Bottom-up Chart Parsing: the CKY algorithm

Data Structures and Algorithms for Computational Linguistics III (ISCL-BA-07)

Çağrı Çöltekin
ccoltekin@sfs.uni-tuebingen.de

University of Tübingen
Seminar für Sprachwissenschaft

Winter Semester 2022/23

version: 95b35bb @2023-01-25

Parsing so far

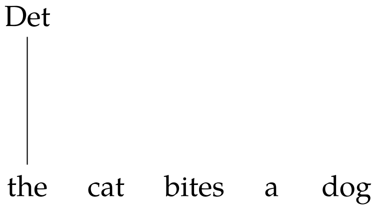
- Parsing is the task of automatic syntactic analysis
- For most practical purposes, context-free grammars are the most useful formalism for parsing
- We can formulate parsing as
  - Top-down: begin with the start symbol, try to *produce* the input string to be parsed
  - Bottom up: begin with the input, and try to *reduce* it to the start symbol
- Both strategies can be cast as search with backtracking
- Backtracking parsers are inefficient: they recompute sub-trees multiple times

Bottom-up parsing as search

the cat bites a dog

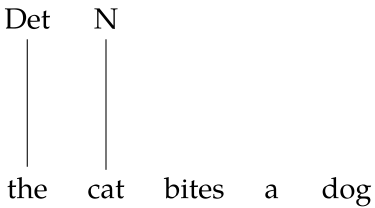
S	$\rightarrow$	NP VP
NP	$\rightarrow$	Det N
VP	$\rightarrow$	V NP
VP	$\rightarrow$	V
Det	$\rightarrow$	a
Det	$\rightarrow$	the
N	$\rightarrow$	cat
N	$\rightarrow$	dog
V	$\rightarrow$	bites
N	$\rightarrow$	bites

Bottom-up parsing as search



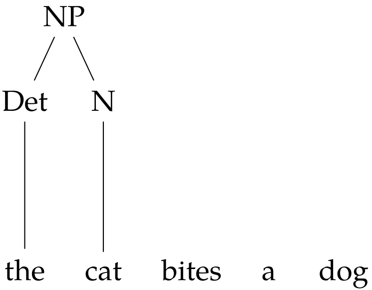
S	→	NP VP
NP	→	Det N
VP	→	V NP
VP	→	V
Det	→	a
Det	→	the
N	→	cat
N	→	dog
V	→	bites
N	→	bites

Bottom-up parsing as search



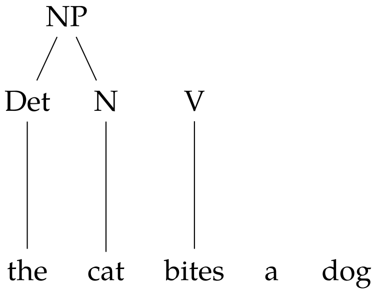
S	→	NP VP
NP	→	Det N
VP	→	V NP
VP	→	V
Det	→	a
Det	→	the
N	→	cat
N	→	dog
V	→	bites
N	→	bites

Bottom-up parsing as search



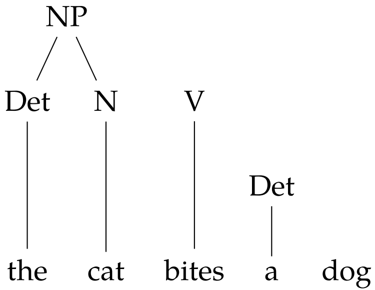
S	$\rightarrow$	NP VP
NP	$\rightarrow$	Det N
VP	$\rightarrow$	V NP
VP	$\rightarrow$	V
Det	$\rightarrow$	a
Det	$\rightarrow$	the
N	$\rightarrow$	cat
N	$\rightarrow$	dog
V	$\rightarrow$	bites
N	$\rightarrow$	bites

Bottom-up parsing as search



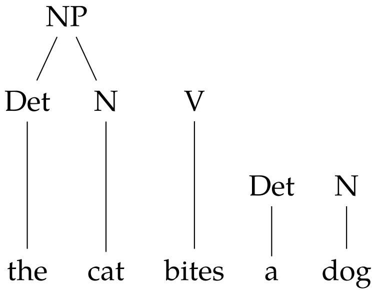
S	→	NP VP
NP	→	Det N
VP	→	V NP
VP	→	V
Det	→	a
Det	→	the
N	→	cat
N	→	dog
V	→	bites
N	→	bites

