

# Computational Lexicology, Morphology and Syntax

Diana Trandabăț

Academic year 2022-2023

The sentence as a string of words

I saw the lady with the binoculars

**a b c d e b f**

string = a b c d e b f

The relations of parts of the string to each other may be different

I saw the lady with the binoculars

is structurally ambiguous

Who has the binoculars?

[ I ] saw the lady [ with the binoculars ]

= [ a ] b c d [ e b f ]

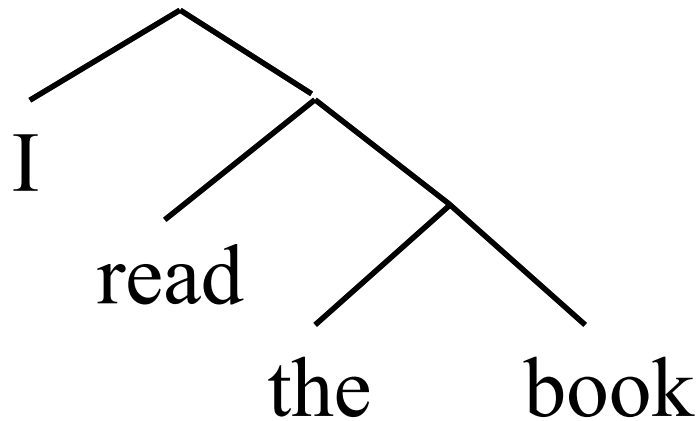
I saw [ the lady with the binoculars ]

= a b [ c d e b f ]

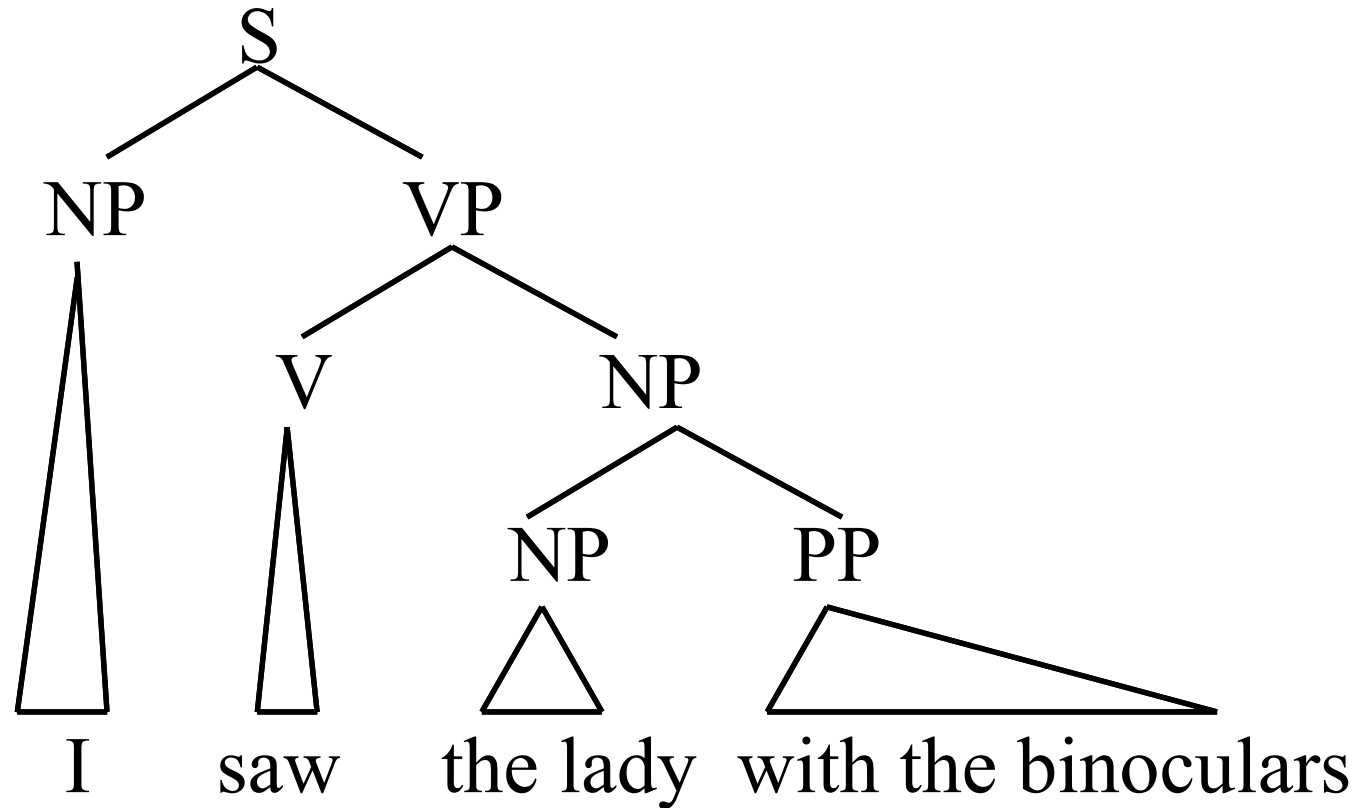
How can we represent the difference?

By assigning them different structures.

We can represent structures with 'trees'.

