Syntax

- Introduction
- Formal Grammars
- Grammars for NLP

Introduction to Syntax_1

- 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 $Syntax_2$

- 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

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

Introduction to Syntax$_3$

- 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$_4$

- 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**

Formal Grammars$_1$

- 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
- $s = w_1, \ldots w_N$

P(s) Probability of s

Formal Grammars$_2$

- Naive Implementation of a LM
  - Enumerate s $\subset$ L
  - Compute p(s), e.g. counting occurrences
  - 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

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

Formal Grammars$_3$

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) or Context Free Grammar (CFG).
    - **$G=\langle N, \Sigma, P, S\rangle$**
      - Set of N non terminal symbols
      - Set of Σ terminal symbols
      - Set of P productions or rules
      - $S \in N$, axiom

Grammars for NLP_1

Example of CFG

$G_1 = \langle N_1, T_1, P_1, SENTENCE \rangle$

$N_1 = \{SENTENCE, NP, VP, RNP, PP\}$

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

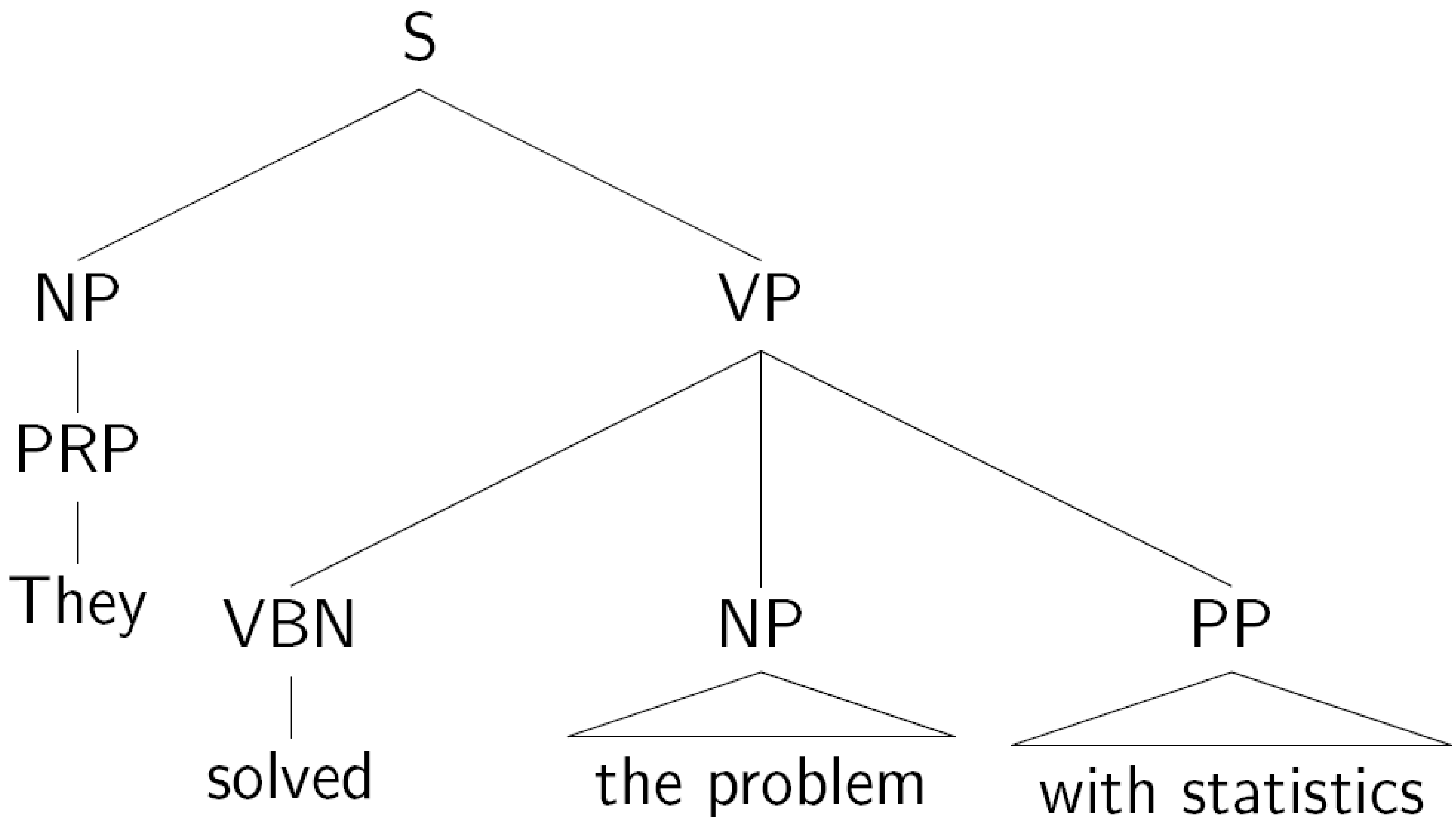
$P_1 = \{$

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

$\}$

Grammars for NLP_2

Parse tree

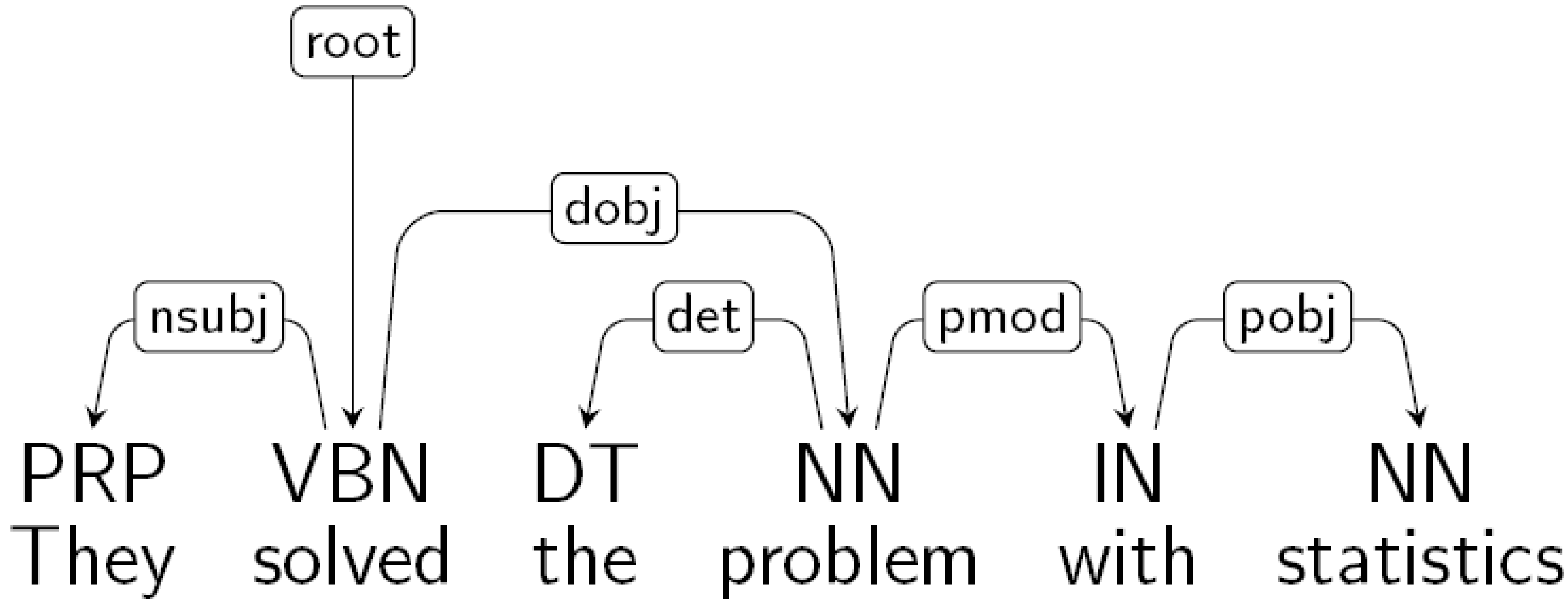


Grammars for NLP_3

- Forms of expressing syntactic structure:
  - Constituent or Phrase Structure (derivation tree = parse tree)
  - Dependency Structure
  - Tree adjoining grammars (TAG)
  - Transformational grammars
  - Systemic grammars
  - Logical Form

