Computational Linguistics: TAG, CG and DG

Raffaella Bernardi

University of Trento

1. Last time and today

- We have seen that Formal Grammars have played a crucial role in the research on Computational Linguistics.
- We have looked at Context Free Grammars/Phrase Structure Grammars.

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

Today, we will look at TAG, CG and DG..

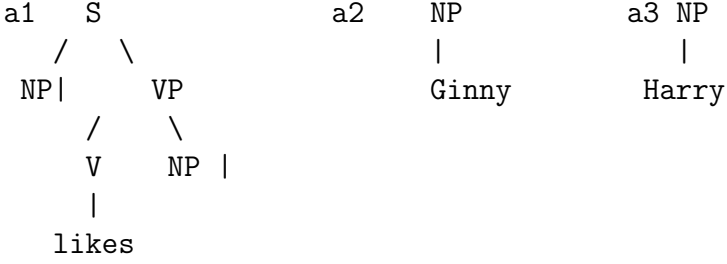
2. Tree Adjoining Grammar (TAG)

- **Who:** Aravind Joshi (1969).
- **Aim:** To build a language recognition device.
- **How:** Linguistic strings are seen as the result of concatenation obtained by means of syntactic rules starting from the trees assigned to lexical items. The grammar is known as Tree Adjoining Grammar (TAG).
- http://www.cis.upenn.edu/~xtag/

2.1. TAG & CFG

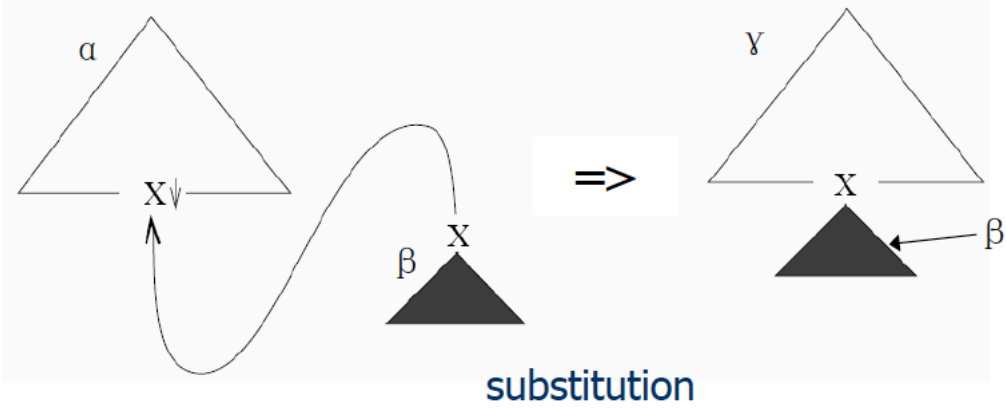
```
CFG:
S -->  NP VP      NP --> Harry       ADV --> passionately
VP --> V NP       NP --> Ginny
VP --> VP ADV     V --> likes
```

TAG: set of lexically anchored elementary trees. The intial trees are:



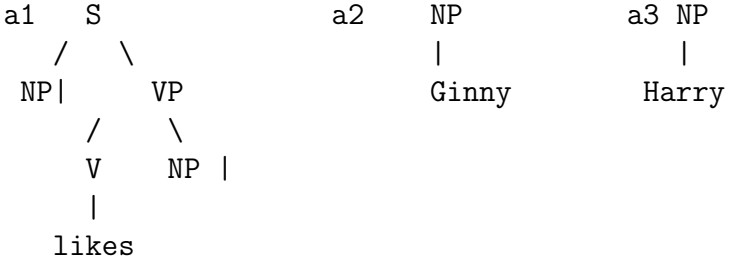
Note: NP | stands for $NP\downarrow$

2.2. TAG rules



2.3. Example

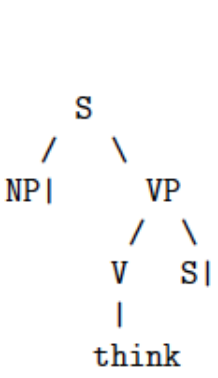
Let's apply the substitution rules to the entries given above:

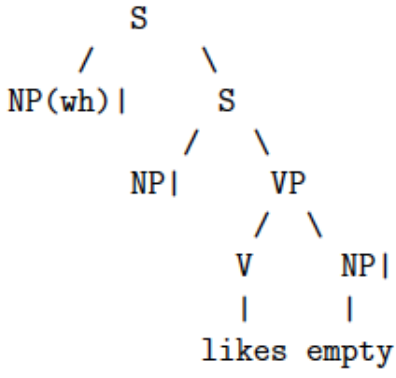


2.4. Example

"Harry thinks Ron likes Hermione"

We need to add the entry for "thinks":

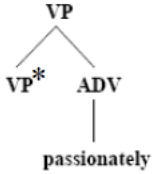




To account for gaps, "Who does Harry think Ron linkes?", new elementary trees are assigned.

