Syntax 1

- Introduction
- Formal Grammars
- Grammars for NLP

Syntax 2

- Syntax describes regularity and productivity of a language making explicit the structure of sentences
- Goal of syntactic analysis (parsing):
  - Detect if a sentence is correct
  - Provide a syntactic structure of a sentence.

Introduction to Syntax

- It refers to the way words are arranged together
- Basic ideas related to syntax
  - **Contituency**

    Groups of words may behave as a single unit or phrase: a constituent. Exemple: ***Noun phrase***
  - **Grammatical relations**

    Formalization of ideas from traditional grammar. Example: ***subject*** and ***object***
  - **Subcategorization and dependency relations**

    Relations between words and phrases

    Example: Verb ***want*** followed by an ***infinitve verb***

Introduction to Syntax

- Regular languages and part of speech refers to the way words are arranged together but cannot support easily: **Contituency, Grammatical relations and Subcategorization and dependency relations**
- They can be modelled by grammars based on **context-free grammars**
- Context-free grammars is a formalism power enough to represent complex relations and can be efficiently implemented.
- Context-free grammars are integrated in many language applications.

Introduction to Syntax

- A **Context-free grammar** consists of **a set of rules** or productions, each expressing the ways the symbols of the language can be grouped together, and a **lexicon** of words
- An example of a set of rules expressing
  - **NP** (noun phrase) **can be either a ProperNoun or a** determiner (**Det**) **folowed by a Nominal**
  - **A nominal can be one or more Nouns**

NP → Det Nominal

NP → ProperNoun

Nominal → Noun | Nominal Noun

  - Other rules can be added:

Det → a **Det → the** **Noun → flight**

Syntax 3

- Text (string)
  - Free monoid over a vocabulary with the concatenation operation (·)
- Vocabulary (V), set of words (w)
  - $w \in V$
- Language Models (LM)
  - Probability distribution over the texts
- Language (L), set of sentences (s)
  - $s \in L$
  - $L \subset V^*$ usually infinite
- $s = w_1, \ldots w_N$
- Probability of s
  - P(s)

Syntax 4

- Naive Implementation of a LM
  - Enumerate s ∈ L
  - Compute p(s), e.g. counting occurrences on a huge corpus
  - Parameters of the model |L|
- But ...
  - L is usually not enumerable
  - How to estimate the parameters?
- Simplifications
- History
  - $h_i = \{ w_i, \ldots w_{i-1} \}$
  - Usually up to 5-grams
  - Google's n-grams
- Markov Models

$$p(s) = p(w_1 \ldots w_N) = P(w_1^N) = \prod_{i=1}^{N} p(w_i | h_i)$$

Syntax 5

- Language (L) over vocabulary V
  - $L \subset V^*$
- How to define L, i.e. how to decide if $s \in L$?
  - If we use a LM:
    - If $p(s) > 0$ then $s \in L$
  - The usual way: Generative Approach
    - A sentence is correct if it is grammatical (acording to a grammar)
    - Getting a grammar G such that $s \in L_G$ being L_G the language generated (or recognized) by the grammar G
    - There are many types of grammars, the most used are the Phrase Structure Grammars (PSG).
      - G=<N, Σ, P, S>
        - Set of N non terminal symbols
        - Set of Σ terminal symbols
        - Set of P productions or rules
        - $S \in N$, axiom

Syntax 6

- In the generative setting, $w \in L_G$ if s can be generated from the axiom through a number of applications of productions of P.

$$S \rightarrow^* w$$

- In general productions in P are of the form
  - $\forall\ \alpha \rightarrow \beta$
  - $\forall\ \alpha, \beta \in (N \cup \Sigma)^*$
    - i.e. a rule can rewrite whatever chain of elements of the vocabulary $(N \cup \Sigma)$ into another chain
  - but can be constrained into the types of Chomsky hierarchy:
    - Type 0 unrestricted
    - Type 1 context sensitive
    - Type 2 context free
    - Type 3 regular