Bottom-up parsing as search



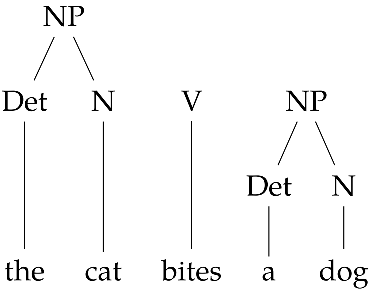
S	→	NP VP
NP	→	Det N
VP	→	V NP
VP	→	V
Det	→	a
Det	→	the
N	→	cat
N	→	dog
V	→	bites
N	→	bites

Bottom-up parsing as search



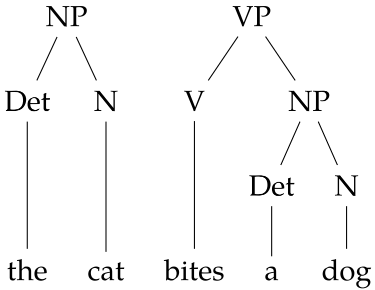
S → NP VP
NP → Det N
VP → V NP
VP → V
Det → a
Det → the
N → cat
N → dog
V → bites
N → bites

Bottom-up parsing as search



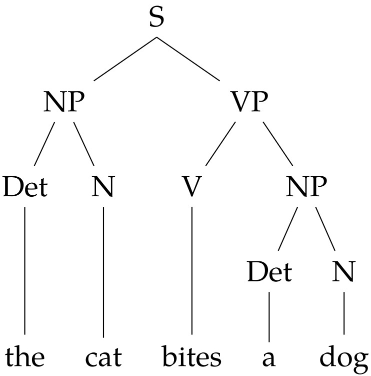
S	$\rightarrow$	NP VP
NP	$\rightarrow$	Det N
VP	$\rightarrow$	V NP
VP	$\rightarrow$	V
Det	$\rightarrow$	a
Det	$\rightarrow$	the
N	$\rightarrow$	cat
N	$\rightarrow$	dog
V	$\rightarrow$	bites
N	$\rightarrow$	bites

Bottom-up parsing as search



S	$\rightarrow$	NP VP
NP	$\rightarrow$	Det N
VP	$\rightarrow$	V NP
VP	$\rightarrow$	V
Det	$\rightarrow$	a
Det	$\rightarrow$	the
N	$\rightarrow$	cat
N	$\rightarrow$	dog
V	$\rightarrow$	bites
N	$\rightarrow$	bites

Bottom-up parsing as search



S	→	NP VP
NP	→	Det N
VP	→	V NP
VP	→	V
Det	→	a
Det	→	the
N	→	cat
N	→	dog
V	→	bites
N	→	bites

Dealing with ambiguity

S	$\rightarrow$	NP VP
NP	$\rightarrow$	Prn N
NP	$\rightarrow$	Prn
VP	$\rightarrow$	V NP
VP	$\rightarrow$	V
VP	$\rightarrow$	V S
N	$\rightarrow$	duck
V	$\rightarrow$	duck
V	$\rightarrow$	saw
Prn	$\rightarrow$	I $\leftarrow$
Prn	$\rightarrow$	she
Prn	$\rightarrow$	her

I saw her duck

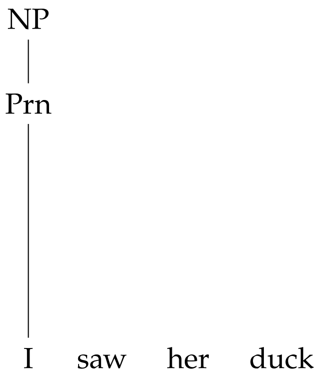
Dealing with ambiguity

Prn

I saw her duck

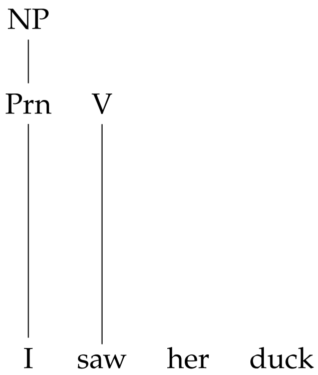
S	→ NP VP	
NP	→ Prn N	
NP	→ Prn	←
VP	→ V NP	
VP	→ V	
VP	→ V S	
N	→ duck	
V	→ duck	
V	→ saw	
Prn	→ I	
Prn	→ she	
Prn	→ her	