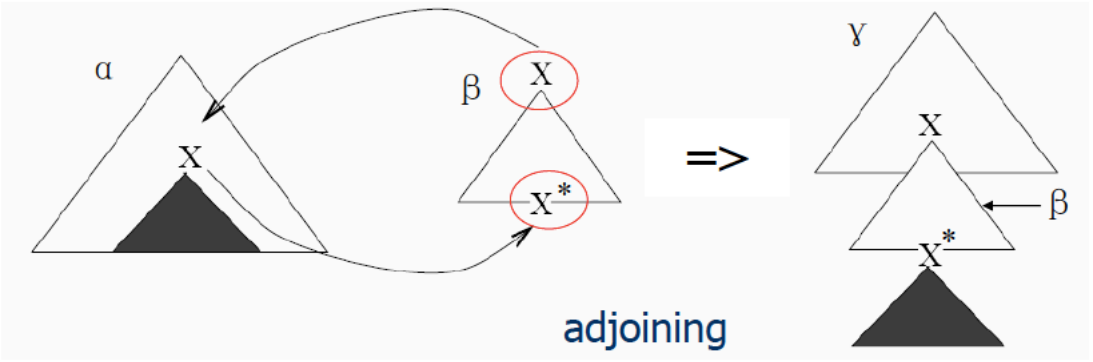
2.5. Auxiliary trees

Elementary trees can also be auxiliary trees, e.g.:



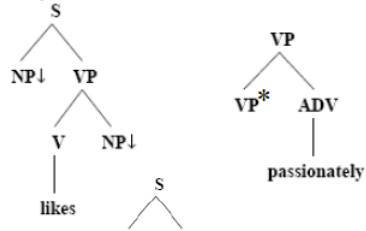
- one of its frontier nodes must be marked as **foot node** (*)
- the foot node must be labeled with a non-terminal symbol which is identical to the label of the root node.

2.6. Adjunction

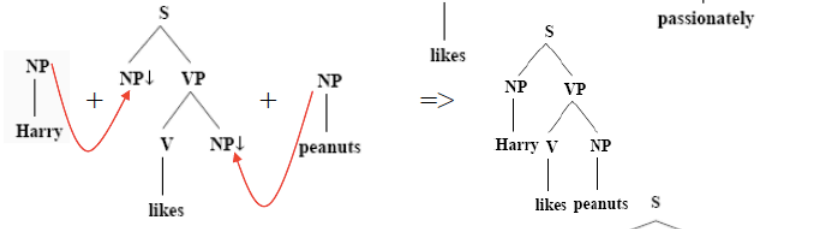


e.g. 'Harry likes peanuts passionately'

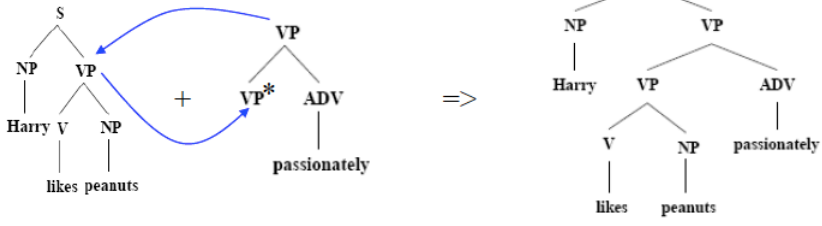
TAG G



step 1
substitution



step 2
adjoining



3. Categorial Grammar

- **Who:** Lesniewski (1929), Ajdukiewicz (1935), Bar-Hillel (1953).
- **Aim:** To build a language recognition device.
- **How:** Linguistic strings are seen as the result of concatenation obtained by means of syntactic rules starting from the categories assigned to lexical items. The grammar is known as Classical Categorial Grammar (CG).

Categories: Given a set of basic categories ATOM, the set of categories CAT is the smallest set such that:

$$\mathsf{CAT} := \mathsf{ATOM} \mid \mathsf{CAT}\backslash\mathsf{CAT} \mid \mathsf{CAT}/\mathsf{CAT}$$

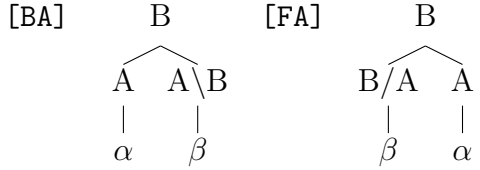
4. CG: Syntactic Rules

Categories can be composed by means of the syntactic rules below

`[BA]` If α is an expression of category A, and β is an expression of category $A\backslash B$, then $\alpha\beta$ is an expression of category B.

