Computational Linguistics: Context Free Grammars

Raffaella Bernardi

e-mail: raffaella.bernardi@unitn.it

1. Syntax

- **Syntax**: "setting out things together", in our case things are words. The main question addressed here is "*How do words compose together to form a grammatical sentence (s) (or fragments of it)?*"

- **Constituents**: Groups of categories may form a single *unit or phrase* called constituent. The main phrases are noun phrases (*np*), verb phrases (*vp*), prepositional phrases (*pp*). Noun phrases for instance are: "she"; "Michael"; "Rajeev Goré"; "the house"; "a young two-year child".

  Tests like substitution help decide whether words form constituents.

  Another possible test is coordination.

2. Dependency

Dependency: Categories are interdependent, for example

Ryanair services [Pescara]$_{np}$	Ryanair flies [to Pescara]$_{pp}$
*Ryanair services [to Pescara]$_{pp}$	*Ryanair flies [Pescara]$_{np}$

the verbs services and flies determine which category can/must be juxtaposed. If their constraints are not satisfied the structure is ungrammatical.

3. Long-distance Dependencies

Interdependent constituents need not be juxtaposed, but may form long-distance dependencies, manifested by gaps

- What cities does Ryanair service [. . .]?

The constituent what cities depends on the verb service, but it is at the front of the sentence rather than at the object position.

Such distance can be large,

- Which flight do you want me to book [. . .]?
- Which flight do you want me to have the travel agent book [. . .]?
- Which flight do you want me to have the travel agent nearby my office book [. . .]?

3.1. Relative Pronouns

Relative Pronoun (eg. who, which): they function as e.g. the subject or object of the verb embedded in the relative clause (rc),

- [[the [student [who [. . .] knows Sara]$_{rc}$]$_{n}$]$_{np}$ [left]$_{v}$]$_{s}$.
- [[the [book [which Sara wrote [. . .]]$_{rc}$]$_{n}$]$_{np}$ [is interesting]$_{v}$]$_{s}$.

3.2. Coordination

Coordination: Expressions of the same syntactic category can be coordinated via "and", "or", "but" to form more complex phrases of the same category. For instance, a coordinated verb phrase can consist of two other verb phrases separated by a conjunction:

- There are no flights [[leaving Denver]$_{vp}$ and [arriving in San Francisco]$_{vp}$]$_{vp}$

The conjuncted expressions belong to traditional constituent classes, *vp*. However, we could also have

- I [[[want to try to write [. . .]] and [hope to see produced [. . .]]] [the movie]$_{np}$]$_{vp}$"

Again, the interdependent constituents are disconnected from each other.

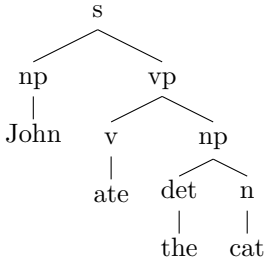
Long-distance dependencies are **challenging phenomena** for formal approaches to natural language analysis.

3.3. Sentence Structures: English

The structure of a sentence can be represented in several ways, the most common are the following notations: (i) brackets or (ii) trees. For instance, "John ate the cat":

$$[\text{John}_{np}\ [\text{ate}_{v}\ [\text{the}_{det}\ \text{cat}_{n}]_{np}]_{vp}]_{s}$$

In words, it is a sentence (s) consisting of noun phrase (np) and a verb phrase (vp). The noun phrase is composed of a verb (v) "ate" and an np, which consists of a determiner (det) "the" and a common noun (n) "cat".



4. Formal Approaches

To examine how a string can be computed, two things must be considered:

1. **The grammar**: A formal specification of the structures allowable in the language. [Data structures]
2. **The parsing technique**: The method of analyzing the string. [Algorithm]

Today we ask:

1. Which Grammar do we need to analyse NLs syntax?
2. Which Formal Language can represent NL?

5. Formal Grammar: Terminology

Formal Grammars are string **re-write systems**. The re-write rules say that a certain sequence of symbols may be substituted by another sequence of symbols. These symbols are divided into two classes:

- **terminal**: symbols that will appear in the string of the language generated by the grammar.
- **non-terminal**: symbols that will be used only in the re-write process.