Syntax

Example of CFG

$G_1 = \langle N_1, T_1, P_1, \text{SENTENCE} \rangle$

$N1 = \{\text{SENTENCE, NP, VP, RNP, PP}\}$

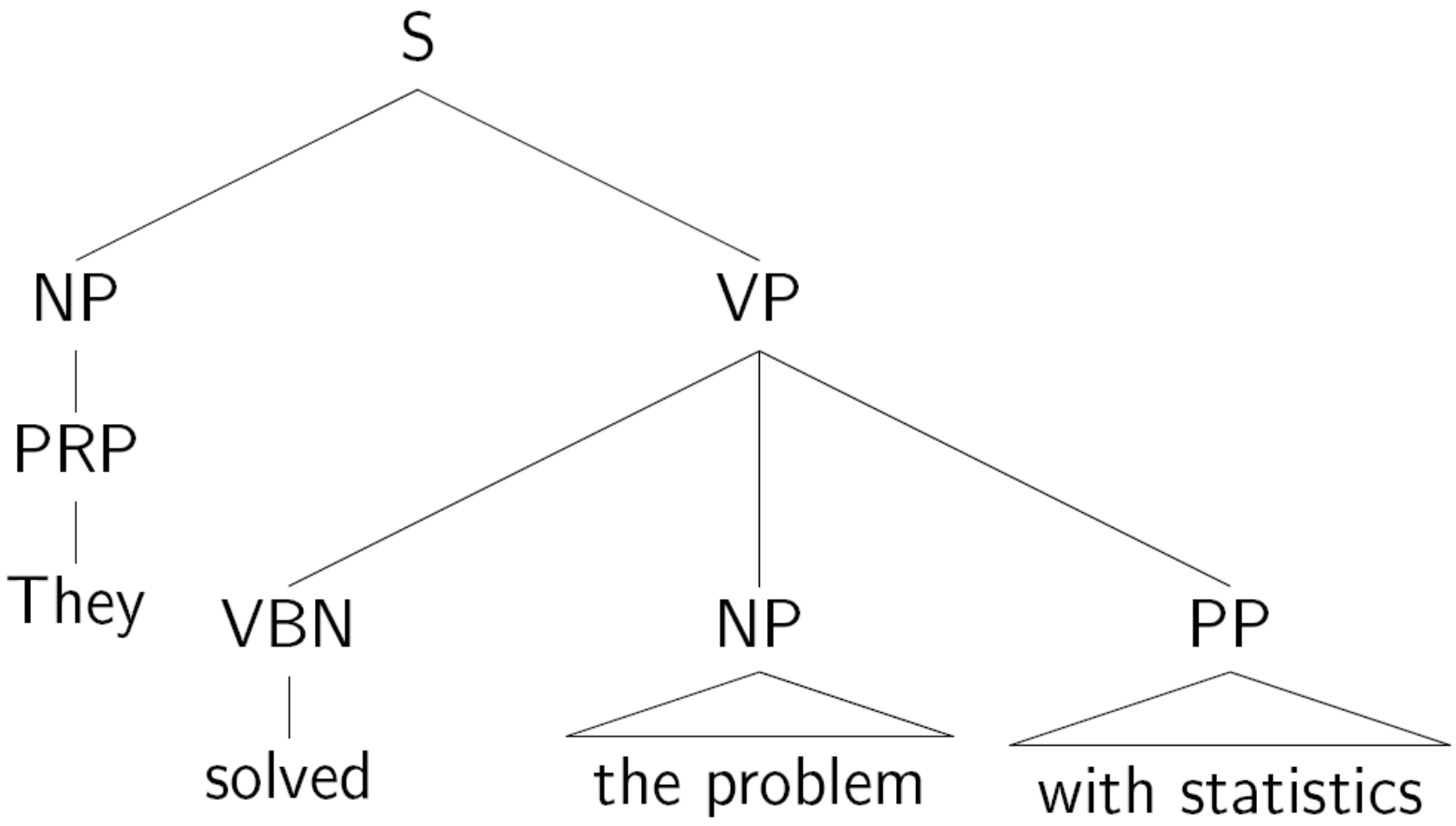
$T_1 = \{\text{det,n,np,adj,vi,vt,prep}\}$