`[FA]` If α is an expression of category A, and β is an expression of category B/A, then $\beta\alpha$ is an expression of category B.

where `[FA]` and `[BA]` stand for Forward and Backward Application, respectively.



5. CG Lexicon: Toy Fragment

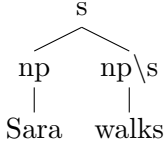
Let ATOM be $\{n, s, np\}$ (for nouns, sentences and noun phrases, respectively) and LEX as given below. Recall CFG rules: $np \to det\ n, s \to np\ vp, vp \to v\ np \ldots$

Lexicon

Sara	np	the	np/n
student	n	walks	$np\backslash s$
wrote	$(np\backslash s)/np$		

Sara walks $\in s$? $\rightsquigarrow$ $\underbrace{np}_{\text{Sara}}, \underbrace{np\backslash s}_{\text{walks}} \in s$? Yes

simply [BA]



6. Classical Categorial Grammar

Alternatively the rules can be thought of as Modus Ponens rules and can be written as below.

$$B/A, A \Rightarrow B \qquad \mathrm{MP_r}$$
$$A, A\backslash B \Rightarrow B \qquad \mathrm{MP_l}$$

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

7. Classical Categorial Grammar. Examples

Given $\mathsf{ATOM} = \{np, s, n\}$, we can build the following lexicon:

Lexicon

John, Mary	$\in$	np	the	$\in$	np/n
student	$\in$	n			
walks	$\in$	$np\backslash s$			
sees	$\in$	$(np\backslash s)/np$			

Analysis

John walks $\in s$? $\rightsquigarrow$ $np, np\backslash s \Rightarrow s$? Yes

$$\frac{np \quad np\backslash s}{s}\ (\mathrm{MP_l})$$

John sees Mary $\in s$? $\rightsquigarrow$ $np, (np\backslash s)/np, np \Rightarrow s$? Yes

$$\frac{np \qquad \dfrac{(np\backslash s)/np \quad np}{np\backslash s}\ (\mathrm{MP_r})}{s}\ (\mathrm{MP_l})$$

7.1. CFG and CG

Below is an example of a simple CFG and an equivalent CG:

CFG

```
S --> NP VP
VP --> TV NP
N --> Adj N

Lexicon:
Adj --> poor
NP --> john
TV --> kisses
```

CG Lexicon:

John:	np
kisses:	$(np\backslash s)/np$
poor:	n/n

7.2. Relative Pronoun

Let's see the syntactic category of a relative pronoun in subject position E.g. "the student **who** knows Lori"

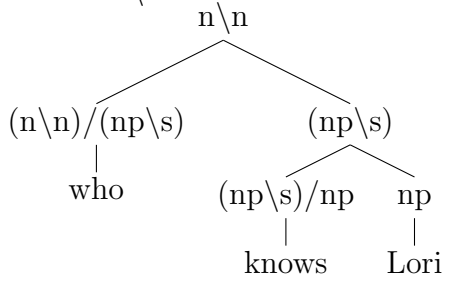
the $[[\text{student}]_n\ [\text{who}\ [\text{knows Lori}]_{(np\backslash s)}]_?]_n$

who knows Lori $\in n\backslash n$?

$$\rightsquigarrow$$

$$(n\backslash n)/(np\backslash s), (np\backslash s)/np, np \Rightarrow n\backslash n?$$

$$\dfrac{\dfrac{\text{who}}{(n\backslash n)/(np\backslash s)} \quad \dfrac{\dfrac{\text{knows}}{(np\backslash s)/np} \quad \dfrac{\text{Lori}}{np}\,(\text{MP}_\text{r})}{np\backslash s}}{n\backslash n}\,(\text{MP}_\text{r})$$



8. Logic Grammar

- **Aim:** To define the logic behind CG.
- **How:** Considering categories as formulae; $\backslash, /$ as logic connectives.
- **Who:** Jim Lambek [1958]

8.1. Lambek Calculi

In the Lambek Calculus besides the elimination rules of $\backslash, /$ (that we saw in CG) we have their introduction rules.