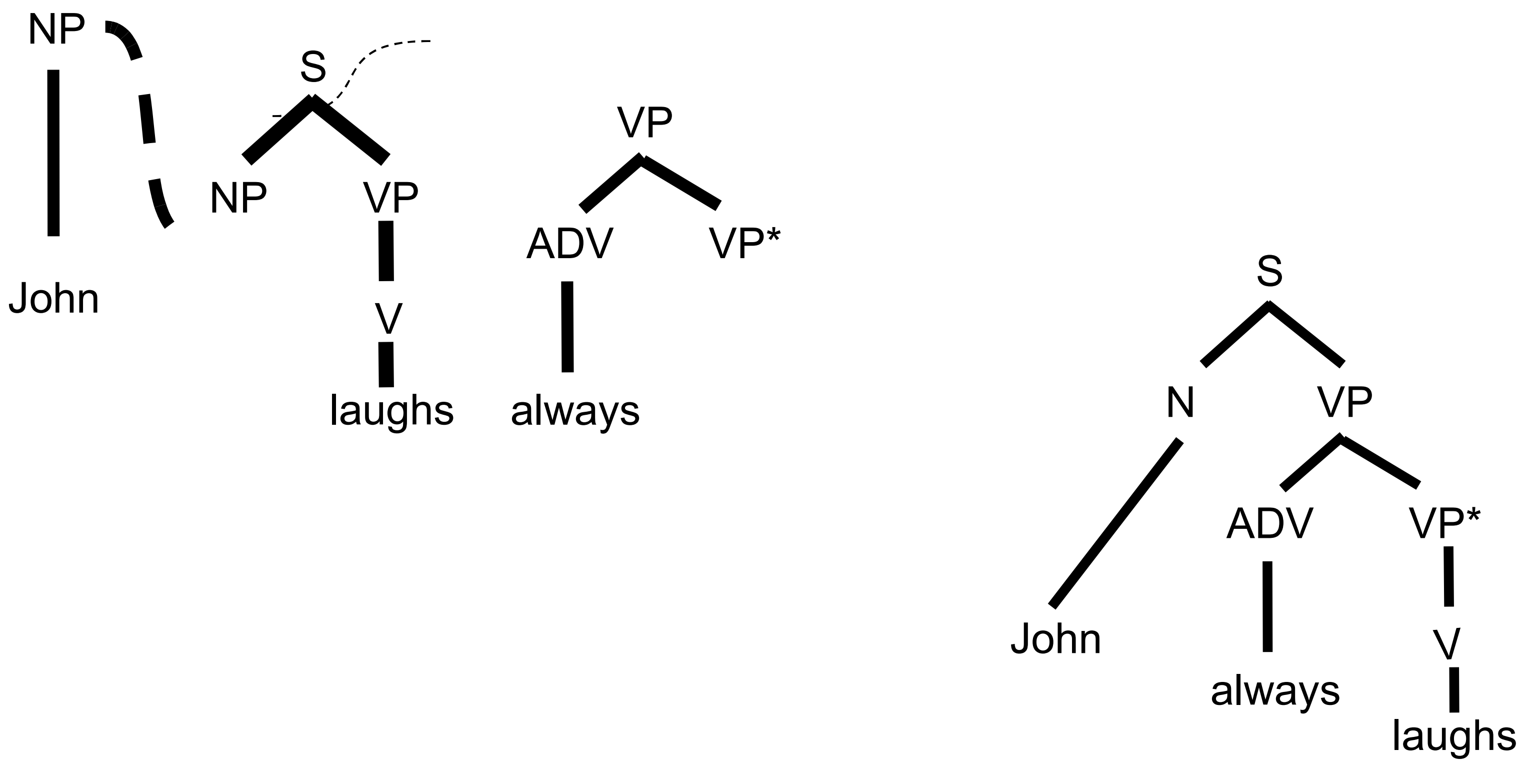
Grammars for NLP_4

Dependency tree



Grammars for NLP_5

Tree Adjoining Grammars (TAG)



Grammars for NLP$_6$

Logical Form (close to semantics)

The cat eats fish

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

Grammars for NLP$_7$

Expressivity of the grammar

- Minimum: Context free grammar(CFG)
- ¿Is NL context free?
- ¿Sufficient? NO (usually)
- Solution

⇒ CFG + {procedural addition of context}

⇒ Middly context sensitive grammars

⇒ Logic Grammars

Grammars for NLP_8

- For practical application in NLP
  - 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 $O(n^3)$
  - Middly context sensitive grammars
    - Tree Adjoining Grammars (TAG)
    - Linear Context Free Rewriting Systems

Grammars for NLP$_9$

Obtaining the grammar

- Definition of **T** from the tagset
- Definition of **V**
- Grammar rules **P**
  - Manually
  - Automatically
    - Grammatical inference
  - Semi- automatically

Grammars for NLP_{10}

Improving the grammar

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

Grammars for NLP$_{11}$

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