Dealing with ambiguity



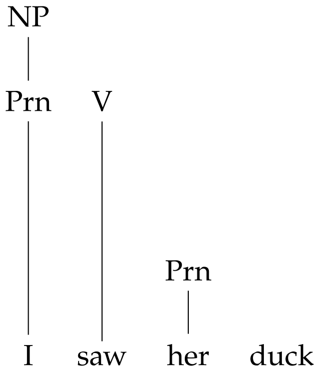
S	→	NP VP
NP	→	Prn N
NP	→	Prn
VP	→	V NP
VP	→	V
VP	→	V S
N	→	duck
V	→	duck
V	→	saw ←
Prn	→	I
Prn	→	she
Prn	→	her

Dealing with ambiguity



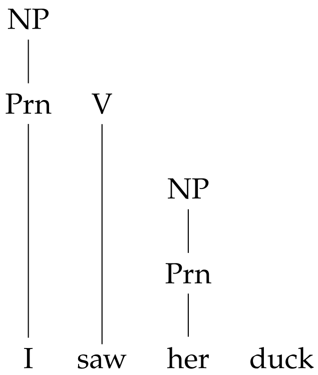
S	→	NP VP
NP	→	Prn N
NP	→	Prn
VP	→	V NP
VP	→	V
VP	→	V S
N	→	duck
V	→	duck
V	→	saw
Prn	→	I
Prn	→	she
Prn	→	her ←

Dealing with ambiguity



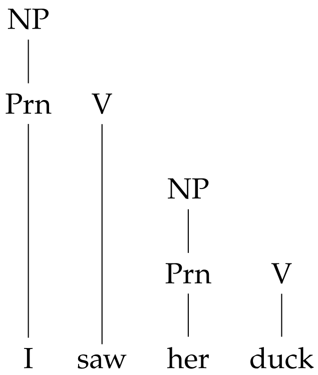
S → NP VP
NP → Prn N
NP → Prn ←
VP → V NP
VP → V
VP → V S
N → duck
V → duck
V → saw
Prn → I
Prn → she
Prn → her

Dealing with ambiguity



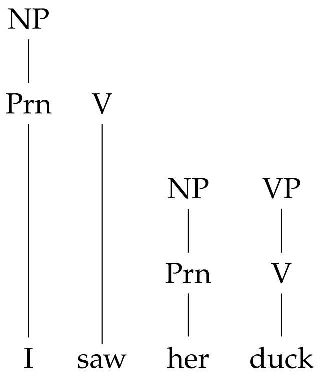
$S \rightarrow$ NP VP
$NP \rightarrow$ Prn N
$NP \rightarrow$ Prn
$VP \rightarrow$ V NP
$VP \rightarrow$ V
$VP \rightarrow$ V S
$N \rightarrow$ duck
$V \rightarrow$ duck $\leftarrow$
$V \rightarrow$ saw
$Prn \rightarrow$ I
$Prn \rightarrow$ she
$Prn \rightarrow$ her

Dealing with ambiguity



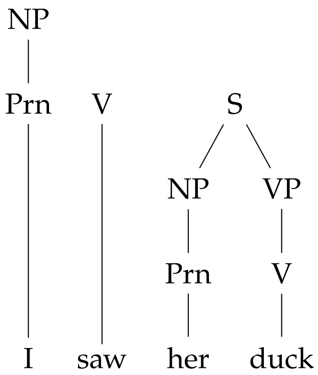
$$
\begin{aligned}
S &\rightarrow NP\ VP \\
NP &\rightarrow Prn\ N \\
NP &\rightarrow Prn \\
VP &\rightarrow V\ NP \\
VP &\rightarrow V \quad \leftarrow \\
VP &\rightarrow V\ S \\
N &\rightarrow duck \\
V &\rightarrow duck \\
V &\rightarrow saw \\
Prn &\rightarrow I \\
Prn &\rightarrow she \\
Prn &\rightarrow her
\end{aligned}
$$

Dealing with ambiguity



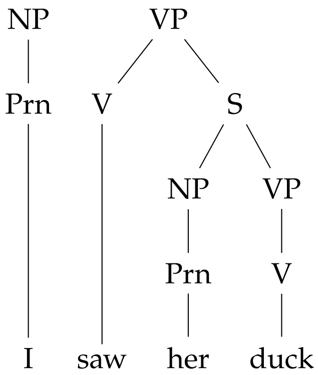
S → NP VP ←
NP → Prn N
NP → Prn
VP → V NP
VP → V
VP → V S
N → duck
V → duck
V → saw
Prn → I
Prn → she
Prn → her