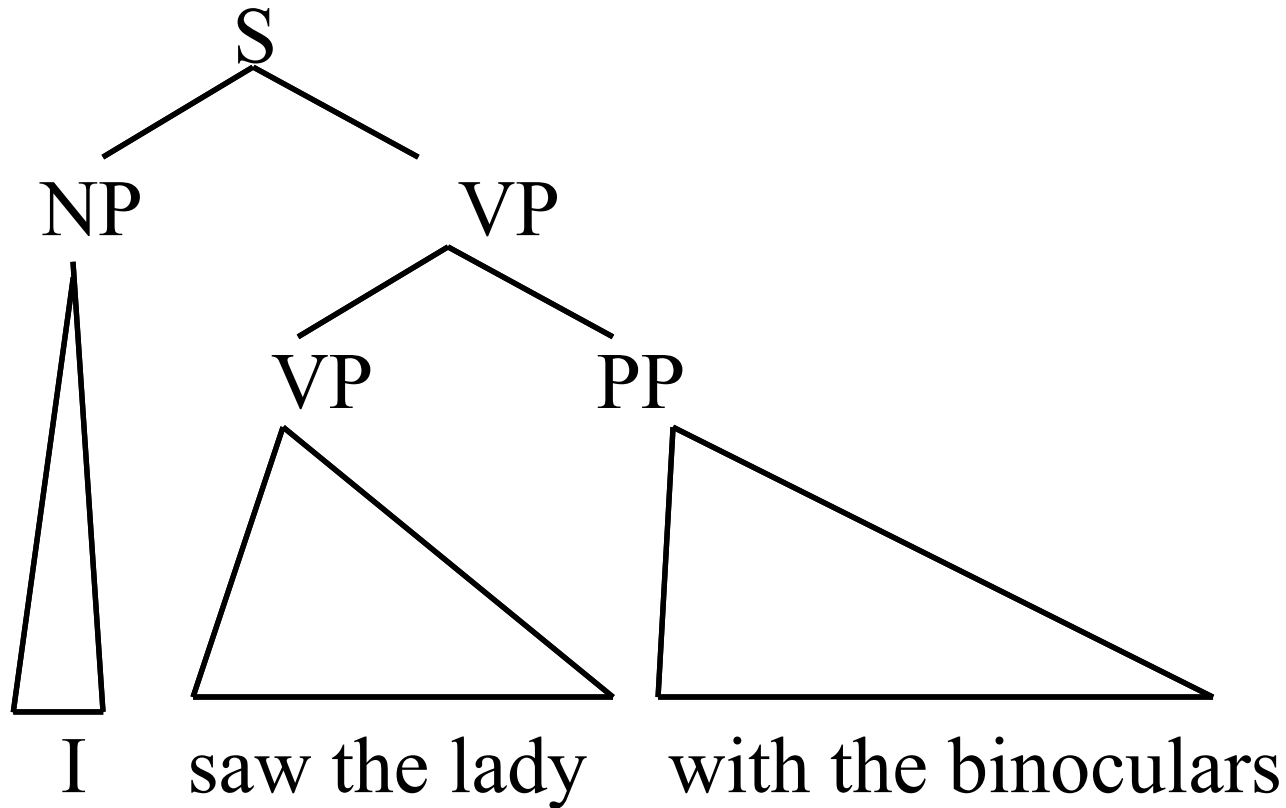


a. I saw the lady with the binoculars

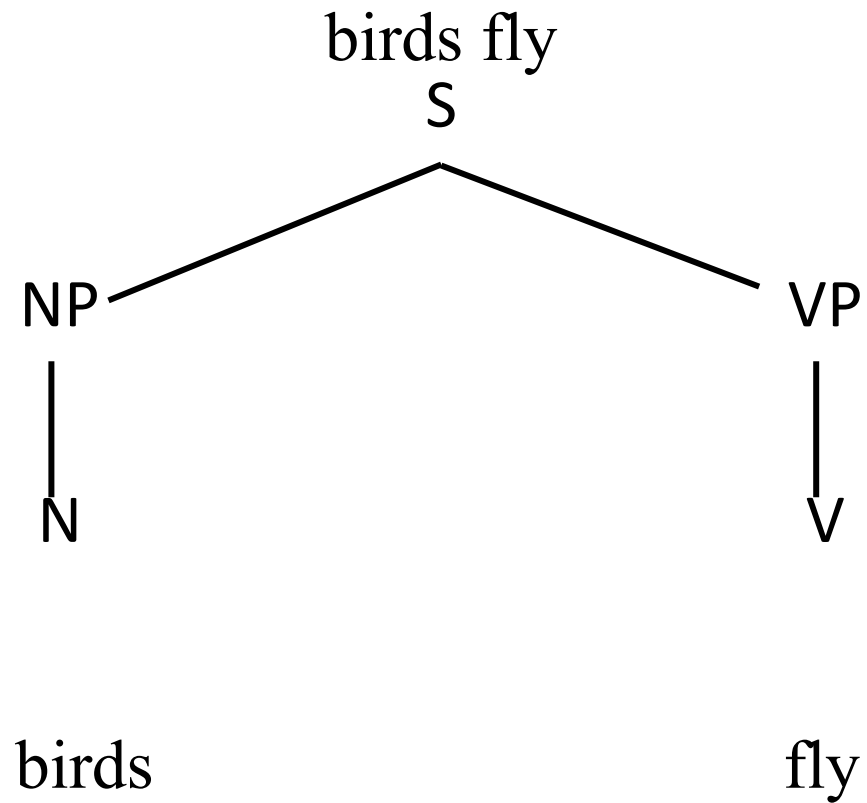


I saw [the lady with the binoculars]

