Department of
Linguistics and Translation
香港城市大學
City University of Hong Kong

Computational Linguistics
LT3233

Jixing Li

Lecture 4: Context-Free Grammars

Slides adapted from Edward Stabler, Michael Collins

Lecture plan

- Syntax 101
- What are context-free grammars?
- Top-down recognition
- Short break (15 mins)
- Hands-on exercises

N-grams review

Bigram model: Condition on the previous word:

$$\mathbf{P}(w_1, w_2, w_3, \ldots w_n) \approx \prod_{i=1}^{n} \mathbf{P}(w_i | w_{i-1})$$

P(*a pile of shaving cream***) ≈**

P(*a***) x P(***pile|a***) x P(***of|pile***) x P(***shaving|of***) x P(***cream|shaving***)**

MLE ↓

$$\frac{\mathbf{Count}(a)}{\#\ \mathbf{of\ words}} \quad \frac{\mathbf{Count}(a\ pile)}{\mathbf{Count}(a)} \quad \frac{\mathbf{Count}(pile\ of)}{\mathbf{Count}(pile)} \quad \frac{\mathbf{Count}(of\ shaving)}{\mathbf{Count}(of)} \quad \frac{\mathbf{Count}(shaving\ cream)}{\mathbf{Count}(shaving)}$$

→ Can extend to trigrams, 4-grams, 5-grams, etc.

Languages are not Markovian

N-gram models are insufficient models of language

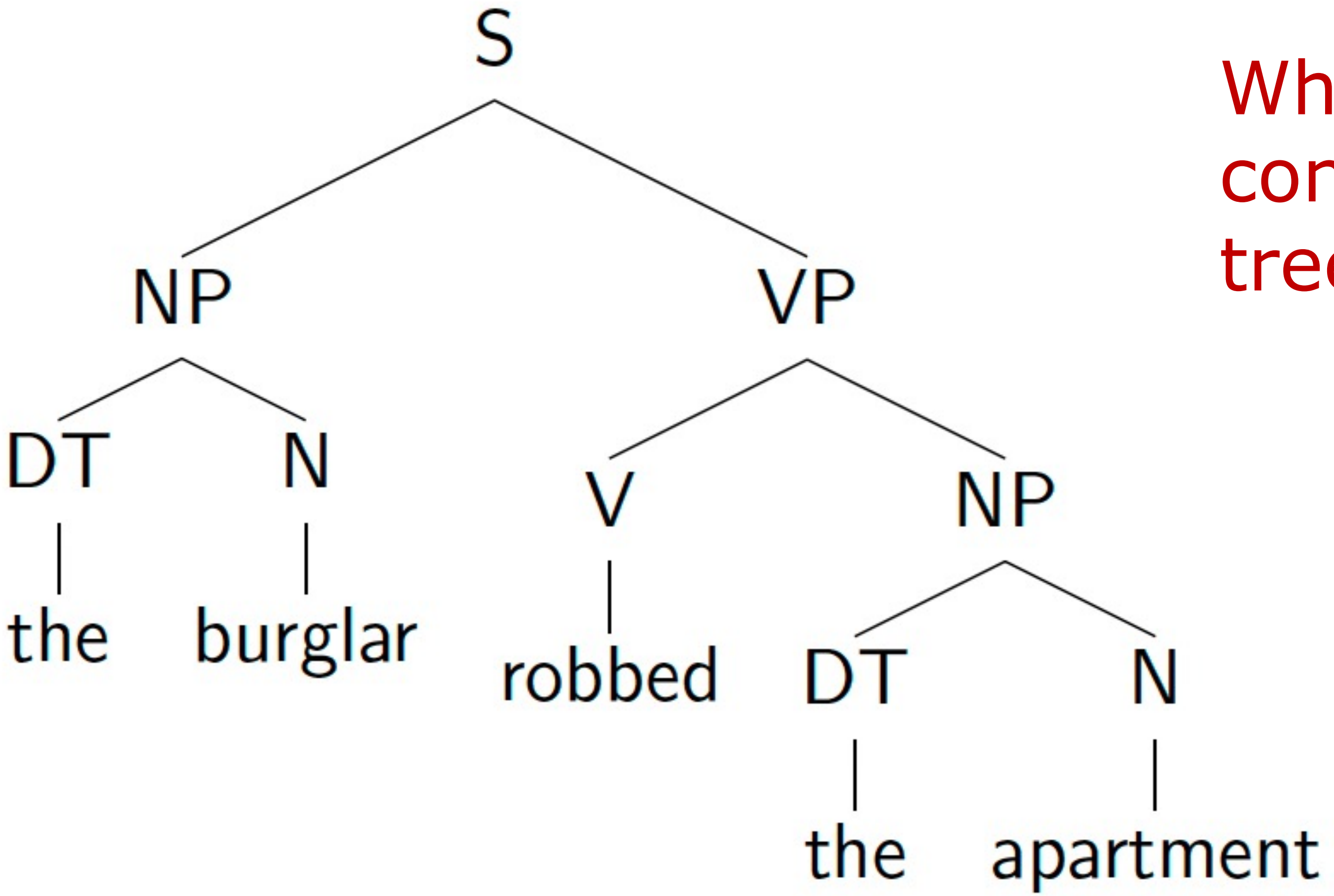
Bulldogs bulldogs bulldogs fight fight fight.