Dealing with ambiguity



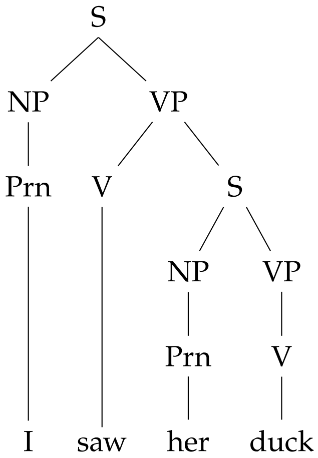
S	→	NP VP	
NP	→	Prn N	
NP	→	Prn	
VP	→	V NP	
VP	→	V	
VP	→	V S	←
N	→	duck	
V	→	duck	
V	→	saw	
Prn	→	I	
Prn	→	she	
Prn	→	her	

Dealing with ambiguity



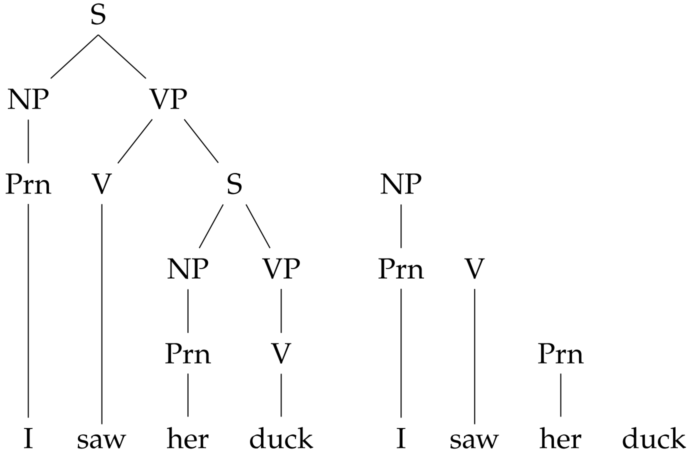
S → NP VP ←
NP → Prn N
NP → Prn
VP → V NP
VP → V
VP → V S
N → duck
V → duck
V → saw
Prn → I
Prn → she
Prn → her

Dealing with ambiguity



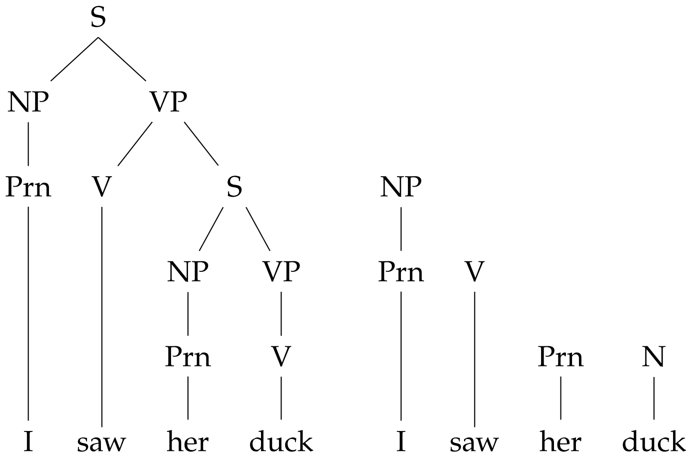
S → NP VP
NP → Prn N
NP → Prn
VP → V NP
VP → V
VP → V S
N → duck
V → duck
V → saw
Prn → I
Prn → she
Prn → her

Dealing with ambiguity



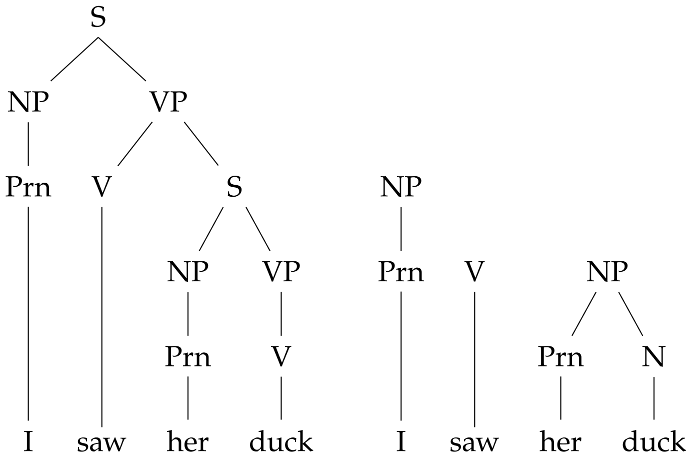
S → NP VP
NP → Prn N
NP → Prn
VP → V NP
VP → V
VP → V S
N → duck ←
V → duck
V → saw
Prn → I
Prn → she
Prn → her