Given a string, we want to know whether it belongs to the (Natural) Language.

6. Formal Grammars: Definition

A Grammar, G, is a tuple: $G = (V_T, V_N, S, P)$, such that:

- V_T is the finite set of Terminal Symbols.
- V_N is the finite set of Non-Terminal Symbols.
- Terminal and Non-Terminal symbols give rise to the alphabet: $V = V_T \cup V_N$.
- Terminal and Non-Terminal symbols are disjoint sets: $V_T \cap V_N = \{\}$.
- S is the start symbol of the Language, and $S \in V_N$.
- P is the finite set of Productions, $P = \{\alpha \rightarrow \beta \mid \alpha \in \wedge \beta \in V^*\}$.

6.1. Derivations

To characterize a Language starting from a Grammar we need to introduce the notion of Derivation.

- The notion of Derivation uses Productions to generate a string starting from the Start symbol S.
- Direct Derivation (in symbols $\Rightarrow$)). If $\alpha \to \beta \in P$ and $\gamma, \delta \in V^*$, then $\gamma\alpha\delta \Rightarrow \gamma\beta\delta$).
- Derivation (in symbols $\Rightarrow^*$)). If $\alpha_1 \Rightarrow \alpha_2$, $\alpha_2 \Rightarrow \alpha_3$, $\ldots, \alpha_{n-1} \Rightarrow \alpha_n$, then $\alpha_1 \Rightarrow^* \alpha_n$.

6.2. Formal Languages and FG

A string belongs to a Language if and only if:

1. The string is made only of Terminal Symbols;
2. The string can be Derived from the Start Symbol, S, of the Language.

Generative Definition of a Language We say that a Language L is **generated** by the Grammar G, in symbols $L(G)$, if:

$$L(G) = \{w \in V_T^* \mid S \Rightarrow^* w\}.$$

6.3. FG and Regular Languages

We have said that the languages generated/recognized by a FSA are called "Regular Languages".

The formal grammars that generate/recognize these languages are known as "Regular Grammar" (RG) or Right Linear Grammars. (or Left Linear Grammar).

Regular Grammars have rules of the form:

- $A \to xB$
- $A \to x$

where A and B are non-terminal symbols and x is any string of terminals (possibly empty).

Moreover, a rule of the form: $S \to \epsilon$ is allowed if S does not appear on the right side of any rule.

6.4. FSA and RG

The association between FSA and RG is straight:

RG	FSA
$A \to xB$	from state A to state B reading x
$A \to x$	from state A reading x to a designed final state.
start symbol	initial state.

As in FSA, the string already generated/recognized by the grammar has no influence on the strings to be read in the future (no memory!).

7. Syntax Recognizer/Generator

In lecture 1, we have used FSA to recognize/generate natural language morphology. We have said that FSA recognize/generate "Regular Languages".

Can FSA or their corresponding grammar (RG) be used to recognize/generate NL syntax?

Pumping Lemma Recall: For instance, a non-regular language is, e.g., $L = \{a^n b^n \mid n \geqslant 0\}$. More generally, FSA cannot generate/recognize balanced open and closed parentheses. The **Pumping Lemma** proves that L is not a regular language

Roughly note that with FSA you cannot record (**no memory**) any arbitrary number of a's you have read, hence you cannot control that the number of a's and b's has to be the same. In other words, you cannot account for the fact that there exists a **relation of dependency** between a^n and b^n.

It has been shown that at the **syntactic level NLs are not regular**.