The bulldogs that are fought by the bulldogs that the bulldogs fight, fight.

→ **long-distance dependencies**

Syntactic structure

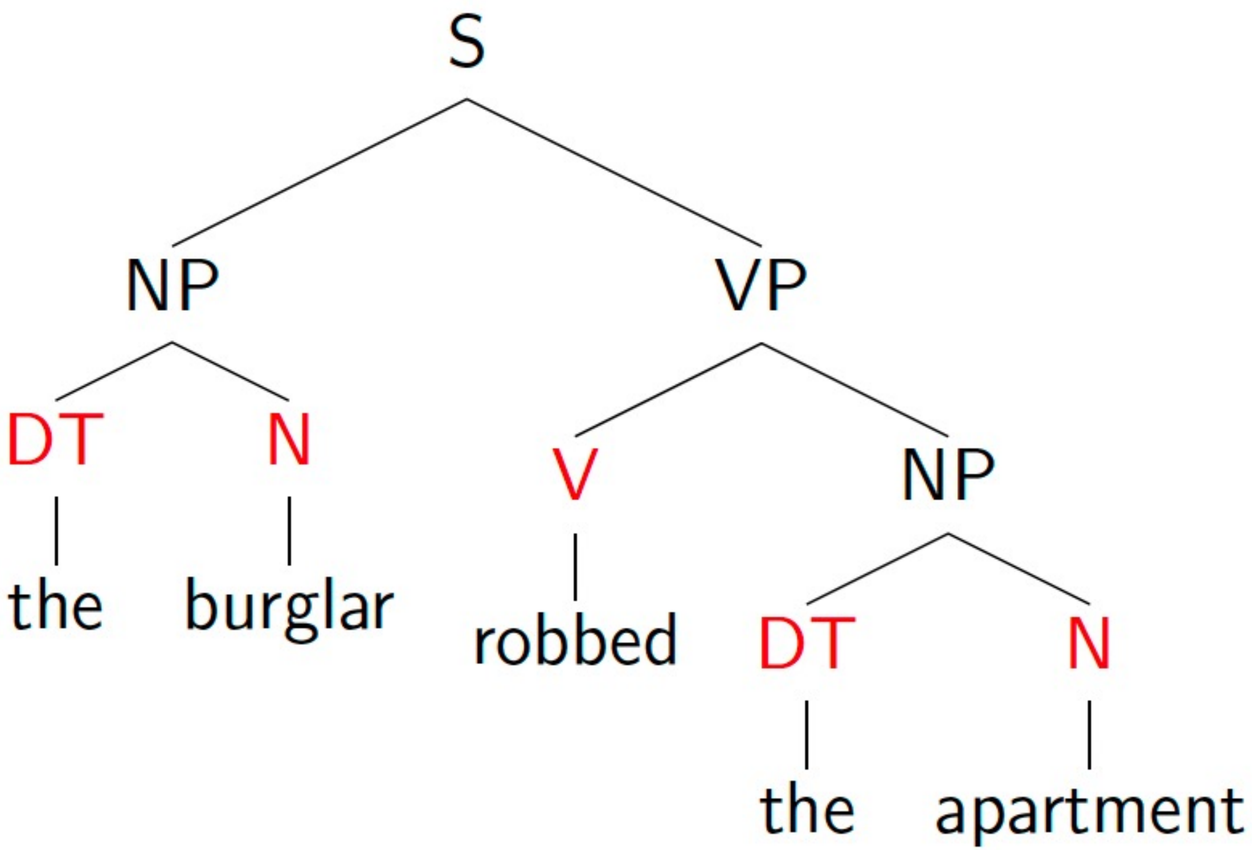
"the burglar robbed the apartment"



What information is conveyed by a syntactic tree?