Dealing with ambiguity



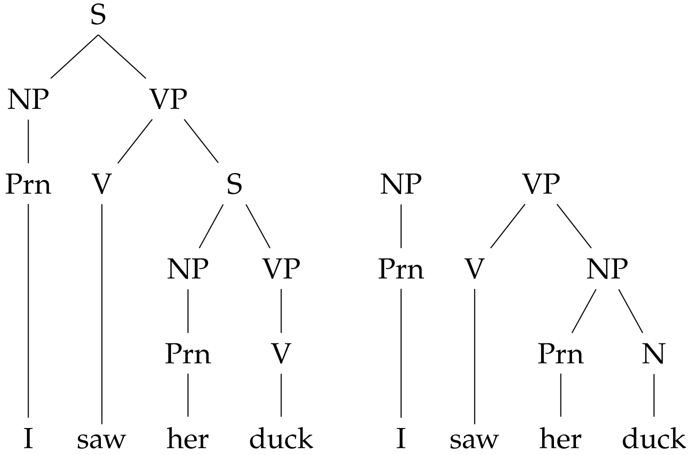
- S → NP VP
- NP → Prn N ←
- NP → Prn
- VP → V NP
- VP → V
- VP → V S
- N → duck
- V → duck
- V → saw
- Prn → I
- Prn → she
- Prn → her

Dealing with ambiguity



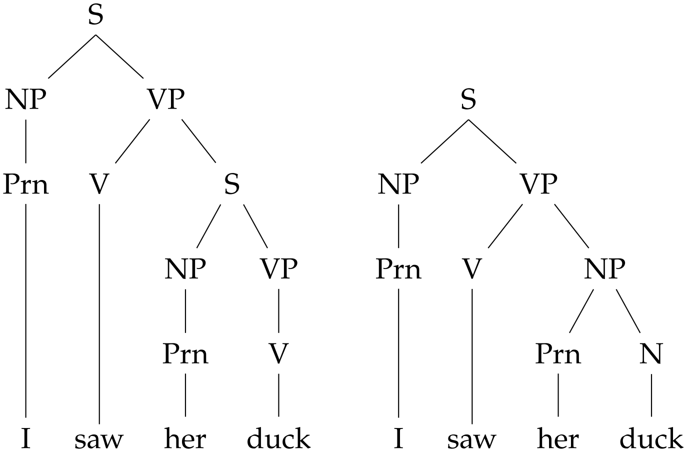
S → NP VP
NP → Prn N
NP → Prn
VP → V NP ←
VP → V
VP → V S
N → duck
V → duck
V → saw
Prn → I
Prn → she
Prn → her

Dealing with ambiguity



S → NP VP ←
NP → Prn N
NP → Prn
VP → V NP
VP → V
VP → V S
N → duck
V → duck
V → saw
Prn → I
Prn → she
Prn → her

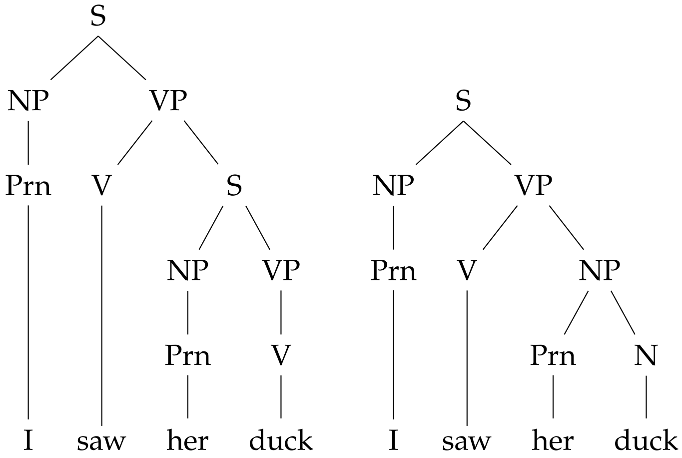
Dealing with ambiguity



S $\rightarrow$ NP VP
NP $\rightarrow$ Prn N
NP $\rightarrow$ Prn
VP $\rightarrow$ V NP
VP $\rightarrow$ V
VP $\rightarrow$ V S
N $\rightarrow$ duck
V $\rightarrow$ duck
V $\rightarrow$ saw
Prn $\rightarrow$ I
Prn $\rightarrow$ she
Prn $\rightarrow$ her