b. I saw the lady with the binoculars



I [ saw the lady ] with the binoculars



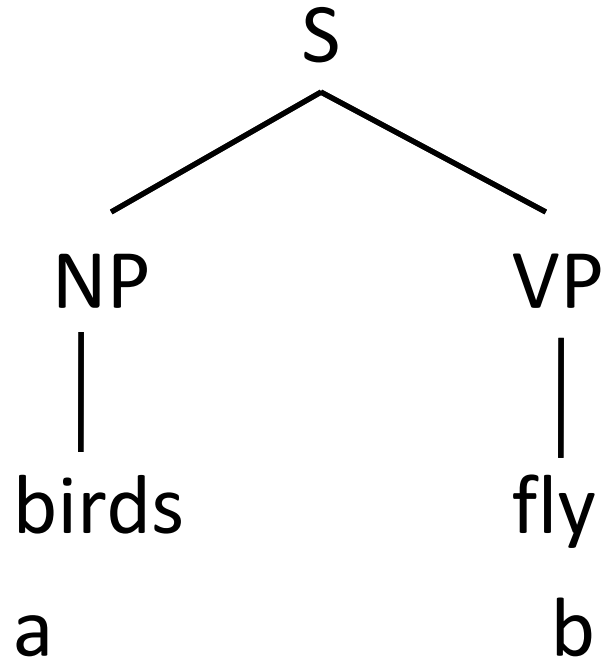
Syntactic rules

$S \rightarrow NP \quad VP$

$NP \rightarrow N$

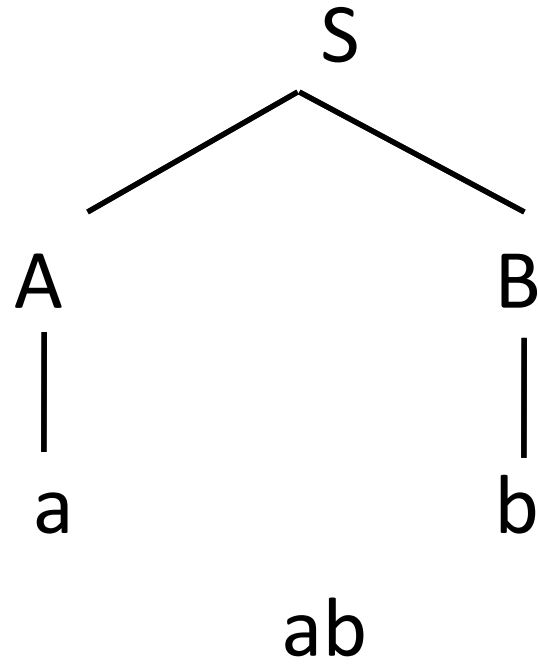
$VP \rightarrow V$





ab

= string



$S \rightarrow A B$

$A \rightarrow a$

$B \rightarrow b$

# Rules

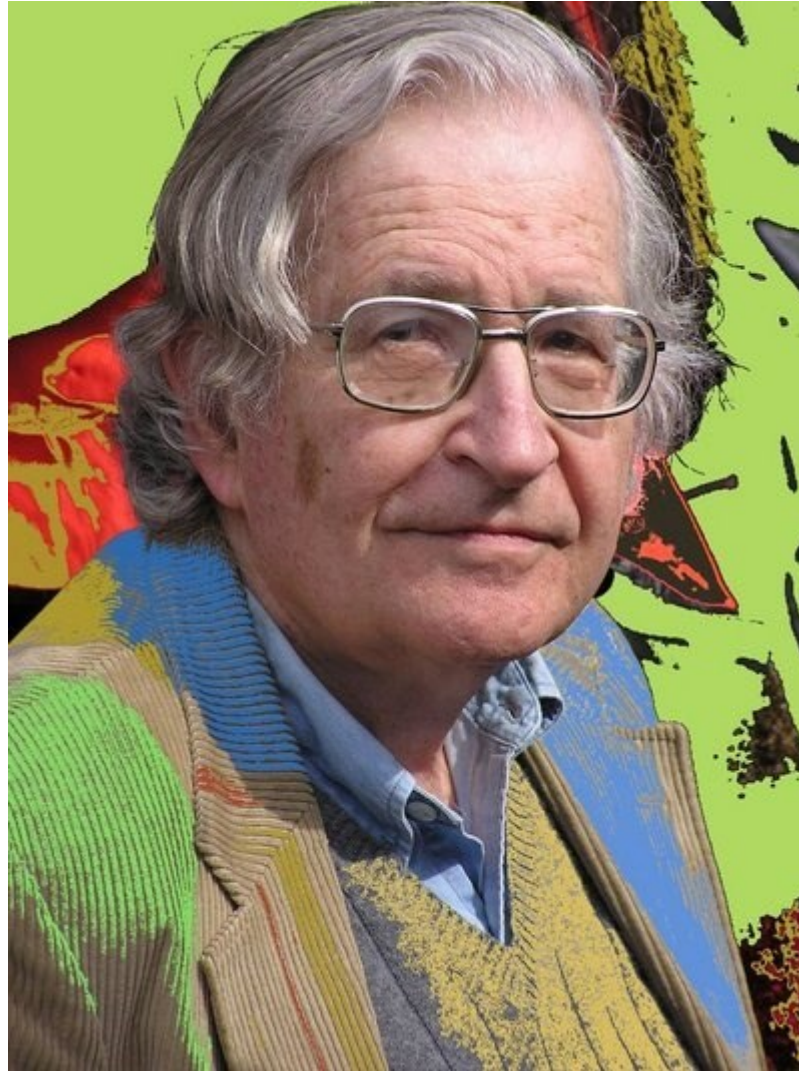
Assumption:

natural language grammars are rule-based systems

What kind of grammars describe natural language phenomena?

What are the formal properties of grammatical rules?

# The Chomsky Hierarchy



Chomsky (1957) *Syntactic Structures*. The Hague: Mouton

Chomsky, N. and G.A. Miller (1958) Finite-state languages *Information and Control* 1, 99-112

Chomsky (1959) On certain formal properties of languages. *Information and Control* 2, 137-167

# Rules in Linguistics

## 1. PHONOLOGY

$$/s/ \rightarrow [\theta] / V \_\_\_\_\_\_ V$$

Rewrite /s/ as [θ] when /s/ occurs in context V \_\_\_\_\_ V

With:

V = auxiliary node

s, θ = terminal nodes

# Rules in Linguistics

## 2. SYNTAX

$$S \rightarrow NP VP$$
$$VP \rightarrow v$$
$$NP \rightarrow n$$

Rewrite  $S$  as  $NP VP$  in any context

With:

$$S, NP, VP = \text{auxiliary nodes}$$
$$v, n = \text{terminal node}$$

# SYNTAX (phrase/sentence formation)

SENTENCE:

The boy

kissed the girl

SUBJECT

PREDICATE

NOUN PHRASE

VERB PHRASE

ART + NOUN

VERB + NOUN PHRASE

S → NP VP

VP → V NP

NP → ART N



# Chomsky Hierarchy

0. Type 0 (recursively enumerable) languages

Only restriction on rules: left-hand side cannot be the empty string (\*  $\emptyset \rightarrow \dots\dots$ )

1. Context-Sensitive languages - Context-Sensitive (CS) rules

2. Context-Free languages - Context-Free (CF) rules

3. Regular languages - Non-Context-Free (CF) rules

$0 \supseteq 1 \supseteq 2 \supseteq 3$

$a \supseteq b$  meaning  $a$  properly includes  $b$  ( $a$  is a superset of  $b$ ),  
i.e.  $b$  is a proper subset of  $a$  or  $b$  is in  $a$

# Generative power

## 0. Type 0 (recursively enumerable) languages

- only restriction on rules: left-hand side cannot be the empty string (\*  $\emptyset \rightarrow \dots\dots$ )
- is the most powerful system

## 3. Type 3 (regular language)

- is the least powerful

# Grammars with rule type 3

Name: Regular

Example: Finite State Automata (Markov-process Grammar)

Rule type:

a) right-linear

$$A \rightarrow xB \text{ or}$$

$$A \rightarrow x$$

with:

$A, B$  = auxiliary nodes and

$x$  = terminal node

b) or left-linear

$$A \rightarrow Bx \text{ or}$$

$$A \rightarrow x$$

Generates:  $a^m b^n$  with  $m, n \geq 1$

Cannot guarantee that there are as many a's as b's; no embedding

## A regular grammar for natural language sentences

$S \rightarrow \text{the } A$

$A \rightarrow \text{cat } B$

$A \rightarrow \text{mouse } B$

$A \rightarrow \text{duck } B$

$B \rightarrow \text{bites } C$

$B \rightarrow \text{sees } C$

$B \rightarrow \text{eats } C$

$C \rightarrow \text{the } D$

$D \rightarrow \text{boy}$

$D \rightarrow \text{girl}$

$D \rightarrow \text{monkey}$

the cat bites the boy

the mouse eats the monkey

the duck sees the girl

## Regular grammars... or not?

Grammar 1:

$A \rightarrow a$

$A \rightarrow a B$

$B \rightarrow b A$

Grammar 2:

$A \rightarrow a$

$A \rightarrow B a$

$B \rightarrow A b$

Grammar 3:

$A \rightarrow a$

$A \rightarrow a B$

$B \rightarrow b$

$B \rightarrow b A$

Grammar 4:

$A \rightarrow a$

$A \rightarrow B a$

$B \rightarrow b$

$B \rightarrow A b$

Grammar 5:

$S \rightarrow A B$

$S \rightarrow b B$

$A \rightarrow a S$

$B \rightarrow b b S$

$S \rightarrow \varepsilon$

Grammar 6:

$A \rightarrow a$

$A \rightarrow B a$

$B \rightarrow b$

$B \rightarrow b A$

$A \rightarrow a$

# Grammars: non-regular

Grammar 5:

$S \rightarrow A B$

$S \rightarrow b B$

$A \rightarrow a S$

$B \rightarrow b b S$

$S \rightarrow \varepsilon$

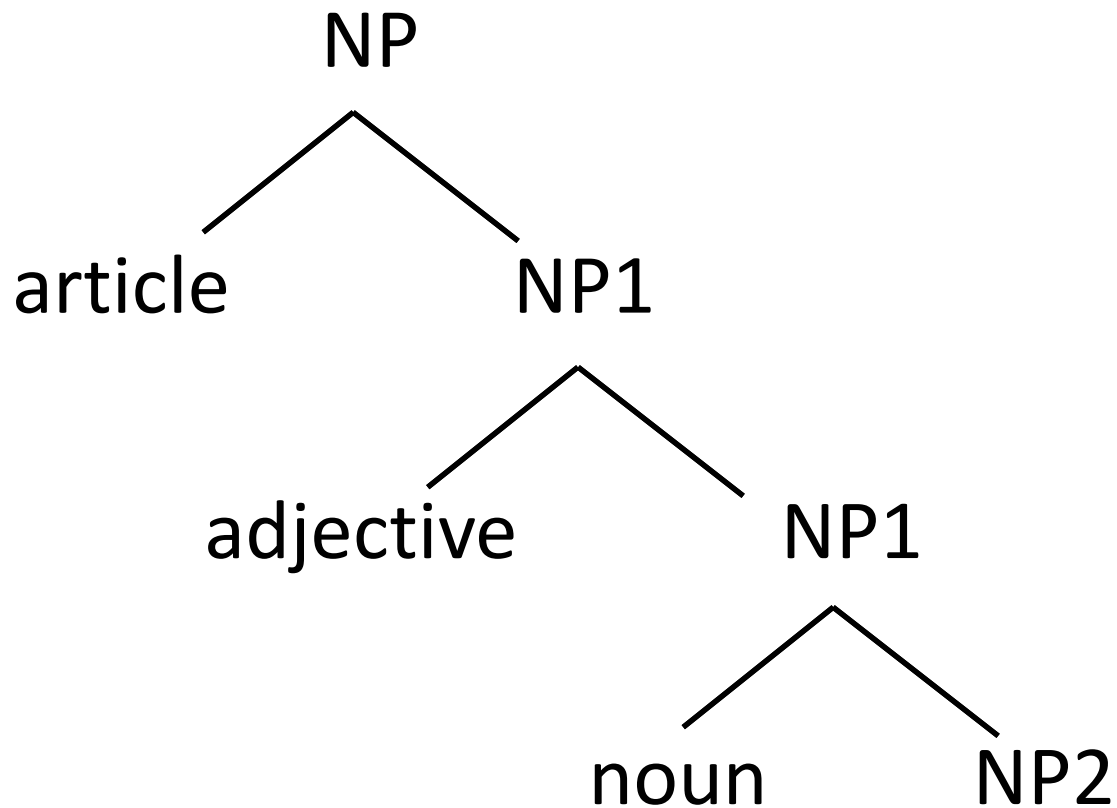
Grammar 6:

$A \rightarrow a$

$A \rightarrow B a$

$B \rightarrow b$

$B \rightarrow b A$

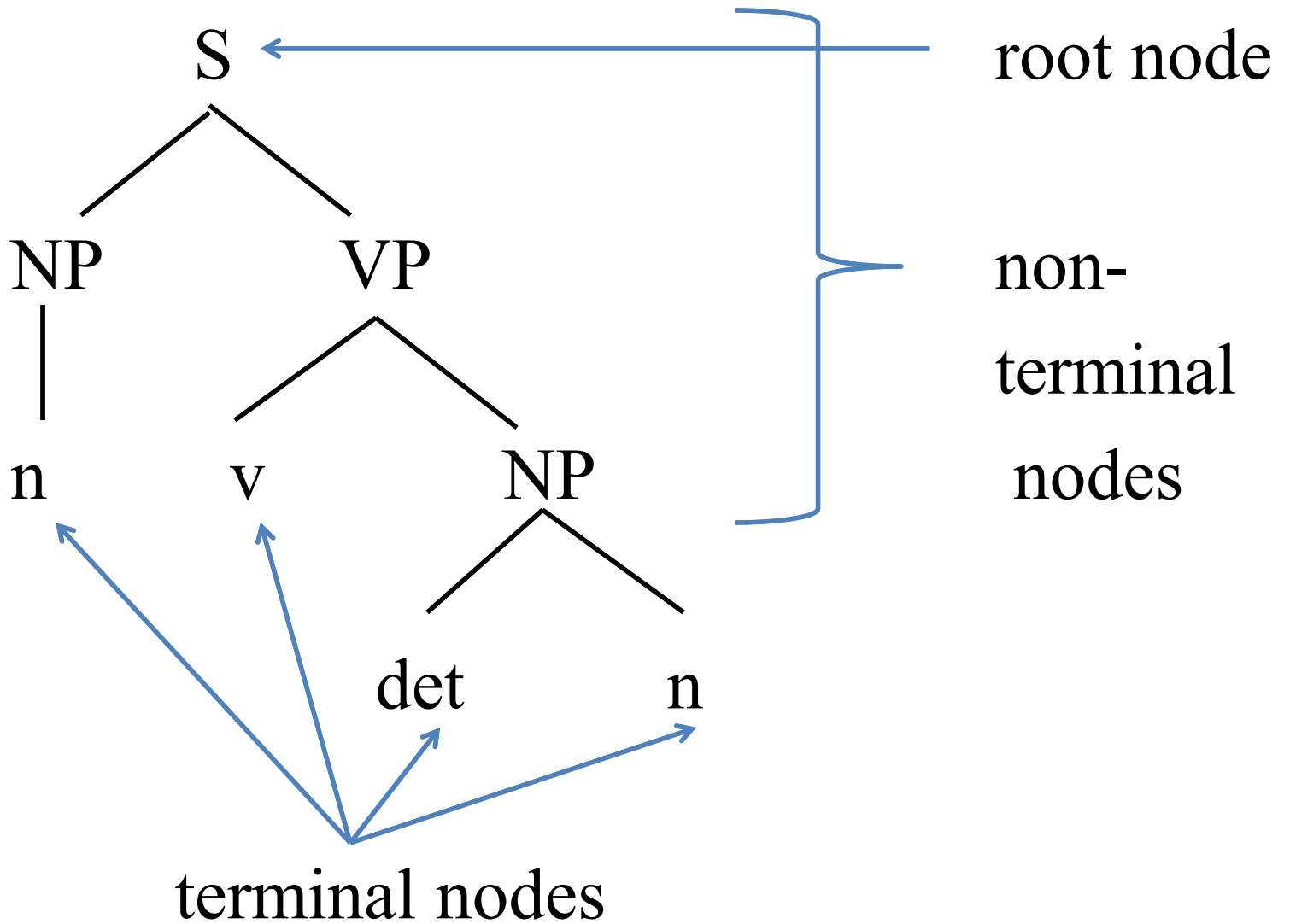


NP → article NP1

NP1 → adjective NP1

NP1 → noun NP2

# A parse tree





# Grammars with rule type 2

Name: Context Free

Example:

Phrase Structure Grammars/  
Push-Down Automata

Rule type:

$$A \rightarrow \gamma$$

with:

$A$  = auxiliary node

$\gamma$  = any number of terminal or auxiliary nodes

Recursiveness (centre embedding) allowed:

$$A \rightarrow \gamma A \chi$$

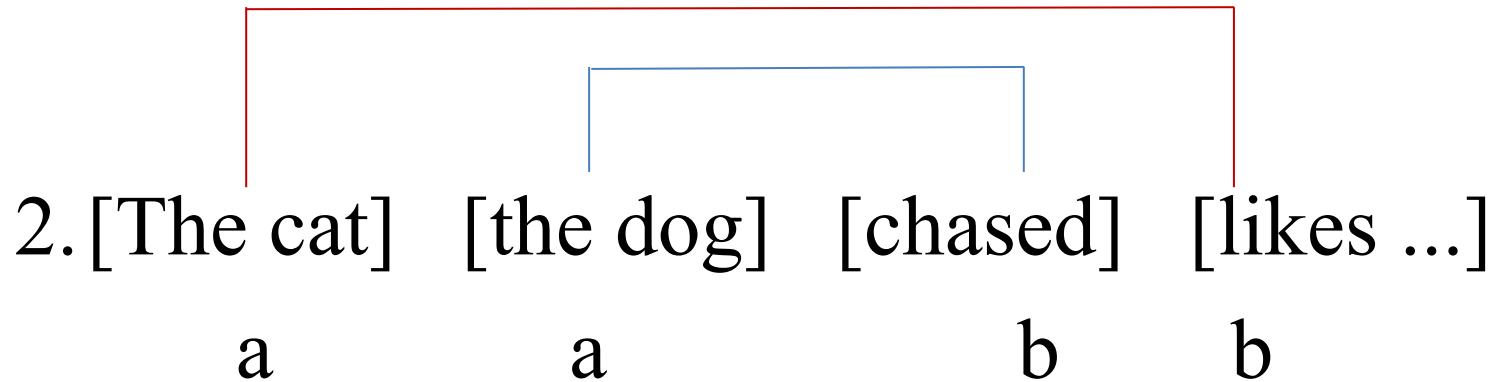
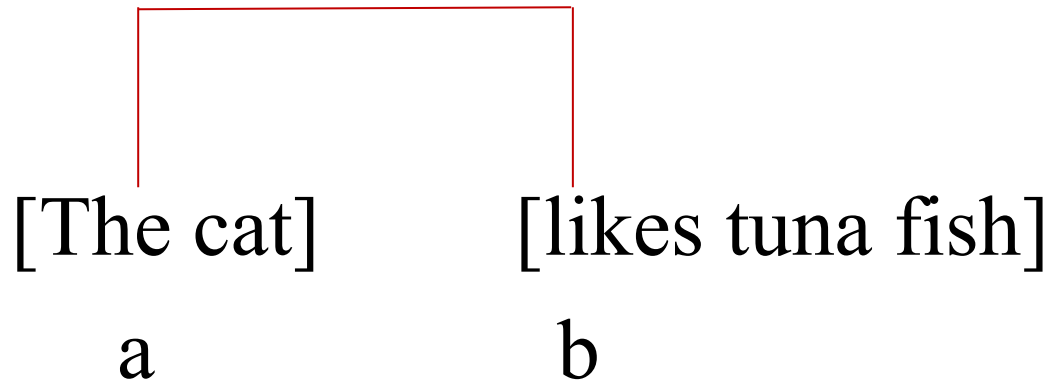
## Grammars with rule type 2

A Context Free grammar consists of:

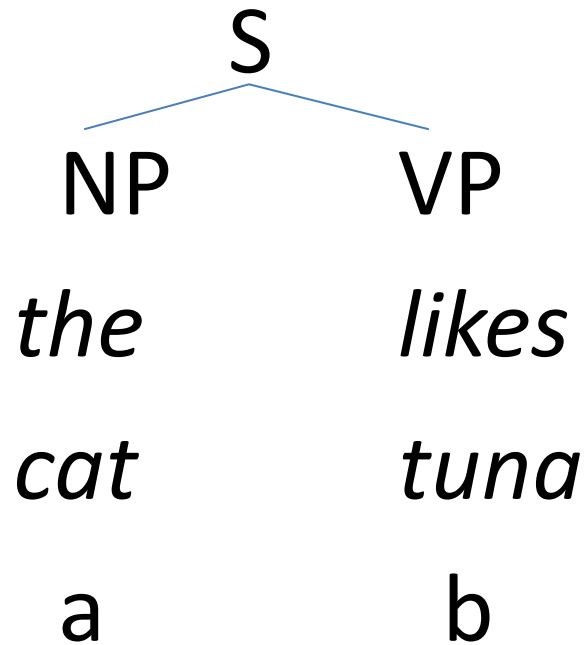
- a) a finite terminal vocabulary  $V_T$
- b) a finite auxiliary vocabulary  $V_A$
- c) an axiom  $S \in V_A$
- d) a finite number of context free rules of  
form  $A \rightarrow \gamma$ ,  
where  $A \in V_A$   
and  $\gamma \in \{V_A \cup V_T\}^*$

In natural language syntax  $S$  is interpreted as the start symbol for sentence, as in  $S \rightarrow NP VP$