$P_1 = \{$

1 SENTENCE --> NP VP.
2 NP --> det n RNP.
3 NP --> n RNP.
4 NP --> np RNP.
5 RGN --> ε.
6 RGN --> PP RNP.
7 RGN --> adj RNP.
8 VP --> vi.
9 VP --> vt NP.
10 PP --> prep NP.
}

Syntax

Parse tree



Syntax 7

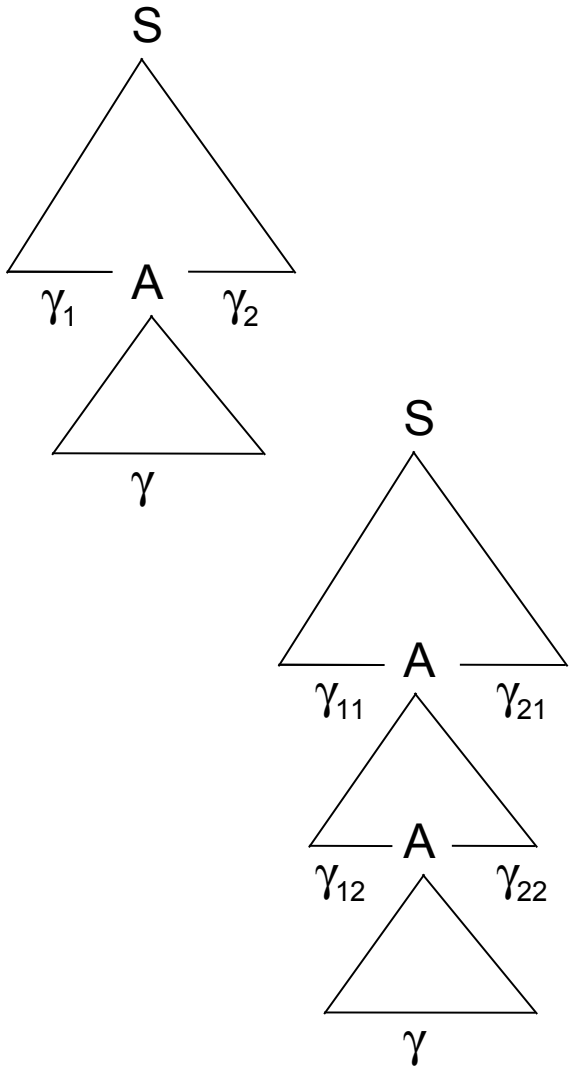
- For practical application in NLP possibilities reduce to
  - Regular grammars (RG)
    - Equivalent to Finite State Automata and regular expressions
    - Parsed in quadratic time $O(n^2)$
  - Context free grammars (CFG)
    - Equivalent to push-down automata
    - Parsed in cubic time $O(n^3)$
    - Important subclass: Deterministic CFG (LR grammars)
    - Always 2-normalizable (Chomsky Normal Form)
  - Middly context sensitive grammars (see Laura Kallmeyer, 2011)
    - Tree Adjoining Grammars (TAG)
    - Linear Context Free Rewriting Systems (LCFRS)
    - Multiple Context Free Grammars (MCFG)
      - Alexander Clark, 2012
    - Range Concatenation Grammars (RCG)

Syntax 8

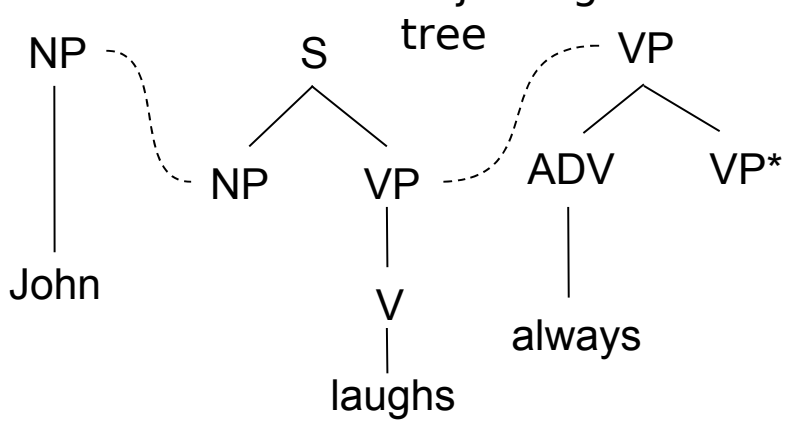
- Lexicalized grammars
  - Grammars such that all the rules contain at least one terminal.
  - The number of analysis of a string is finite
  - CFG cannot be strongly lexicalized
  - CFG can be lexicalized by TAGs
  - TAGs are closed under strong lexicalization
  - A CFG or a TAG has an strongly equivalent lexicalized TAG (LTAG)