How to represent multiple parses

parse forest grammar



$$
\begin{aligned}
S_{0:4} &\rightarrow NP_{0:1}\ VP_{1:4} \\
NP_{0:1} &\rightarrow Prn_{0:1} \\
Prn_{0:1} &\rightarrow I_{0:1} \\
VP_{1:4} &\rightarrow V_{1:2}\ S_{2:4} \\
V_{1:2} &\rightarrow saw_{1:2} \\
S_{2:4} &\rightarrow Prn_{2:3}\ V_{3:4} \\
V_{3:4} &\rightarrow duck_{3:4} \\
\hline
VP_{1:4} &\rightarrow V_{1:2}\ NP_{2:4} \\
NP_{2:4} &\rightarrow Prn_{2:3}\ N_{3:4}
\end{aligned}
$$

CKY algorithm

- The CKY (Cocke–Kasami–Younger) parsing algorithm is a dynamic programming algorithm
- It processes the input *bottom up*, and saves the intermediate results on a *chart*
- Time complexity for *recognition* is $O(n^3)$
- Space complexity is $O(n^2)$
- It requires the CFG to be in *Chomsky normal form* (CNF) (can somewhat be relaxed, but not common)

Chomsky normal form (CNF)

- A CFG is in CNF, if the rewrite rules are in one of the following forms
  - A $\rightarrow$ B C
  - A $\rightarrow$ a

  where A, B, C are non-terminals and a is a terminal
- Any CFG can be converted to CNF
- Resulting grammar is *weakly equivalent* to the original grammar:
  - it generates/accepts the same language
  - but the derivations are different

Converting to CNF: example

$$
\begin{aligned}
\text{S} &\rightarrow \text{NP VP}\\
\text{S} &\rightarrow \text{Aux NP VP}\\
\text{NP} &\rightarrow \text{the N}\\
\text{VP} &\rightarrow \text{V NP}\\
\text{VP} &\rightarrow \text{V}\\
\text{N} &\rightarrow \text{cat}\\
\text{N} &\rightarrow \text{dog}\\
\text{V} &\rightarrow \text{bites}\\
\text{N} &\rightarrow \text{bites}
\end{aligned}
$$

Converting to CNF: example

S	→	NP VP
S	→	Aux NP VP
NP	→	the N
VP	→	V NP
VP	→	V
N	→	cat
N	→	dog
V	→	bites
N	→	bites

- S → Aux NP VP

$$\text{S} \rightarrow \text{Aux NP VP} \quad \Rightarrow \quad \begin{array}{l} \text{S} \rightarrow \text{Aux X} \\ \text{X} \rightarrow \text{NP VP} \end{array}$$

Converting to CNF: example

S	$\rightarrow$	NP VP
S	$\rightarrow$	Aux NP VP
NP	$\rightarrow$	the N
VP	$\rightarrow$	V NP
VP	$\rightarrow$	V
N	$\rightarrow$	cat
N	$\rightarrow$	dog
V	$\rightarrow$	bites
N	$\rightarrow$	bites

- S $\rightarrow$ Aux NP VP

  S $\rightarrow$ Aux NP VP $\Rightarrow$ S $\rightarrow$ Aux X

  X $\rightarrow$ NP VP

- NP $\rightarrow$ the N

  NP $\rightarrow$ the N $\Rightarrow$ NP $\rightarrow$ X N

  X $\rightarrow$ the

Converting to CNF: example

$$
\begin{array}{lcl}
S & \rightarrow & NP\ VP \\
S & \rightarrow & Aux\ NP\ VP \\
NP & \rightarrow & the\ N \\
VP & \rightarrow & V\ NP \\
VP & \rightarrow & V \\
N & \rightarrow & cat \\
N & \rightarrow & dog \\
V & \rightarrow & bites \\
N & \rightarrow & bites
\end{array}
$$

