Computational Linguistics: History & Comparison of Formal Grammars

Raffaella Bernardi

Contents

1. Formal Grammars

- We have seen that Formal Grammars play a crucial role in the research on Computational Linguistics.
- We have looked at Context Free Grammars/Phrase Structure Grammars which were "imported" from CS.

But through the years, computational linguists have developed other formal grammars too.

With Alberto, you have looked at:

- Unification-Based Grammars
- Dependency Grammars

Today, we give a brief histortical overview of FGs and do exercizes on CFG.

2. Undergeneration and Overgeneration

We would like the Formal Grammar we have built to be able to recognize/generate **all and only** the grammatical sentences.

- **Undergeration**: If the FG does not generate some sentences which are actually grammatical, we say that it undergenerates.
- **Overgeneration**: If the FG generates as grammatical also sentences which are not grammatical, we say that it overgenerates.

2.1. Undergeneration: Long-distance dep.

Consider these two English np. First, an np with an object relative clause:

"The witch who Harry likes".

Next, an np with a subject relative clause:

"Harry, who likes the witch."

What is their syntax? That is, how do we build them?

2.2. Relative clauses

The traditional explanation basically goes like this. We have the following sentence:

Harry likes the witch

We can think of the np with the object relative clause as follows.

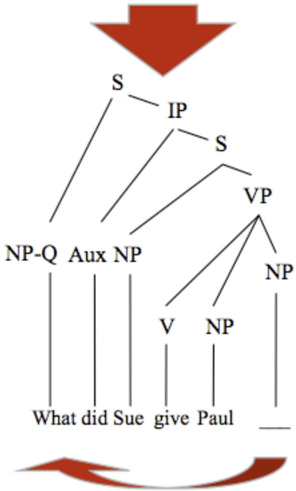
```
    -----------------------
   |                       |
the witch who Harry likes GAP(np)
```

That is, we have

1. extracted the np "the witch" from the object position, leaving behind an np-**gap**,
2. **moved** it to the front, and
3. placed the relative pronoun "who" between it and the gap-containing sentence.

The Transformational Tradition (cont.)

Sue gave Paul an old penny



2.3. Overgeneration: Agreement

For instance, can the CFG we have built distinguish the sentences below?

1. He hates a red shirt
2. *He like a red shirt
3. He hates him
4. *He hates he

With Alberto you have seen how to encode features and feature pergolatation in Unification-Based Grammar.

3. History of Formal Grammars

Important steps in the historical developments of Formal grammar started in the 1950's and can be divided into five phases:

1. Formalization: Away from descriptive linguistics and behavioralism (performance vs. competence) [1950's 1960's]
2. Inclusion of meaning: Compositionality [1970's]
3. Problems with word order: Need of stronger formalisms [1970's 1980's]
4. Grammar meets logic & computation [1990's]
5. Grammar meets statistic [1990's 2000's]

In these phases, theoretical linguists addressed similar issues, but worked them out differently depending on the perspective they took:

- constituency-based or
- dependency-based.

3.1. Reminder: Constituency-based vs. Dependency-based

Constituency (cf. structural linguists like Bloomfield, Harris, Wells) is a **horizontal** organization principle: it groups together constituents into phrases (larger structures), until the entire sentence is accounted for.

- Terminal and non-terminal (phrasal) nodes.
- Immediate constituency: constituents need to be adjacent (CFPSG).
- But we have seen that meaningful units may not be adjacent –Discontinuous constituency or long-distance dependencies.
- This problem has been tackled by allowing flexible constituency: "phrasal rebracketing"

Dependency is an asymmetrical relation between a head and a dependent, i.e. a **vertical** organization principle.

3.2. Reminder: Constituency vs. Dependencies

Dependency and constituency describe different dimensions.

1. A phrase-structure tree is closely related to a derivation, whereas a dependency tree rather describes the product of a process of derivation.
2. Usually, given a phrase-structrue tree, we can get very close to a dependency tree by constructing the transitive collapse of headed structures over nonterminals.

Constituency and dependency are not adversaries, they are complementary notions.

Using them together we can overcome the problems that each notion has individually.

4. Meaning entered the scene

Chomsky was, in general, **sceptical of efforts to formalize semantics**. Interpretative semantics or the autonomy of syntax: Syntax can be studied without reference to semantics (cf. also Jackendoff).

Different ongoing efforts

- Developing a notion of (meaningful) logical form, to which a syntactic structure could be mapped using transformations. Efforts either stayed close to a constituency-based notion of structure, like in generative semantics (Fodor, Katz), or were dependency-based (Sgall et al, particularly Panevová (1974; 1975); Fillmore (1968)). Cf. also work by Starosta, Bach, Karttunen.
- Montague's formalization of semantics – though Montague and the semanticists in linguistics were unaware of one another, cf. (Partee, 1997)

5. Grammars meet Logic & ...

Logics to specify a **grammar** framework as a mathematical system:

- Feature logics: HPSG, cf. (King, 1989; Pollard and Sag, 1993; Richter et al., 1999)
- Categorial Type Logics (Kurtonina, 1995; Moortgat, 1997)

Logics to interpret linguistically realized **meaning**:

- Montague semantics: used in early LFG, GPSG, Montague Grammar, Categorial Type Logic, TAG (Synchronous LTAG)
- Modal logic: used in dependency grammar frameworks, e.g. (Broeker, 1997; Kruijff , 2001).
- Linear logic: used in contemporary LFG, (Crouch and van Genabith, 1998).

6. .. Computation

Computation of linguistic structures

- Unification (constraint-based reasoning): LFG, HPSG, categorial grammar (UCG), dependency grammar (UDG, DUG, TDG)
- "Parsing as deduction": CTL
- Optimality theory: robust constraint-solving, e.g. LFG

Statistics You have seen part of this with Alberto.

7. Next time

Next time we will look at mathematical grammar framework: TAG and CG.