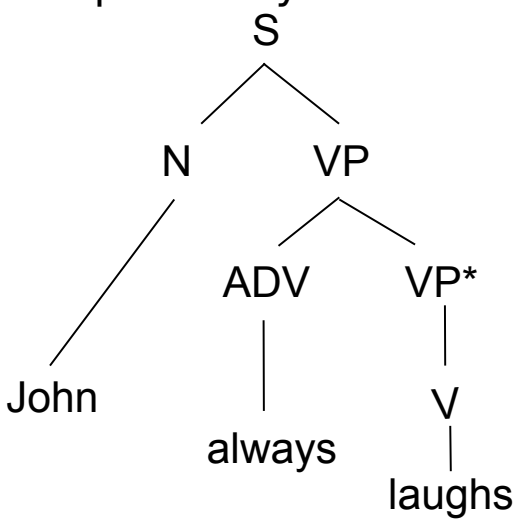
Syntax 9

- CFG
  - A leaf non-terminal symbol corresponding to the left-hand side of a rule is replaced by the right-hand side.
- Middly context sensitive grammars
  - Tree Adjoining Grammars (TAG)
    - We can insert new trees somewhere within the already derived tree, not only at the leaves.
  - Linear Context Free Rewriting Systems (LCFRS)
    - Non-terminals can span tuples of strings and the productions specify how to compute the span of the left hand side non-terminals from the spans of the right-hand side terminals.
    - This leads to trees with crossing branches.
    - Can be learned from constituent or dependency treebanks (Kallmeyer, 2011)



- Tree Adjoining Grammars (TAG)
  - Joshi et al, 1975, Joshi, Schabes, 1997, Kallmeyer, 2011
  - A TAG consists of:
    - Finite set of syntactic trees (Elementary Trees)
    - Finite set of syntactic trees having a foot node, i.e. a leaf correponding to the root of the tree (Auxiliary Trees)
    - Two operations:
      - Replacing: A non-terminal leaf node is replaced by a tree
      - Adjoining: An internal non-terminal is replaced by an auxiliary tree





Syntax 11

- Instead of a text we consider a pair of texts
  - <t_1, t_2> such that t_1 and t_2 have some relation:
    - t_1 is a translation of t_2
    - t_1 entails t_2
    - t_1 and t_2 are paraphrase of each other
    - t_1 is a simplification of t_2
    - t_1 is a compression of t_2
    - t_1 is the interrogative form of the assertion t_2
    - t_1 is the lower case, puntuation signs removed form of t_2
- Generative setting
  - The bigram generates the pair

$$S \xrightarrow{*} < t_1, t_2 >$$

- Discriminative setting
  - Given one element of the pair the grammar generates the other

Syntax 12

- A transduction is a set of sentence translation pairs or bisentences—just as a language is a set of sentences. The set defines a relation between the input and output languages.
- In the generative view, a transduction grammar generates a transduction, i.e., a set of bisentences—just as an ordinary (monolingual) language grammar generates a language, i.e., a set of sentences.
- In the recognition view, alternatively, a transduction grammar biparses or accepts all sentence pairs of a transduction—just as a language grammar parses or accepts all sentences of a language.
- In the transduction view, a transduction grammar transduces (translates) input sentences to output sentences.