- S → Aux NP VP

  $$S \rightarrow Aux\ NP\ VP \quad \Rightarrow \quad \begin{array}{l} S \rightarrow Aux\ X \\ X \rightarrow NP\ VP \end{array}$$

- NP → the N

  $$NP \rightarrow the\ N \quad \Rightarrow \quad \begin{array}{l} NP \rightarrow X\ N \\ X \rightarrow the \end{array}$$

- VP → V

  $$VP \rightarrow V \quad \Rightarrow \quad VP \rightarrow bites$$

Converting to CNF

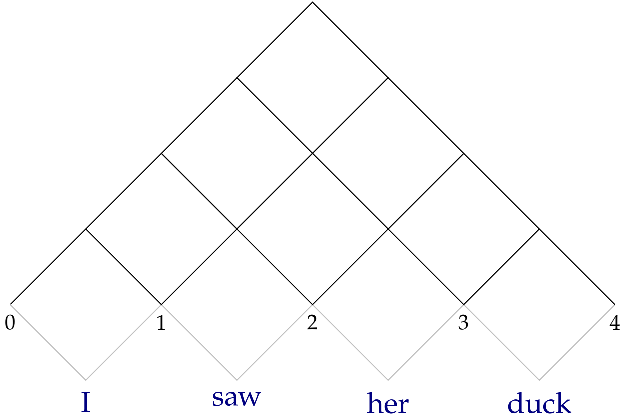
1. Eliminate the ϵ rules: if $A \rightarrow \epsilon$ is in the grammar
   - replace any rule $B \rightarrow \alpha\ A\ \beta$ with two rules

     $B \rightarrow \alpha\ \beta$

     $B \rightarrow \alpha\ A'\ \beta$
   - add $A' \rightarrow \alpha$ for all α (except ϵ) whose LHS is A
   - repeat the process for newly created ϵ rules
   - remove the rules with ϵ on the RHS (except $S \rightarrow \epsilon$)
2. Eliminate unit rules: for a rule $A \rightarrow B$
   - Replace the rule with $A \rightarrow \alpha_1 \mid \ldots \mid \alpha_n$, where $\alpha_1, \ldots, \alpha_n$ are all RHS or rule B
   - Remove the rule $A \rightarrow B$
   - Repeat the process until no unit rules remain
3. Binarize all the non-binary rules with non-terminal on the RHS: for a rule $A \rightarrow X_1\ X_2 \ldots X_n$:
   - Replace the rule with $A \rightarrow A_1\ X_3 \ldots X_n$, and add $A_1 \rightarrow X_1\ X_2$
   - Repeat the process until all new rules are binary

CKY demonstration

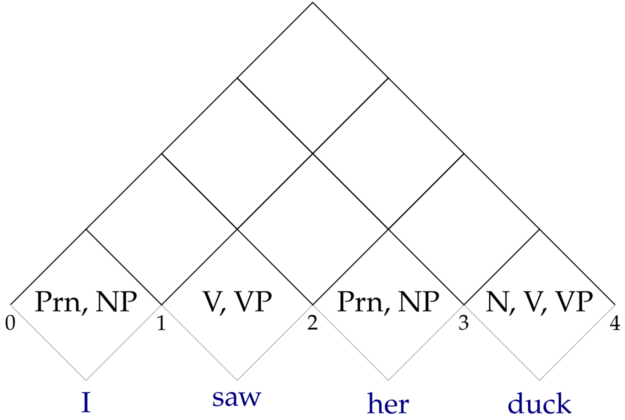
an ambiguous example



$$
\begin{aligned}
S &\rightarrow NP\ VP \\
NP &\rightarrow Prn\ N \\
VP &\rightarrow V\ NP \\
VP &\rightarrow V\ S \\
N &\rightarrow duck \\
VP &\rightarrow duck \mid saw \\
V &\rightarrow duck \mid saw \\
Prn &\rightarrow I \mid she \mid her \\
NP &\rightarrow I \mid she \mid her
\end{aligned}
$$