Trees visualize parts of speech

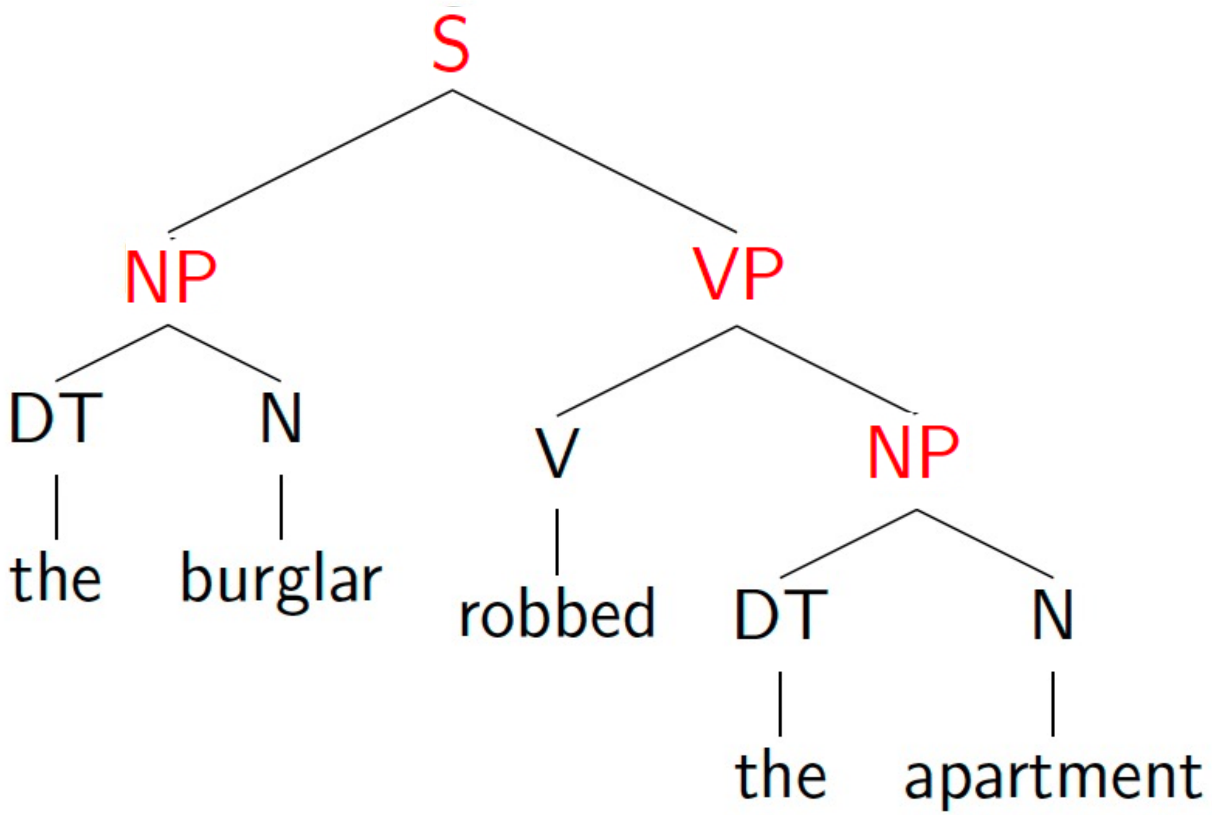
DT: determiner; N: noun; V: verb



DT → the
N → burglar | apartment
V → robbed

Trees visualize constituency

S: Sentence; NP: noun phrase; VP: verb phrase;



S → NP VP
NP → DT N
VP → V NP

Constituency

Example noun phrases:

Harry the Horse	a high-class spot such as Mindy's
the Broadway coppers	the reason he comes into the Hot Box
they	three parties from Brooklyn

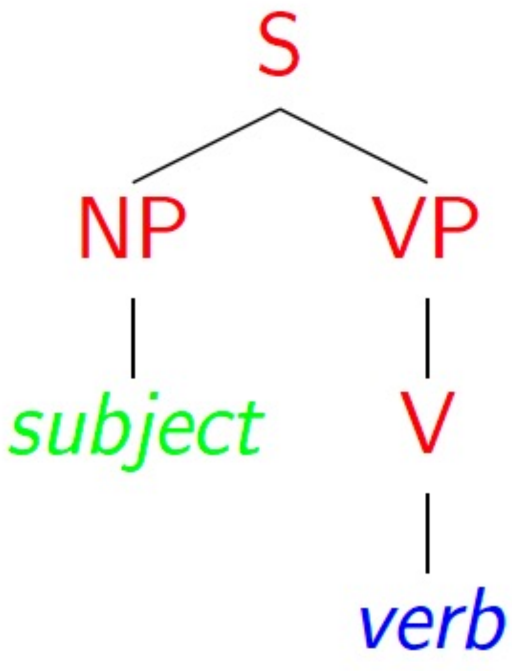
NPs appear in similar syntactic environments, i.e., before a verb:

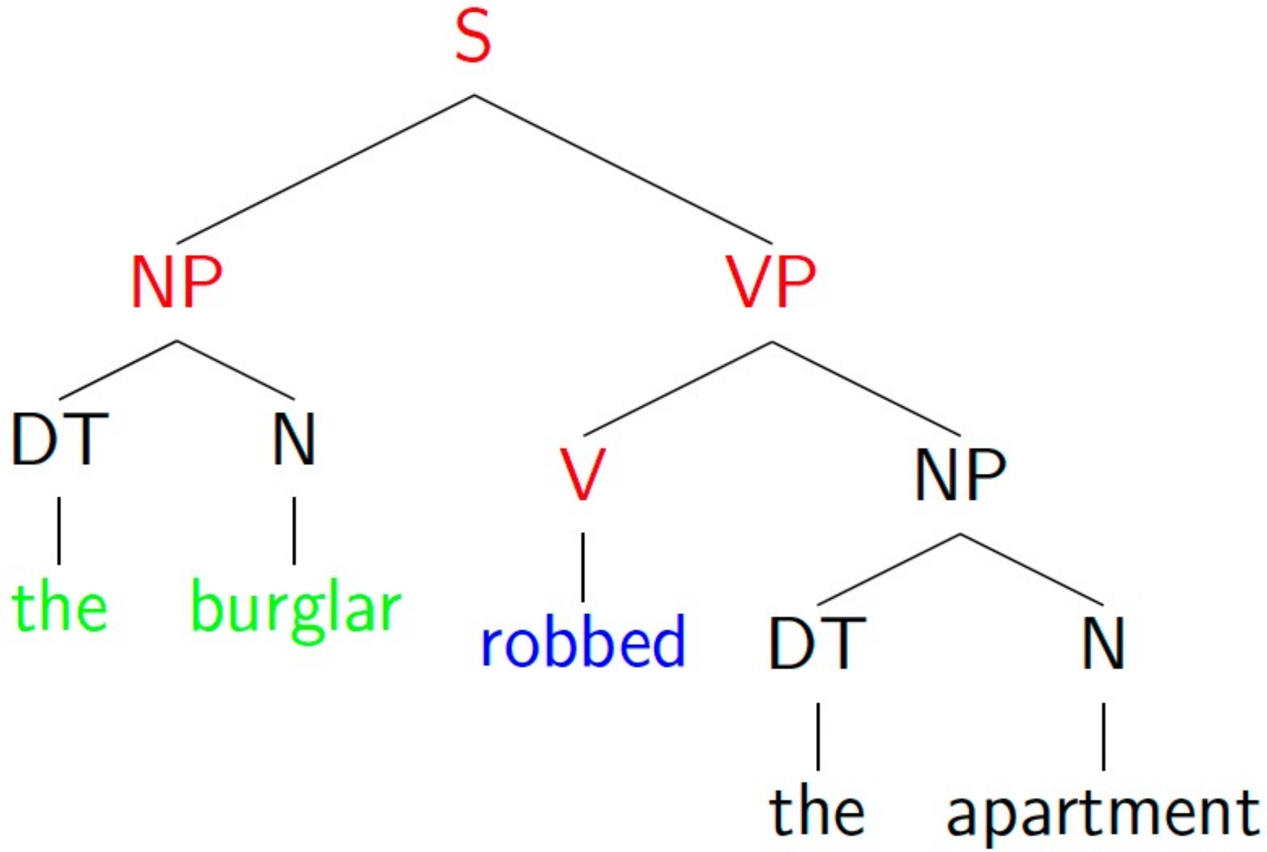
three parties from Brooklyn *arrive*…
a high-class spot such as Mindy's *attracts*…
the Broadway coppers *love*…
they *sit*

Individual words in an NP may not occur before a verb:

*from *arrive*…	*as *attracts*…
*the *is*…	*spot *sat*…

Trees visualize grammatical relationships





‘the burglar’ is the subject of ‘robbed’

Context-free grammars

Context-free grammars (CFG) are also called **Phrase-Structure Grammars.** The idea of basing a grammar on constituent structures is formalized by Chomsky (1956)

A CFG consists of **a set of rules** and **a lexicon** of words and symbols

start symbol

S → NP VP
NP → DT N
VP → V NP
DT → the
V → robbed
N → burglar | apartment

non-terminal symbols

terminal symbols

alternate possible expansions

Context-free grammars

The **formal language** defined by a CFG is the set of strings that are derivable from the start symbol.

L0:
S → NP VP
NP → DT N
VP → V NP
DT → the
V → robbed
N → burglar | apartment

Grammatical:

the burglar robbed the apartment

$[_S\ [_{NP}\ [_{DT}$ the$]\ [_{NP}$ burglar$]]\ [_{VP}\ [_V$ robbed$]\ [_{NP}\ [_{DT}$ the$]\ [_N$ apartment$]]]]$

the apartment robbed the burglar

$[_S\ [_{NP}\ [_{DT}$ the$]\ [_{NP}$ apartment$]]\ [_{VP}\ [_V$ robbed$]\ [_{NP}\ [_{DT}$ the$]\ [_N$ burglar$]]]]$

Ungrammatical:

* *robbed the apartment the burglar*

Formal definition of context-free grammar

N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of the form $A \rightarrow \beta$, where A is a non-terminal, β is a string of symbols from the infinite set of strings $(\Sigma \cup N)^*$
S	a designated **start symbol** and a member of N

Capital letters like A, B, and S	Non-terminals
S	The start symbol
Lower-case Greek letters like α, β, and γ	Strings drawn from $(\Sigma \cup N)^*$
Lower-case Roman letters like u, v, and w	Strings of terminals

Formal language

Hopcroft and Ullman (1979):

if $A \rightarrow \beta$ is a production of R and α and γ are any strings in the set $(\Sigma \cup N)^*$, then we say that $\alpha A \gamma$ **directly derives** $\alpha \beta \gamma$, or $\alpha A \gamma \Rightarrow \alpha \beta \gamma$.