Syntax 13

	MONOLINGUAL (Chomsky hierarchy)		BILINGUAL	
			↑	
no binary 2-normal form			syntax-directed transductions **SDTG** *or (or synchronous CFG)*	$O(n^{2n+2})$ for $n \geq 4$
	↑		↑	
has binary 2-normal form	context-free languages **CFG**	$O(n^3)$	inversion transductions **ITG** *or SDTG (or synchronous CFG) that is binary or ternary or inverting*	$O(n^6)$
	↑		↑	
	regular or finite-state languages **FSA** *or CFG that is right regular or left regular*	$O(n^2)$	regular or finite-state transductions **FST** *or SDTG (or synchronous CFG) that is right regular or left regular*	$O(n^4)$

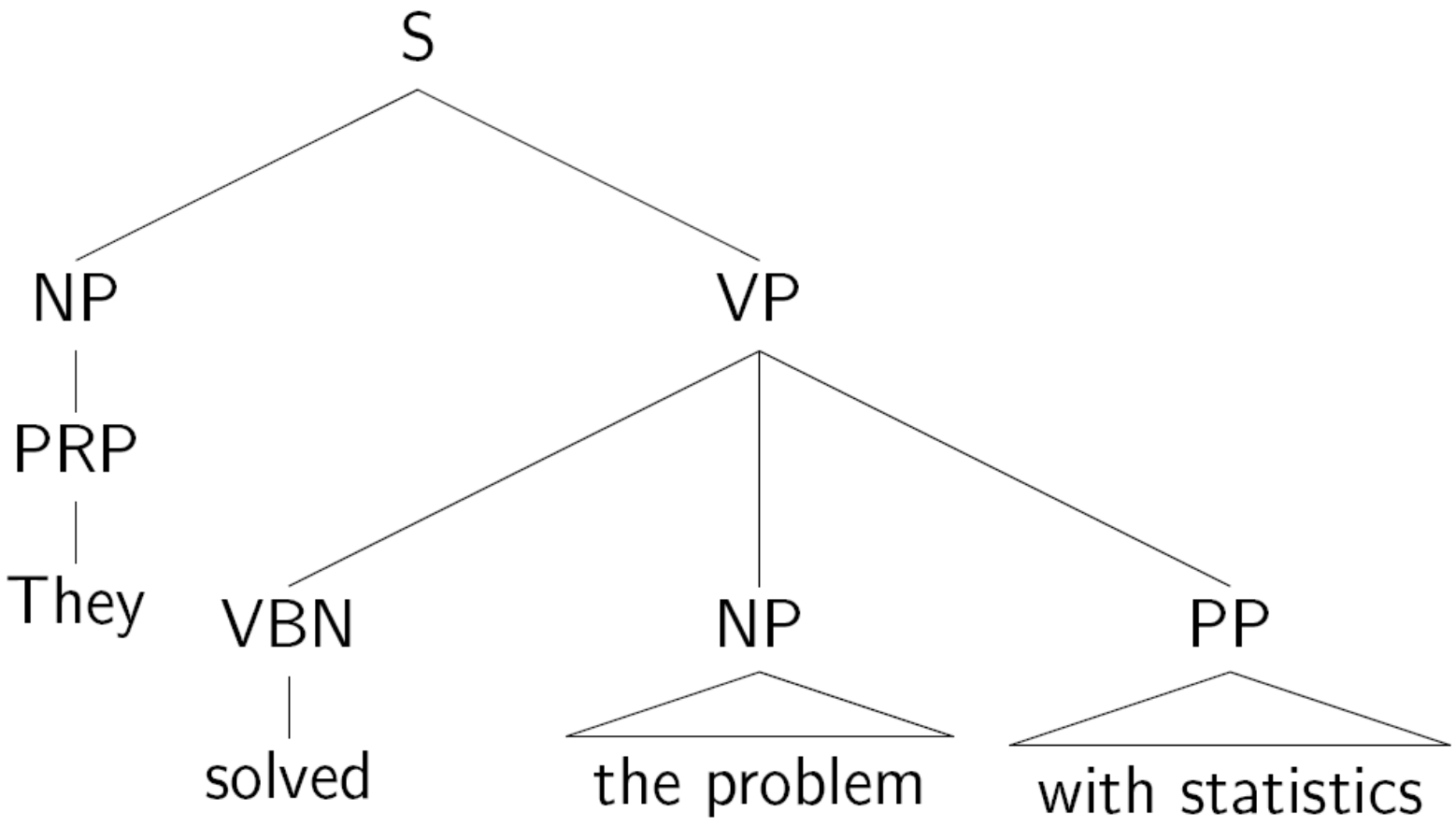
Fig. 4. Summary comparison of computational complexity for Viterbi and chart (bi)parsing, and EM training algorithms for both monolingual and bilingual hierarchies.