8. NLs are not RL: Example I

1. The cat died.
2. The cat the dog chased died.
3. The cat the dog the rat bit chased died.
4. ...

Let, determiner+noun be in the set A : { the cat, the dog, ...}, and the transitive verbs in B : { chased, bit, ...}. Thus the strings illustrated above are all of the form:

$x^n y^{n-1}$ died, where $x \in A$ and $y \in B$, which can be proved to be not a RL.

8.1. NLs are not RL: Example II

Another evidence was provided by Chomsky in 1956. Let $S_1, S_2, \ldots S_n$ be declarative sentences, the following syntactic structures are grammatical English sentences:

- If S_1, then S_2
- Either S_3, or S_4
- The man who said S_5 is arriving today

In each case there is a lexical dependency between one part of each structure and another. "If" must be followed by "then" "either" must be followed by "or".

Moreover, these sentences can be embedded in English one in another.

> **If either** the man who said S_5 is arriving today **or** the man who said S_5 is arriving tomorrow, **then** the man who said S_6 is arriving the day after.
> Let

if	$\rightarrow a$
then	$\rightarrow a$
either	$\rightarrow b$
or	$\rightarrow b$
other words	$\rightarrow \epsilon$

The sentence above would be represented as **abba**.

We can prove via the Pumping Lemma that this language is not in a regular language.

Again, this is an example of open and closed balanced parentheses (or **nested dependencies**) that are not in RL.

9. Context Free Grammars

Formal Grammar **more powerful** than Regular Grammars are Context Free Grammars (CFG).

These grammars are called **context free** because all rules contain only one symbol on the left hand side — and wherever we see that symbol while doing a derivation, we are free to replace it with the stuff on the right hand side. That is, the 'context' in which a symbol on the left hand side of a rule occurs is unimportant — we can always use the rule to make the rewrite while doing a derivation.

A language is called context free if it is generated by some context free grammar.

Well known CFG are **Phrase Structure Grammars** (PSG) , they are based on **rewrite rules**. They can be used for both understanding and generating sentences.

10. CFG: Formal Language

Let's start by using simple grammars that generate formal languages. E.g., take the grammar below.

Rules

Rule 1	$S \to A\ B$	Rule 2	$S \to A\ S\ B$
Rule 3	$A \to a$	Rule 4	$B \to b$

the above grammar let us rewrite 'S' to 'aabb'.

```
S
ASB Rule 2
aSB Rule 3
aSb Rule 4
aABb Rule 1
aaBb Rule 3
aabb Rule 4
```

Such a sequence is called a **derivation** of the symbols in the last row, in this case, i.e. a derivation of the string 'aabb' ($S \Rightarrow^* aabb$).

10.1. CFG: More derivations

Note that there may be many derivations of the same string. For example,

```
S
ASB Rule 2
ASb Rule 4
aSb Rule 3
aABb Rule 1
aAbb Rule 4
aabb Rule 3
```

is another derivation of ‘aabb’.

10.2. CFG: Language Generated

The above grammar generates the language $a^n b^n - \epsilon$ (the language consisting of all strings consisting of a block of a's followed by a block of b's of equal length, except the empty string).

If we added the rule $S \to \epsilon$ to this grammar we would generate the language $a^n b^n$. Therefore, these two languages **are context free**.

On the other hand, $a^n b^n c^n$ **is not**. That is, no matter how hard you try to find CFG rules that generate this language, you won't succeed. No CFG can do the job. The same holds for, e.g. $a^n b^m c^n d^m$.

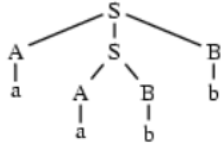
Again, there are formal ways to prove whether a language is or is not context free.

11. FG and Natural Language: parse trees

CFG rules can be used to verify which trees are admissible. A tree is licensed by the grammar when the presence of every node with daughters can be justified by some rule.

There is a close correspondence between parse trees and derivations: every derivation corresponds to a parse tree, and every parse tree corresponds to (maybe many) derivations.