Let $\alpha_1, \alpha_2, \ldots, \alpha_m$ be strings in $(\Sigma \cup N)^*, m \geq 1$, such that

$$\alpha_1 \Rightarrow \alpha_2, \alpha_2 \Rightarrow \alpha_3, \ldots, \alpha_{m-1} \Rightarrow \alpha_m$$

We say that α_1 **derives** α_m, or $\alpha_1 \stackrel{*}{\Rightarrow} \alpha_m$.

$$\mathscr{L}_G = \{w | w \text{ is in } \Sigma * \text{ and } S \stackrel{*}{\Rightarrow} w\}$$

Example

'the dog laughs'

S → NP VP
NP → DT N
DT → the
N → dog
VP → VB
VB → laughs

Derivation

S
NP VP
DT N VP
the N VP
the dog VP
the dog VB
the dog laughs

Rules used

S → NP VP
NP → DT N
DT → the
N → dog
VP → VB
VB → laughs

Problems of CFG

- Cannot express many syntactic phenomena, e.g., subject-verb agreement, *wh*-movement, etc.

- Lack headedness: The head is the word in the phrase that is grammatically the most important. Heads are passed up the parse tree.
  - N is the head of an NP
  - V is the head of a VP
  - P is the head of PP

Nevertheless, CFG is a good starting point.

Recognizer

The problem of recognition: Given a grammar and a string of words, return *True* if the string has a derivation and *False* otherwise.

'the dog laughs'

