers a more efficient architecture for implementing constraint satisfaction. For instance,

1. The delegates met for an hour.
2. The committee met for an hour.
3. *The woman met for an hour.

The use of "met" as intransitive verb requires a subject denoting a plural entity.

3. Computational Semantics

To build a meaning representation we need to fulfill three tasks:

Task 1 Specify a reasonable **syntax** for the natural language fragment of interest.

Task 2 Specify semantic representations for the **lexical items**.

Task 3 Specify the **translation** of constituents **compositionally**. That is, we need to specify the translation of such expressions in terms of the translation of their parts, parts here referring to the substructure given to us by the syntax.

Moreover, when interested in Computational Semantics, all three tasks need to be carried out in a way that leads to computational implementation naturally.

3.1. Compositionality

The question to answer is: "How can we specify in which way the bit and pieces combine?"

1. Meaning (representation) ultimately flows from the lexicon.
2. Meaning (representation) are combined by making use of syntactic information.
3. The meaning of the whole is function of the meaning of its parts, where "parts" refer to substructures given us by the syntax.

4. Montague Universal Grammar

The rule-to-rule and lambda techniques are used in the approach to natural language semantics developed by Richard Montague. In his theory, there are

- **syntactic rules** which show how constituents maybe combined to form other constituents.
- **translation rules** (associated with each such syntax rule) which show how the logical expressions for the constituents have to be joined together to form the logical form of the whole.

For instance, the syntactic and semantics rule for composing and NP with and IV:

S2: If $\delta \in P_{IV}$ and $\alpha \in P_{NP}$, then $F_1(\alpha, \delta) \in P_S$ and $F_1(\alpha, \delta) = \alpha\delta'$, where δ' is the result of replacing the main verb in δ by its third-person singular present form.

T2: If $\delta \in P_{IV}$ and $\alpha \in P_{NP}$ and $\delta | \rightarrow \delta'$ and $\alpha | \rightarrow \alpha'$, then $F_1(\alpha, \delta) | \rightarrow \alpha'(\delta')$.

As grammar, he used Categorial Grammar.

5. 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 $\alpha(\beta)$.

5.1. 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.

$$S(\gamma) \rightarrow NP(\alpha)\ VP(\beta) \quad \gamma = \beta(\alpha)$$

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

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

5.1.1. 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"

6. CG: syntax-semantics interface

In CG **the closed relation between syntax and semantics is formally caputred**: the two syntactic rules correspond to λ-calculus rules: the application of a functor category to its argument corresponds to functional application of the λ-calculus; and hypthetical reasoning corresponds to abstraction of the λ- calculus.

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

6.1. Categorial Grammar (Syntax)

$$A \vdash A$$

$$\frac{\Gamma \vdash A/B \quad \Delta \vdash B}{\Gamma \circ \Delta \vdash A}\ (/\mathrm{E}) \qquad \frac{(\Gamma \circ B) \vdash A}{\Gamma \vdash A/B}\ (/\mathrm{I})$$

$$\frac{\Delta \vdash B \quad \Gamma \vdash B\backslash A}{\Delta \circ \Gamma \vdash A}\ (\backslash\mathrm{E}) \qquad \frac{(B \circ \Gamma) \vdash A}{\Gamma \vdash B\backslash A}\ (\backslash\mathrm{I})$$

Lambda calculus (Semantics)

$$\frac{t \quad u}{t(u)}\ (FA) \qquad \frac{t[x]}{\lambda x.t}\ (Abs)$$

6.2. Formal Correspondence (Syntax-Semantics)

- **Who**: van Benthem (1987), Buszkowski (1987)
- **Aim**: Syntax-Semantic interface
- **How**: Curry-Howard Correspondence between proofs and terms.

$$x : A \vdash x : A$$

$$\frac{\Gamma \vdash t : A/B \quad \Delta \vdash u : B}{\Gamma \circ \Delta \vdash t(u) : A}\ (/\mathrm{E}) \qquad \frac{(\Gamma \circ x : B) \vdash t : A}{\Gamma \vdash \lambda x.t : A/B}\ (/\mathrm{I})$$

$$\frac{\Delta \vdash u : B \quad \Gamma \vdash t : B\backslash A}{\Delta \circ \Gamma \vdash t(u) : A}\ (\backslash\mathrm{E}) \qquad \frac{(x : B \circ \Gamma) \vdash t : A}{\Gamma \vdash \lambda x.t : B\backslash A}\ (\backslash\mathrm{I})$$

6.3. 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 6.1 (Types) Given a non-empty set of basic types Base, the set of types TYPE is the smallest set such that

i. Base $\subseteq$ TYPE;
ii. $(a \rightarrow b) \in$ TYPE, if a and $b \in$ 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 6.2 (Categories and Types) *Let us define a function* $\mathtt{type} : \mathit{CAT} \rightarrow \mathit{TYPE}$ *which maps syntactic categories to semantic types.*

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

6.4. Example: Elimination rules and Function application

Elimination rules/Modus ponens corresponds to functional application.

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

Example

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

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

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

6.5. Examples: Hypothetical reasoning and abstraction

The book which Sara wrote

$$\dfrac{\dfrac{\text{sara} \vdash np : \texttt{sara} \qquad \dfrac{\text{wrote} \vdash (np\backslash s)/np : \texttt{wrote} \quad [z \vdash np : z]^1}{\text{wrote } z \vdash np\backslash s : \texttt{wrote}(z)}\,(/\text{E})}{\text{sara wrote } z \vdash s : \texttt{wrote}(z)(\texttt{sara})}\,(\backslash\text{E})}{\text{sara wrote} \vdash s/np : \lambda z.\texttt{wrote}(z)(\texttt{sara})}\,(/\text{I})^1$$

$$\Downarrow$$

The introduction rules correspond to λ-abstraction.

6.6. NP and quantified NP

John and one student left.

We can assign to John the category np and term assignment `john` and **derive** the category and term of quantified np.

$$\dfrac{\dfrac{\text{john} \vdash np : \texttt{john} \quad [P \vdash np\backslash s : P]^1}{\text{john } P \vdash s : P(\texttt{john})}\,(\backslash\text{E})}{\text{john} \vdash s/(np\backslash s) : \lambda P.P(\texttt{john})}\,(/\text{I})^1$$

We have proved: $np \vdash s/(np\backslash s)$. This means, we can assign John the category np (considering it an entity, i.e. a term of type e) and derive from it the **higher order category** of quantified NP as it would be necessary for, e.g. coordination of a NP and a QP.

More on this with RZ.

6.7. Remarks

First of all, note how the system assigns a variable to the hypothesis. The latter is **discharged** by means of [/I] (or [\I]) which corresponds to the abstraction over the variable.

Starting from the labelled lexicon, the task for the Lambek derivational engine is to compute the lambda term representing the meaning assembly for a complex structure as a **by-product** of the derivation that establishes its grammaticality.

7. Conclusion

We have seen that:

- FS main concern is compentence, not perfomance.
- Denotations have a crucial role.
- The denotation of sentence is the truth value; of noun phrase entities, of IV, TV, etc. are functions $e \to t, e \to (e \to t)$, etc.
- functions can be represented and composed by the λ-calculus.

8. Criticisms to FS: it is not a Cognitive Semantics

- Poor representation of the semantic content of words.
- Do humans have sets in their heads? (Cognitive plausibility.)
- Is there truth?
- Where do models come from?

But these issues were not in the agenda of Formal semanticists.