Taken from Wu, 2007

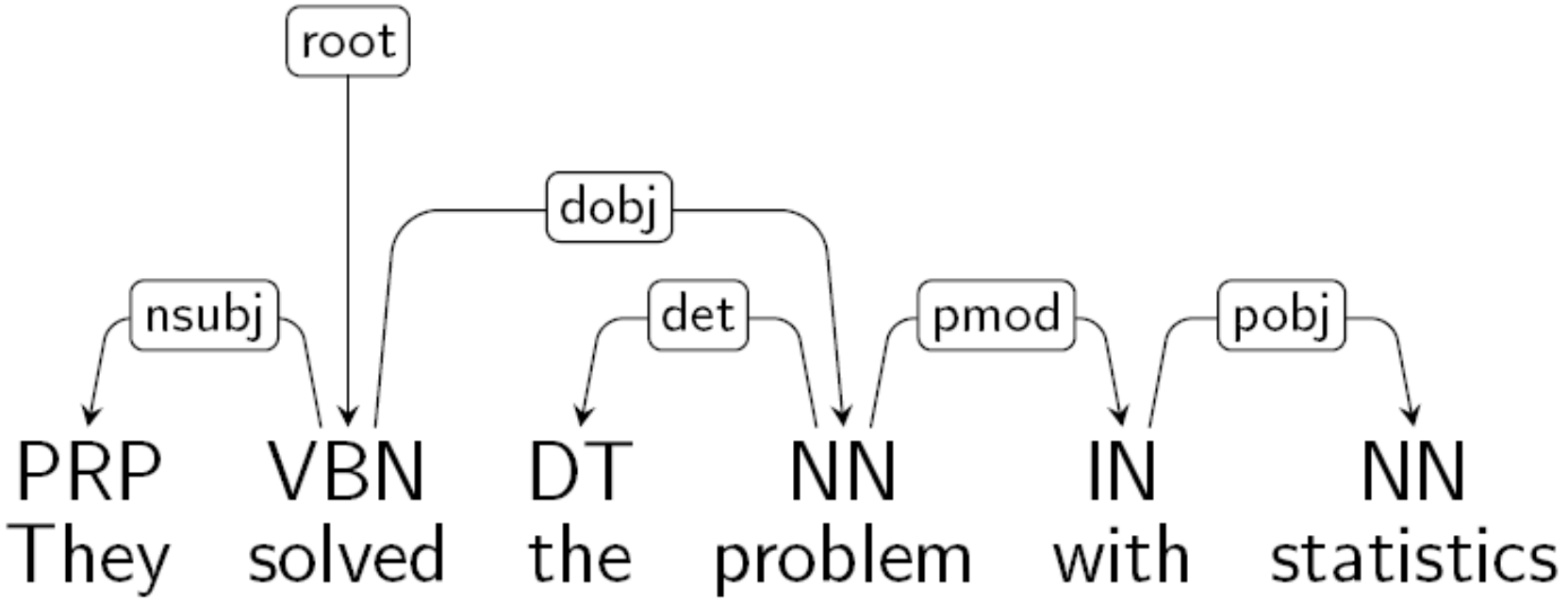
Syntax 14

- Forms of expressing syntactic structure :
  - Constituent Str. (derivation tree = parse tree)
  - Dependency Str.
  - Case Str. (actant models)
  - Transformational grammars
  - Systemic grammars
  - Logical Form

Parse tree



Dependency tree



Syntax 17

Logical Form (close to semantics)