S → NP VP
NP → DT N
DT → the
N → dog
VP → VB
VB → laughs

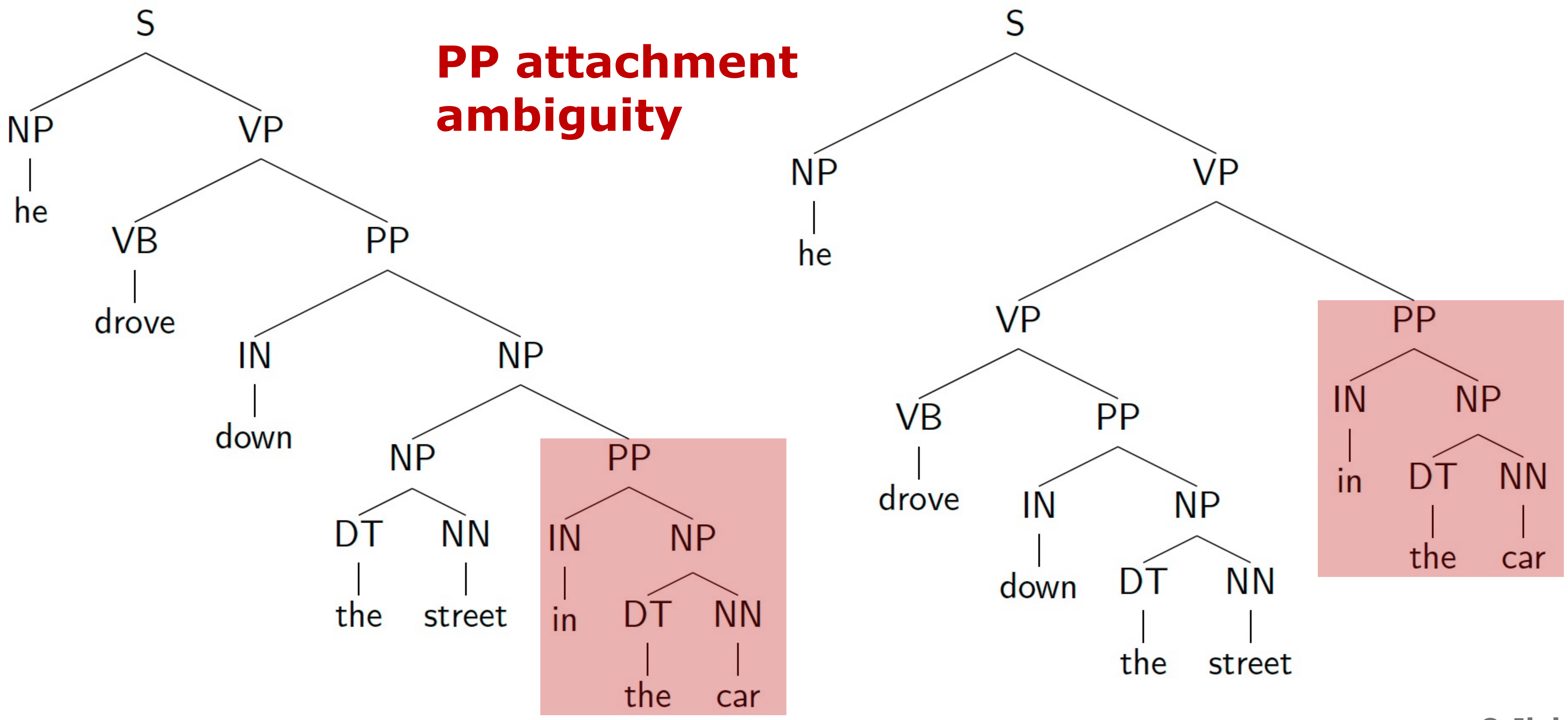
Derivation

S
NP VP
DT N VP
the N VP
the dog VP
the dog VB
the dog laughs

→ ***True***

Ambiguity

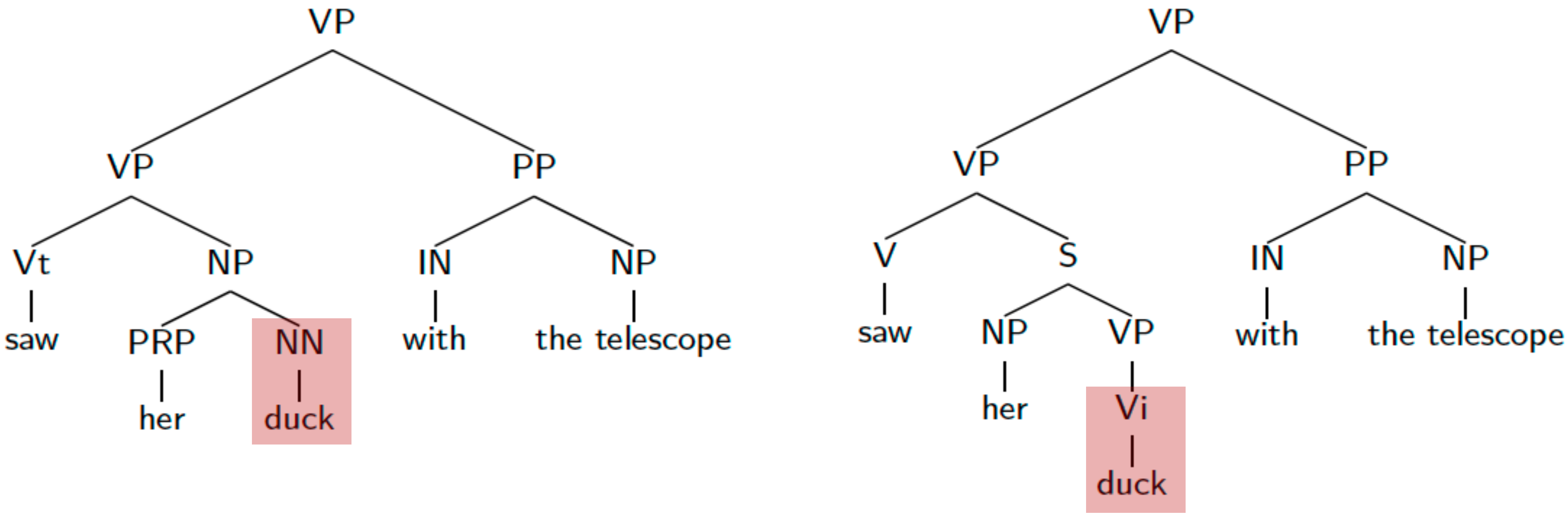
'He drove down the street in the car.'