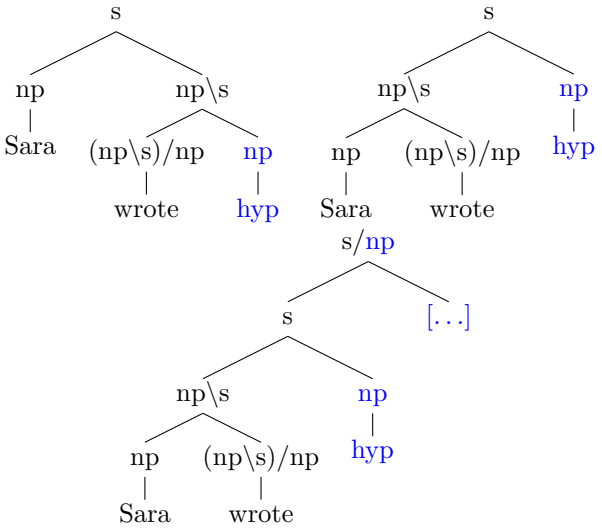
$$\frac{B/A \quad A}{B}\ /\mathrm{E} \qquad \frac{A \quad A\backslash B}{B}\ \backslash\mathrm{E}$$

$$\frac{\begin{array}{c}[A]^i \\ \vdots \\ B\end{array}}{B/A}\ /\mathrm{I}^i \qquad \frac{\begin{array}{c}[A]^i \\ \vdots \\ B\end{array}}{A\backslash B}\ \backslash\mathrm{I}^i$$

9. Extraction: Right-branch (tree)

A book which [Sara wrote [. . .]]$_s$ is interesting .

np $np\backslash s$



10. CCG and TLG

A well known version of CG is CCG (Combinatory Categorial Grammar) developed by Mark Steedman (Edinburgh University).

- CCG Bank
- C&C parser
- C&C parser together with Boxer (MR builder).

Link to some softwares: http://groups.inf.ed.ac.uk/ccg/software.html

Another mathematically elegant version is Type Logical Grammar (TLG) developed by Michael Moortgat (Utrecht University)

- Grail parser: http://www.labri.fr/perso/moot/grail3.html (Richard Moot)

See ESSLLI for various courses on these grammars.

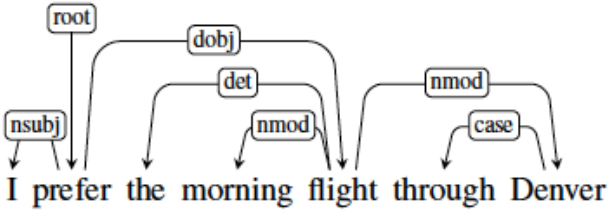
11. History of Formal Grammars

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]
6. We live in the Neural Network era: applications also to parsing.

Two main perspectives: constituency-based or dependency-based.

11.1. Dependency Grammars (example)



The labeled arcs go from heads to dependents. The root is the head of the sentence.

Note, the arguments to the verb "prefer" are directly linked to it vs. in a phrase-structure tree they would be far from each other.

11.2. DG: dependency tree

The dependency tree is a directed graph that satisfies the following constraints:

1. There is a single designated root node that has not incoming arcs.
2. With the exception of the root note, each vertex has exactly one incoming arc.
3. There ia a unique path from the root node to each vertex in V.

11.3. DG: advantages

There are many version of DGs, in general in DG there are no non-terminal or phrasal nodes: each link holds between two lexical nodes. Links can be labelled by the grammatical relation.

(Claimed) Advantages:

- Parsing: less choices about dependency links, hence more precise parsers.
- Better for relatively free word order languages. (eg. Czech)

11.4. 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.

12. Recall: Generative Power/Complexity of FGs

Every (formal) grammar generates a unique language. However, one language can be generated by several different (formal) grammars.

Formal grammars differ with respect to their **generative power**:

One grammar is of a greater generative power than another if it can recognize a language that the other cannot recognize.

Two grammars are said to be

- **weakly** equivalent if they generate the same string language.
- **strongly** equivalent if they generate both the same string language and the same tree language.

12.1. DG, CG, TLG, CCG, and TAG

- DG: Gross (1964)(p.49) claimed that the dependency languages are **exactly** the context-free languages. This claim turned out to be a mistake, and hence there has been new interest in DG. (Used in various NLP application, eg. QA)
- CG: Chomsky (1963) conjectured that **Lambek calculi** were also **context-free**. This conjecture was proved by Pentus and Buszkowski in 1997.
- TAG and CCG: have been proved to be Mildly Context Free.
- TLG (A version of Lambek calculi) has been proved to be Mildly Sensitive (Moot), or Context Sensitive (Moot) or Turing Complete (Carpenter), accordingly to the structural rules allowed.

12.2. Most re-known treebanks

- CFG based: Penn Treebank: By Marcus et ali. first edition 1993. Released via the Linguistic Data Consortium. 1M words 1987-1989 Wall Street Journal (WSJ)
- DG: http://ufal.mff.cuni.cz/prague-english-dependency-treebank.
- DG: (Multilingual) http://universaldependencies.org/
- CCG: http://groups.inf.ed.ac.uk/ccg/ccgbank.html. CCGbank is a translation of the Penn Treebank into a corpus of Combinatory Categorial Grammar derivations.