The cat eats fish

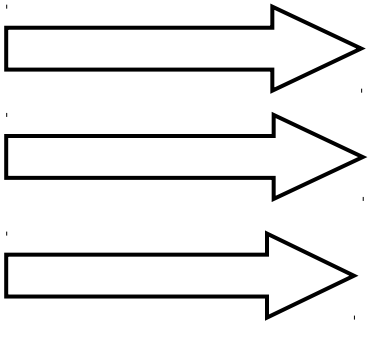
$$(\exists\ X \text{ and } (\text{cat } (X),$$
$$(\exists Y \text{ and } (\text{fish } (Y),$$
$$\text{eat}(X,Y)))))$$

Properties of parsers

- Soundness
  - The output of the parsing is correct according to the grammar
- Termination
  - Any parsing process terminates
- Completedness
  - A parser is complete if given a grammar and a sentence it is sound, produces all the correct parse trees and terminates.

Expressivity of the grammar

- Minimum: CFG
  - except RG for specific applications
- ¿Is NL context free?
- ¿Sufficient? NO (usually)
- Solution



CFG + {procedural addition of context}

Middly context sensitive grammars

Logic Grammars

Example of CFG

$G_1 = \langle N_1, T_1, P_1, \text{SENTENCE} \rangle$

$N1 = \{\text{SENTENCE, NP, VP, RNP, PP}\}$

$T_1 = \{\text{det,n,np,adj,vi,vt,prep}\}$

$P_1 = \{$

1 SENTENCE --> NP VP.
2 NP --> det n RNP.
3 NP --> n RNP.
4 NP --> np RNP.
5 RGN --> ε.
6 RGN --> PP RNP.
7 RGN --> adj RNP.
8 VP --> vi.
9 VP --> vt NP.
10 PP --> prep NP.

}

Example of CFG with procedural extension

intervencion	⟶pregunta | orden | ...
orden	⟶v, sn {imperativo(1), orden(1)}
sn	⟶snbase, [snmods] | np {concordancia(1,2)}
snbase	⟶[det], n, [adjs] {concordancia(1,2,3)}
adjs	⟶adj, [adjs]
snmods	⟶snmod, [snmods]
snmod	⟶sp | ...
sp	⟶prep, sn
np	⟶"barcelona" | "valencia" | ...
n	⟶"billete" | "euromed" ...
v	⟶"deme" | ...
det	⟶"un" | "el" | ...

Syntax 22

Obtaining the grammar

- Definition of Σ from the tagset
- Definition of V
  - $\bar{X}$ theory
  - slashed categories
- Grammar rules P
  - manually
  - automatically
    - Grammatical inference
  - Semi- automatically

Syntax 23

CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular
NNP	Proper noun, singular
NNS	Noun, plural
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Posessive ending
PRP	Personal pronoun
PP	Possessive pronoun

PTB tagset

RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund
VBN	Verb, past participle
VBP	Verb, non-3rd ps. sing. present
VBZ	Verb, 3rd ps. sing. present
WDT	wh-determiner
WP	wh-pronoun
WP	Possessive wh-pronoun
WRB	wh-adverb

PTB tagset 2

#	Pound sign
$	Dollar sign
.	Sentence-final punctuation
,	Comma
:	Colon, semi-colon
(	Left bracket character
)	Right bracket character
"	Straight double quote
`	Left open single quote
``	Left open double quote
'	Right close single quote
''	Right close double quote

Changing the grammar

- Transformations of grammars for getting equivalent ones :
  - Removing symbols and productions that cannot be reached
  - Removing unary productions
  - Removing ε productions
  - Lexicalization
  - Normal Forms
    - Binarization
    - Chomsky
    - Greibach
- Aproximation of CFG by RG

Chomsky NF

- A CFG is in CNF iff only productions of the following type are allowed:
  - unary $A \rightarrow a$
  - binary $A \rightarrow BC$
  - with $a \in \Sigma$ and $A,B,C \in V$
- Getting the CNF of a CFG is trivial

Greibach NF

- A CFG is in GNF iff only productions of the following type are allowed: :
  - $A \rightarrow a\ \alpha$
  - with $a \in \Sigma$ and $\alpha \in V^*$
- Getting the GNF of a CFG is trivial