For example, the two derivations above correspond to the same tree.



12. FG for NL: Lexicon vs. Grammatical Rules

CFG applied to natural language: it is convenient to distinguish rules from non-terminal to terminal symbols which define the lexical entries (or lexicon).

- Terminal: The terminal symbols are **words** (e.g. sara, dress ...).
- Non-terminal: The non-terminal symbols are syntactic **categories** (CAT) (e.g. *np*, *vp*, ...).
- Start symbol: The start symbol is the s and stands for **sentence**.

The production rules are divided into:

- Lexicon: e.g. $np \rightarrow$ sara.
- Grammatical Rules: They are of the type $s \rightarrow np\ vp$.

13. PSG: English Toy Fragment

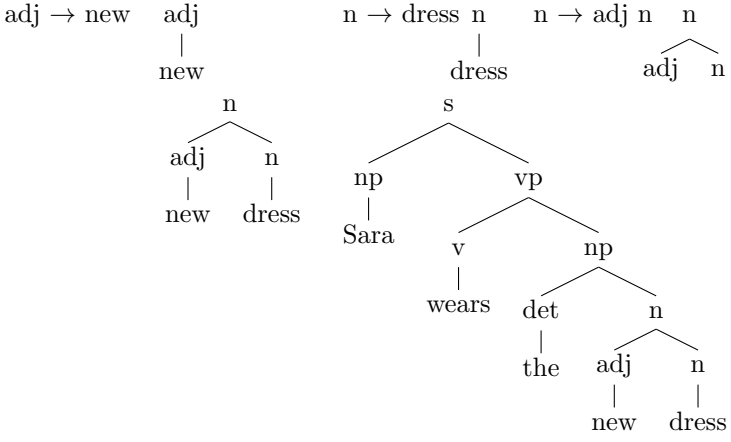
We consider a small fragment of English defined by the following grammar $G = \langle \mathsf{LEX}, \mathsf{Rules}, V, CAT \rangle$.

- $V = \{\text{Sara}, \text{dress}, \text{wears}, \text{the}, \text{new}\}$,
- $\mathsf{CAT} = \{det, n, np, s, v, vp, adj\}$,
- $\mathsf{LEX} = \{np \rightarrow \text{sara}, det \rightarrow \text{the}, adj \rightarrow \text{new}, v \rightarrow \text{wears}\}$
- $\mathsf{Rules} = \{s \rightarrow np\ vp,\ np \rightarrow det\ n,\ vp \rightarrow v\ np, n \rightarrow adj\ n\}$

14. English Toy Fragment: Strings

S
np vp
Sara vp
Sara v np
Sara wears np
Sara wears det n
Sara wears the n
Sara wears the adj n
Sara wears the new n
Sara wears the new dress

15. English Toy Fragment: Phrase Structure Trees



16. Summing up (I)

We have seen that

- There is a close correspondence between **parse trees** and **derivations**: every derivation corresponds to a parse tree, and every parse tree corresponds to (maybe many) derivations.
- PSG, besides deciding whether a string belongs to a given language, deals with phrase structures represented as trees.
- An important difference between strings and phrase structures is that whereas string concatenation is assumed to be associative, trees are bracketed structures.
- Thus trees preserve aspects of the compositional (constituent) structure or derivation which is lost in the string representations.

17. Summing up (II)

- The **language generated** by a grammar consists of all the strings that the grammar classifies as grammatical.
- A CFG **recognizer** is a program that correctly tells us whether or not a string belongs to the language generated by a CFG.
- A CFG **parser** is a program that correctly decides whether a string belongs to the language generated by a CFG and also tells us what its structure is.
- A **Context Free Language** is a language that can be generated by a CFG.

18. Next class

On the 5th of October, during the blended lecture (in class and zoom: 10:30-12:00):

- you will answer a quizz about the content of this video,
- we will look at the answers and discuss them together.