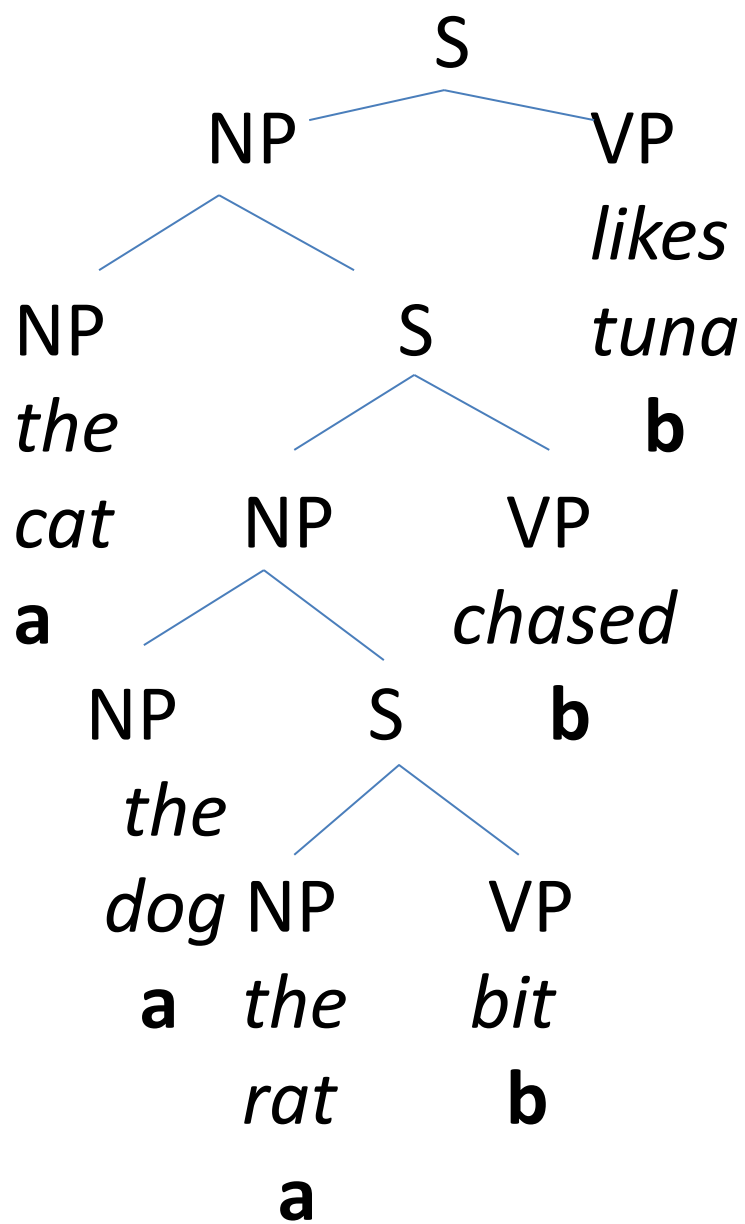


# Centre embedding



= ab





= aaabbbb

## More natural centre embedding

1. If  $S_1$ , then  $S_2$   
a            a

2. Either  $S_3$ , or  $S_4$   
b            b

Sentence with embedding:

If either the man is arriving today or the woman is arriving tomorrow, then the child is arriving the day after.

a = [if

b = [either the man is arriving today]

b = [or the woman is arriving tomorrow]]

a = [then the child is arriving the day after]

= abba



# Grammars with rule type 1

Name: context sensitive

The following languages cannot be generated by a CF grammar (by pumping lemma):

$a^n b^m c^n d^m$

## Swiss German:

A string of dative nouns (e.g. aa), followed by a string of accusative nouns (e.g. bbb), followed by a string of dative-taking verbs (cc), followed by a string of accusative-taking verbs (ddd)

= aabbbccddd

=  $a^n b^m c^n d^m$

# Swiss German:

Jan sait das (Jan says that) ...

mer	em Hans	es Huus	hälfed	aastrische
we	Hans/DAT	the house/ACC	helped	paint
we helped Hans paint the house				

abcd

NPdat	NPdat	NPacc	NPacc	Vdat	Vdat	Vacc	Vacc
a	a	b	b	c	c	d	d

# Context Free Grammars (CFGs)

- Most used in NLP
- Sets of rules expressing how symbols of the language fit together, e.g.

S → NP VP

NP → Det N

Det → the

N → dog

# What Does Context Free Mean?

- LHS of rule is **just one** symbol.

- Can have

NP  $\rightarrow$  Det N

- **Cannot** have

(X) NP (Y)  $\rightarrow$  (X) Det N (Y)

# Grammar Symbols

- Non Terminal Symbols
- Terminal Symbols
  - Words
  - Preterminals

# Non Terminal Symbols

- Symbols which have definitions
- Symbols which appear on the LHS of rules

**S**       $\rightarrow$  NP VP

**NP**      $\rightarrow$  Det N

**Det**    $\rightarrow$  the

**N**       $\rightarrow$  dog

# Non Terminal Symbols

- Same Non Terminals can have several definitions

**S**       $\rightarrow$  NP VP

**NP**      $\rightarrow$  Det N

**NP**      $\rightarrow$  N

**Det**    $\rightarrow$  the

**N**       $\rightarrow$  dog

# Terminal Symbols

- Symbols which appear in final string
- Correspond to words
- Are not defined by the grammar

S → NP VP

NP → Det N

Det → **the**

N → **dog**



# Parts of Speech (POS)

- NT Symbols which produce terminal symbols are sometimes called **pre-terminals**

S → NP VP

NP → **Det N**

**Det** → the

**N** → dog

- Sometimes we are interested in the shape of sentences formed from pre-terminals

**Det N V**

**Aux N V D N**

# CFG - formal definition

A CFG is a tuple  $(N, \Sigma, R, S)$

- $N$  is a set of non-terminal symbols
- $\Sigma$  is a set of terminal symbols disjoint from  $N$
- $R$  is a set of rules each of the form  $A \rightarrow \beta$  where  $A$  is non-terminal
- $S$  is a designated start-symbol

# CFG - Example

## **grammar:**

S → NP VP

NP → N

VP → V NP

## **lexicon:**

V → kicks

N → John

N → Bill

**N** = {S, NP, VP, N, V}

**$\Sigma$**  = {kicks, John, Bill}

**R** = (see opposite)

**S** = "S"

# Exercise

- Write grammars that generate the following languages, for  $m > 0$

$(ab)^m$

$a^n b^m$

$a^n b^n$

- Which of these are Regular?
- Which of these are Context Free?