Source of ambiguity

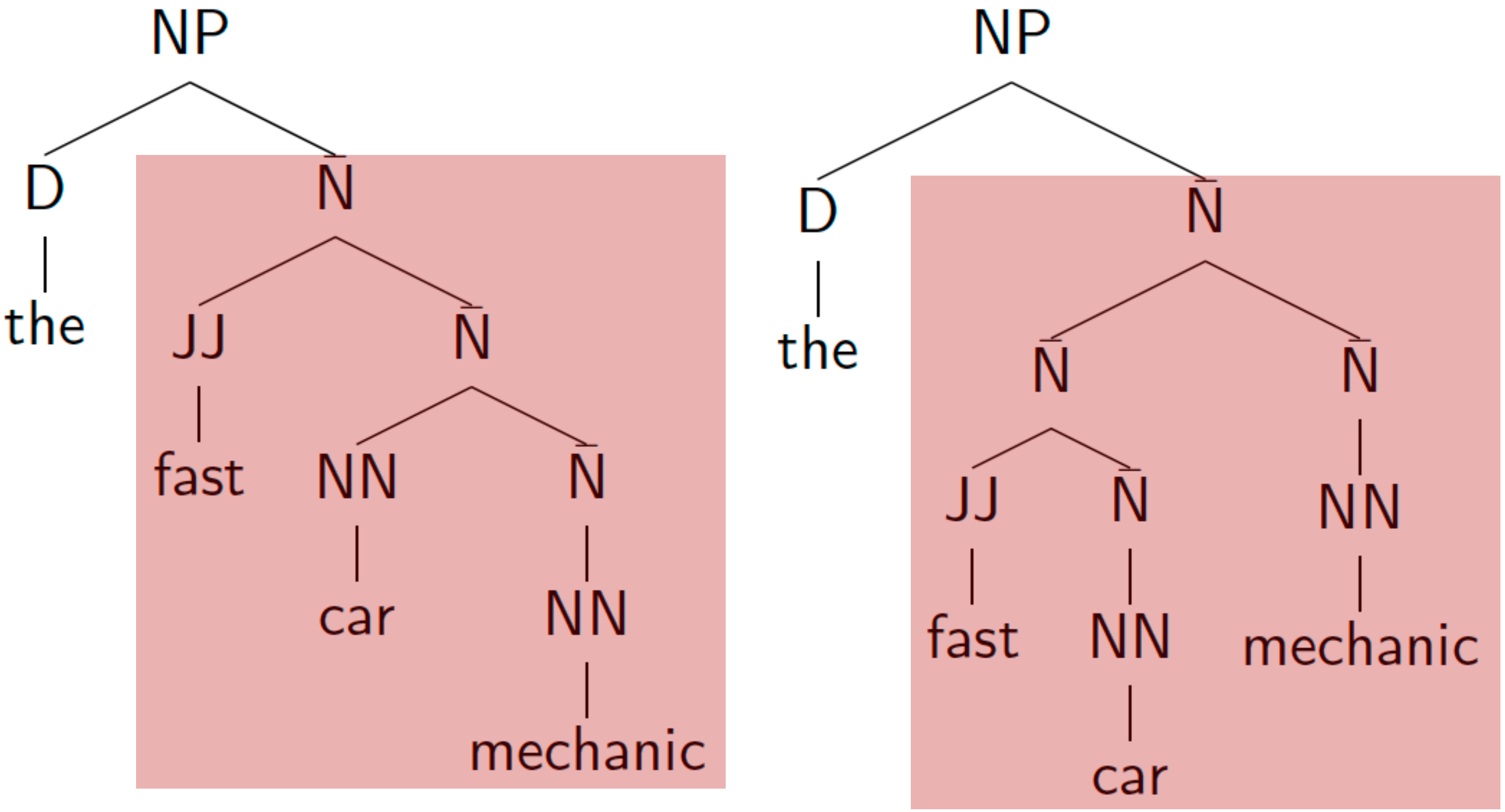
Part-of-speech ambiguity

- NN → duck
- Vi → duck



Source of ambiguity

Noun premodifier ambiguity



Top-down backtrack recognition

Rules:

$$S \rightarrow DP \quad VP$$

$$DP \rightarrow \left\{ \begin{array}{c} (D)\ NP \\ Name \\ Pronoun \end{array} \right\}$$

$$NP \rightarrow N \quad (PP)$$

$$VP \rightarrow V\ (DP)\ \left(\left\{ \begin{array}{c} PP \\ CP \\ VP \end{array} \right\} \right)$$

$$PP \rightarrow P \quad (DP)$$

$$AP \rightarrow A \quad (PP)$$

$$CP \rightarrow C \quad S$$

$$AdvP \rightarrow Adv$$

$$NP \rightarrow AP \quad NP$$

$$NP \rightarrow NP \quad PP$$

$$NP \rightarrow NP \quad CP$$

$$VP \rightarrow AdvP \quad VP$$

$$VP \rightarrow VP \quad PP$$

$$AP \rightarrow AdvP \quad AP$$

$$\alpha \rightarrow \alpha \ \text{Coord} \ \alpha$$

(for α=D,V,N,A,P,C,Adv,VP,NP,DP,AP,PP,AdvP,S,CP)

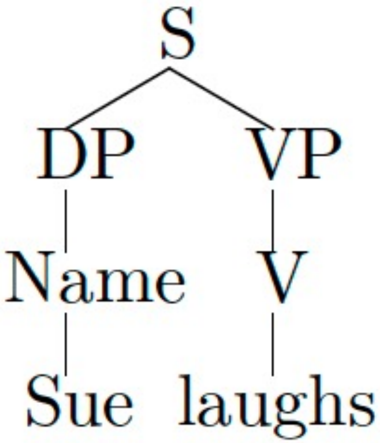
Top-down backtrack recognition

Lexicon:

- D → { the, a, some, every, one, two }
- A → { gentle, clear, honest, compassionate, brave, kind }
- N → { student, teacher, city, university, beer, wine }
- V → { laughs, cries, praises, criticizes, says, knows }
- Adv → { happily, sadly, impartially, generously }
- Name → { Bill, Sue, José, Maria, Presidents Day, Tuesday }
- Pronoun → { he, she, it, her, him }
- P → { in, on, with, by, to, from }
- C → { that, ϵ, whether }
- Coord → { and, or, but }