$(ab)^m$  for  $m > 0$

$S \rightarrow a b$

$S \rightarrow a b S$

$(ab)^m$  for  $m > 0$

$S \rightarrow a b$

$S \rightarrow a b S$

$S \rightarrow a X$

$X \rightarrow b Y$

$Y \rightarrow a b$

$Y \rightarrow S$

$a^n b^m$

$S \rightarrow A B$

$A \rightarrow a$

$A \rightarrow a A$

$B \rightarrow b$

$B \rightarrow b B$

$a^n b^m$

$S \rightarrow A B$

$A \rightarrow a$

$A \rightarrow a A$

$B \rightarrow b$

$B \rightarrow b B$

$S \rightarrow a AB$

$AB \rightarrow a AB$

$AB \rightarrow B$

$B \rightarrow b$

$B \rightarrow b B$



# Grammar Defines a Structure

## **grammar:**

S → NP VP

NP → N

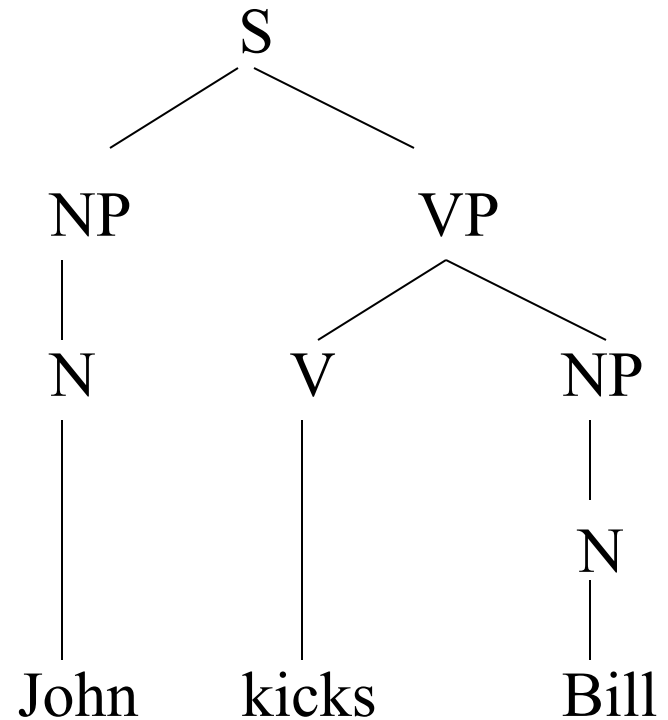
VP → V NP

## **lexicon:**

V → kicks

N → John

N → Bill



# Different Grammar Different Structure

## **grammar:**

S → NP NP

NP → N V

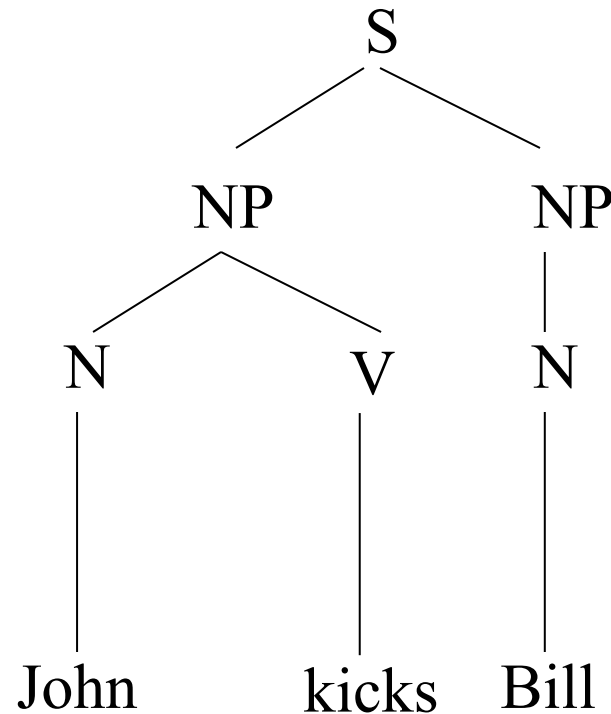
NP → N

## **lexicon:**

V → kicks

N → John

N → Bill



# Which Grammar is Best?

- The structure assigned by the grammar should be *appropriate*.
- The structure should
  - Be understandable
  - Allow us to make generalisations.
  - Reflect the underlying meaning of the sentence.

# Ambiguity

- A grammar is ambiguous if it assigns two or more structures to the same sentence.

NP → NP CONJ NP

NP → N

## **lexicon:**

CONJ → and

N → John

N → Bill

- The grammar should not generate too many possible structures for the same sentence.

# Criteria for Evaluating Grammars

- Does it undergenerate?
- Does it overgenerate?
- Does it assign appropriate structures to sentences it generates?
- Is it simple to understand? How many rules are there?
- Does it contain just a few generalisations or is it full of special cases?
- How ambiguous is it? How many structures does it assign for a given sentence?

# Probabilistic Context Free Grammar (PCFG)

- A PCFG is a probabilistic version of a CFG where each production has a probability.
- String generation is now probabilistic where production probabilities are used to non-deterministically select a production for rewriting a given non-terminal.

# Characteristics of PCFGs

- In a PCFG, the probability  $P(A \rightarrow \beta)$  expresses the likelihood that the non-terminal  $A$  will expand as  $\beta$ .
  - e.g. the likelihood that  $S \rightarrow NP VP$ 
    - (as opposed to  $S \rightarrow VP$ , or  $S \rightarrow NP VP PP$ , or... )
- can be interpreted as a conditional probability:
  - probability of the expansion, given the LHS non-terminal
  - $P(A \rightarrow \beta) = P(A \rightarrow \beta | A)$
- Therefore, for any non-terminal  $A$ , probabilities of every rule of the form  $A \rightarrow \beta$  must sum to 1
  - If this is the case, we say the PCFG is **consistent**

# Simple PCFG for English

Grammar	Prob	Lexicon
S → NP VP	0.8	
S → Aux NP VP	0.1	Det → the   a   that   this
S → VP	0.1	0.6 0.2 0.1 0.1
NP → Pronoun	0.2	Noun → book   flight   meal   money
NP → Proper-Noun	0.2	0.1 0.5 0.2 0.2
NP → Det Nominal	0.6	Verb → book   include   prefer
Nominal → Noun	0.3	0.5 0.2 0.3
Nominal → Nominal Noun	0.2	Pronoun → I   he   she   me
Nominal → Nominal PP	0.5	0.5 0.1 0.1 0.3
VP → Verb	0.2	Proper-Noun → Houston   NWA
VP → Verb NP	0.5	0.8 0.2
VP → VP PP	0.3	Aux → does
PP → Prep NP	1.0	1.0
		Prep → from   to   on   near   through
		0.25 0.25 0.1 0.2 0.2



# Parse tree and Sentence Probability

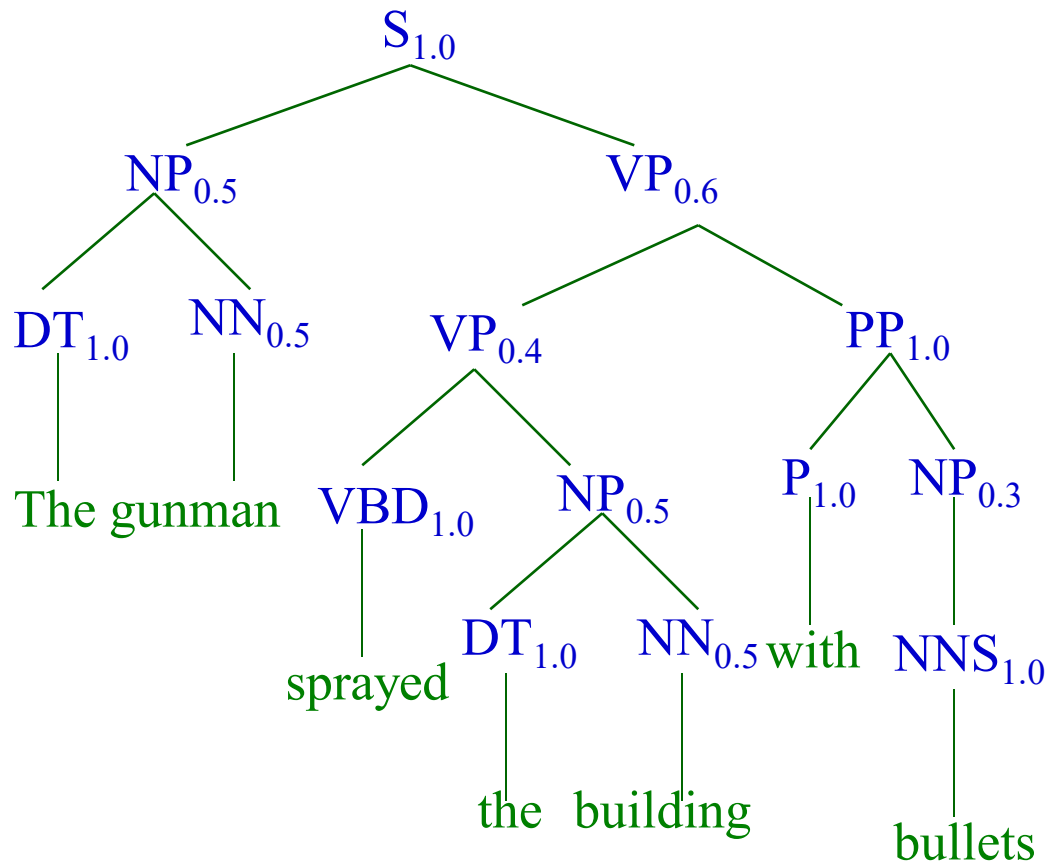
- Assume productions for each node are chosen independently.
- Probability of a parse tree (derivation) is the product of the probabilities of its productions.
- Resolve ambiguity by picking most probable parse tree.
- Probability of a sentence is the sum of the probabilities of all of its derivations.

# Example PCFG Rules & Probabilities

- $S \rightarrow NP VP$  1.0
- $NP \rightarrow DT NN$  0.5
- $NP \rightarrow NNS$  0.3
- $NP \rightarrow NP PP$  0.2
- $PP \rightarrow P NP$  1.0
- $VP \rightarrow VP PP$  0.6
- $VP \rightarrow VBD NP$  0.4
- $DT \rightarrow the$  1.0
- $NN \rightarrow gunman$  0.5
- $NN \rightarrow building$  0.5
- $VBD \rightarrow sprayed$  1.0
- $NNS \rightarrow bullets$  1.0
- $P \rightarrow with$  1.0

# Example Parse $t_1$

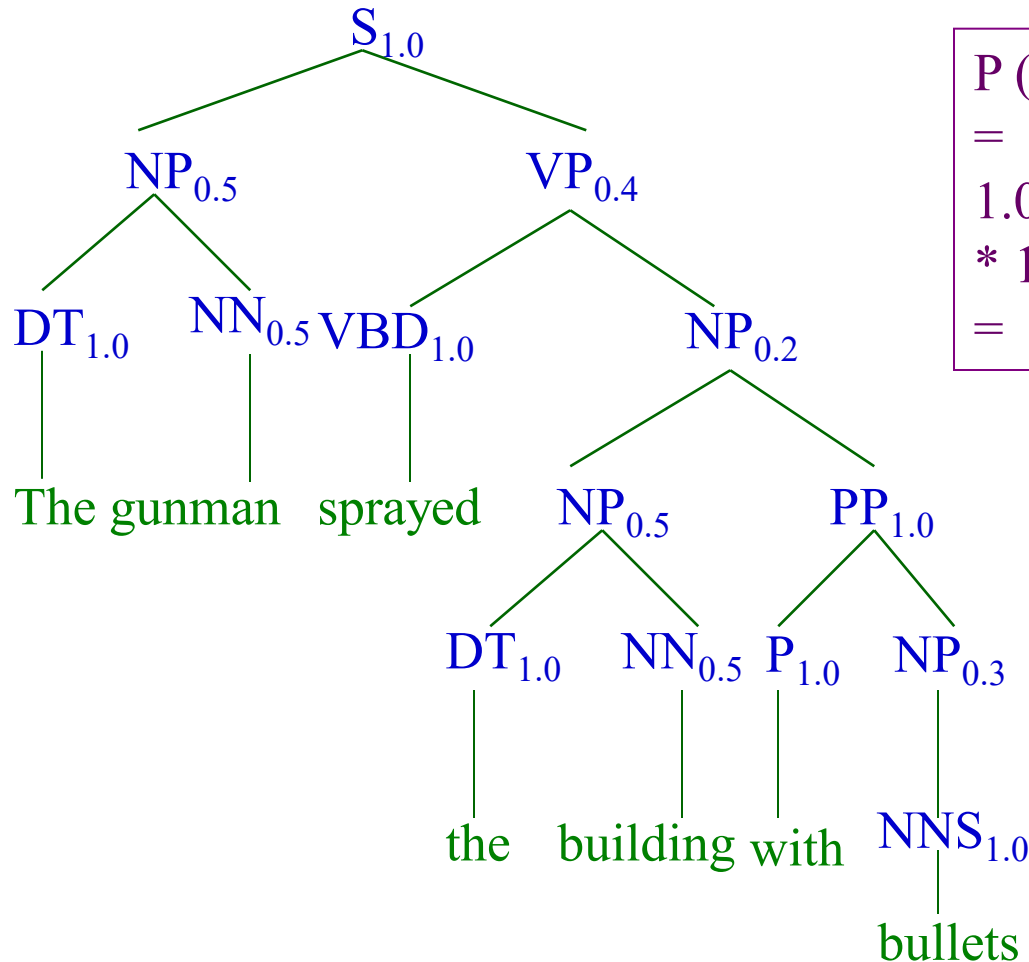
- The gunman sprayed the building with bullets.



$$\begin{aligned} P(t_1) &= 1.0 * 0.5 * 1.0 * 0.5 * 0.6 * \\ &0.4 * 1.0 * 0.5 * 1.0 * 0.5 * 1.0 \\ &* 1.0 * 0.3 * 1.0 \\ &= 0.00225 \end{aligned}$$

# Another Parse $t_2$

- The gunman sprayed the building with bullets.



$$\begin{aligned} P(t_2) &= 1.0 * 0.5 * 1.0 * 0.5 * 0.4 * \\ & 1.0 * 0.2 * 0.5 * 1.0 * 0.5 * 1.0 \\ & * 1.0 * 0.3 * 1.0 \\ &= 0.0015 \end{aligned}$$

$$\begin{aligned} P(\text{sentence}) &= \\ &= P(t_1) + P(t_2) = \\ &= 0.00225 + 0.0015 = \\ & 0.00375 \end{aligned}$$

# NLP with Regular Expressions

Transcript with Eliza, simulation of a Rogerian psychotherapist  
(Weizenbaum, 1966)

User: Men are all alike

ELIZA: IN WHAT WAY?

User: They're always bugging us about something or other

ELIZA: CAN YOU THINK OF A SPECIFIC EXAMPLE?

User: Well, my boyfriend made me come here

ELIZA: YOUR BOYFRIEND MADE YOU COME HERE.

User: He says I'm depressed much of the time

ELIZA: I AM SORRY TO HEAR THAT YOU ARE DEPRESSED.

# How did it work?

- .\* all .\*  
→ IN WHAT WAY
- .\* always .\*  
→ CAN YOU THINK OF A SPECIFIC EXAMPLE
- .\* I'm (depressed | sad) .\*  
→ I AM SORRY TO HEAR YOU ARE \1
- .\* I'm (depressed | sad) .\*  
→ WHY DO YOU THINK YOU ARE \1?

# Aside...

- What is intelligence?
- What does Eliza tell us about intelligence?

Great!

See you next time!



# PATR-II

- The PATR-II formalism can be viewed as a computer language for encoding linguistic information.
- A PATR-II grammar consists of a set of rules and a lexicon.
- The rules are CFG rules augmented with constraints.
- The lexicon provides information about terminal symbols.

# Example PATR-II

## Grammar and Lexicon

### Grammar (grammar.grm)

Rule

s -> np vp

Rule

np -> n

Rule

vp -> v

### Lexicon (lexicon.lex)

\w uther

\c n

\w sleeps

\c v