Top-down backtrack recognition

Derive: ‘*Sue laughs*’



step	predicted	input
0.	S	Sue laughs
1.	DP VP	Sue laughs

step	predicted	input
0.	S	Sue laughs
1.	DP VP	Sue laughs
2a.	D NP VP	Sue laughs
2b.	NP VP	Sue laughs
2c.	Name VP	Sue laughs
2d.	Pronoun VP	Sue laughs
2e.	DP Coord DP VP	Sue laughs

expand: DP

Top-down backtrack recognition

step	predicted	input
0.	S	Sue laughs
1.	DP VP	Sue laughs
2a.	D NP VP	Sue laughs
2b.	NP VP	Sue laughs
2c.	Name VP	Sue laughs
2d.	Pronoun VP	Sue laughs
2e.	DP Coord DP VP	Sue laughs
3aa.	D Coord D NP VP	Sue laughs
3ab.	the NP VP	Sue laughs
3ac.	a NP VP	Sue laughs
3ad.	some NP VP	Sue laughs
3ae.	every NP VP	Sue laughs
3af.	one NP VP	Sue laughs
3ag.	two NP VP	Sue laughs

3aa.	D Coord D NP VP	
4aaa.	D Coord D Coord D NP VP	Sue laughs
4aab.	the Coord D NP VP	Sue laughs
4aab.	a Coord D NP VP	Sue laughs
4aab.	some Coord D NP VP	Sue laughs
4aab.	every Coord D NP VP	Sue laughs
4aab.	one Coord D NP VP	Sue laughs
4aab.	two Coord D NP VP	Sue laughs

The process will never terminate!

Left recursion: category X can contain another category X as its first element

Top-down backtrack recognition

Remove left recursion:

$$S \rightarrow DP \quad VP$$

$$DP \rightarrow \left\{ \begin{matrix} (D)\ NP \\ Name \\ Pronoun \end{matrix} \right\}$$

$$NP \rightarrow N \quad (PP)$$

$$VP \rightarrow V\ (DP)\ \left(\left\{ \begin{matrix} PP \\ CP \\ VP \end{matrix} \right\} \right)$$

$$PP \rightarrow P \quad (DP)$$

$$AP \rightarrow A \quad (PP)$$

$$CP \rightarrow C \quad S$$

$$AdvP \rightarrow Adv$$

$$NP \rightarrow AP \quad NP$$

~~$NP \rightarrow NP \quad PP$~~

~~$NP \rightarrow NP \quad CP$~~

$$VP \rightarrow AdvP \quad VP$$

~~$VP \rightarrow VP \quad PP$~~

$$AP \rightarrow AdvP \quad AP$$

~~$\alpha \rightarrow \alpha \ Coord \ \alpha$~~

(for α=D,V,N,A,P,C,Adv,VP,NP,DP,AP,PP,AdvP,S,CP)

Top-down backtrack recognition

step	predicted	input
0.	S	Sue laughs
1.	DP VP	Sue laughs
2a.	D NP VP	Sue laughs
2b.	NP VP	Sue laughs
2c.	Name VP	Sue laughs
2d.	Pronoun VP	Sue laughs
3aa.	the NP VP	Sue laughs
3ab.	a NP VP	Sue laughs
3ac.	some NP VP	Sue laughs
3ad.	every NP VP	Sue laughs
3ae.	one NP VP	Sue laughs
3af.	two NP VP	Sue laughs

step	predicted	input
0.	S	Sue laughs
1.	DP VP	Sue laughs
2a.	~~D NP VP~~	Sue laughs
2b.	NP VP	Sue laughs
2c.	Name VP	Sue laughs
2d.	Pronoun VP	Sue laughs
3ba.	N VP	Sue laughs
3bb.	N PP VP	Sue laughs

N → { student, teacher, city, university, beer, wine }

step	predicted	input
0.	S	Sue laughs
1.	DP VP	Sue laughs
2a.	~~D NP VP~~	Sue laughs
2b.	~~NP VP~~	Sue laughs
2c.	Name VP	Sue laughs
2d.	Pronoun VP	Sue laughs
3ca.	Bill VP	Sue laughs
3cb.	Sue VP	Sue laughs
3cc.	José VP	Sue laughs
3cd.	Maria VP	Sue laughs
3ce.	Presidents Day VP	Sue laughs
3cf.	Tuesday VP	Sue laughs

scan: Sue

4cb'	VP	~~Sue~~ laughs

Top-down backtrack recognition

4cb’	VP	~~Sue~~ laughs
4cb’a	V	~~Sue~~ laughs
4cb’b	V DP	~~Sue~~ laughs
4cb’c	V PP	~~Sue~~ laughs
4cb’d	V CP	~~Sue~~ laughs
4cb’e	V VP	~~Sue~~ laughs
4cb’f	V DP PP	~~Sue~~ laughs
4cb’g	V DP CP	~~Sue~~ laughs
4cb’h	V DP VP	~~Sue~~ laughs
4cb’aa	laughs	~~Sue~~ laughs

scan: laughs

4cb’	VP	~~Sue~~ laughs

Problems of top-down parsing

- Slow: **expand** requires looping through the whole list of rules
- Non-terminating with left-recursion

To do

- Do HW3
- Optional reading: **NLTK** Ch8:1-3; **SLP** Ch12.1-4; **Stabler** Ch1