CKY demonstration

an ambiguous example

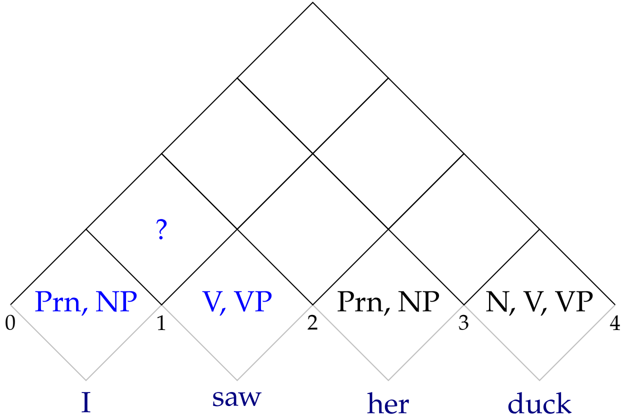


S → NP VP
NP → Prn N
VP → V NP
VP → V S
N → duck
VP → duck | saw
V → duck | saw
Prn → I | she | her
NP → I | she | her

CKY demonstration

an ambiguous example

$$S \rightarrow NP\ VP$$

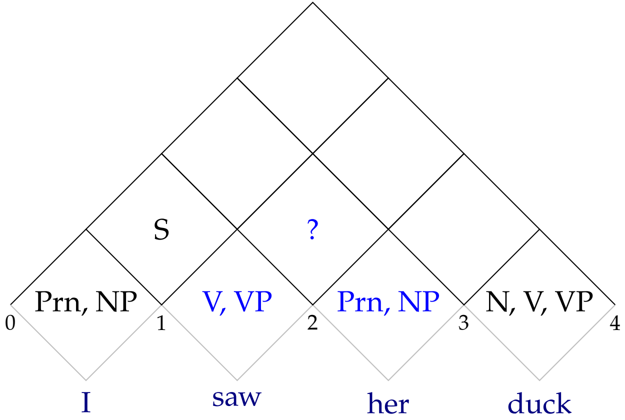


S	→	NP VP
NP	→	Prn N
VP	→	V NP
VP	→	V S
N	→	duck
VP	→	duck | saw
V	→	duck | saw
Prn	→	I | she | her
NP	→	I | she | her

CKY demonstration

an ambiguous example

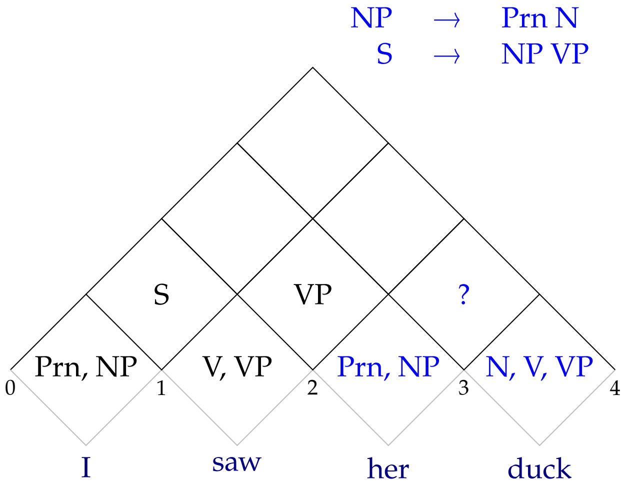
VP → V NP



S → NP VP
NP → Prn N
VP → V NP
VP → V S
N → duck
VP → duck | saw
V → duck | saw
Prn → I | she | her
NP → I | she | her

CKY demonstration

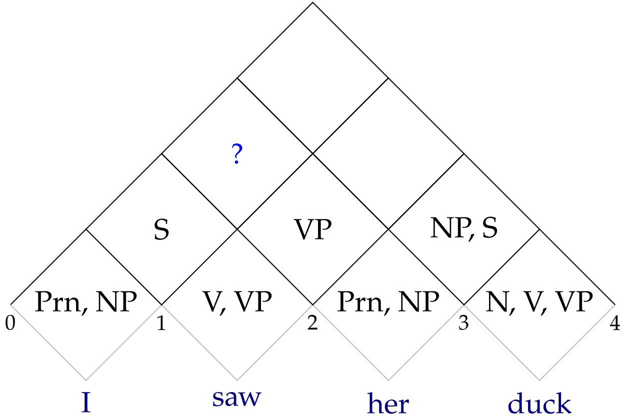
an ambiguous example



S	→	NP VP
NP	→	Prn N
VP	→	V NP
VP	→	V S
N	→	duck
VP	→	duck | saw
V	→	duck | saw
Prn	→	I | she | her
NP	→	I | she | her

CKY demonstration

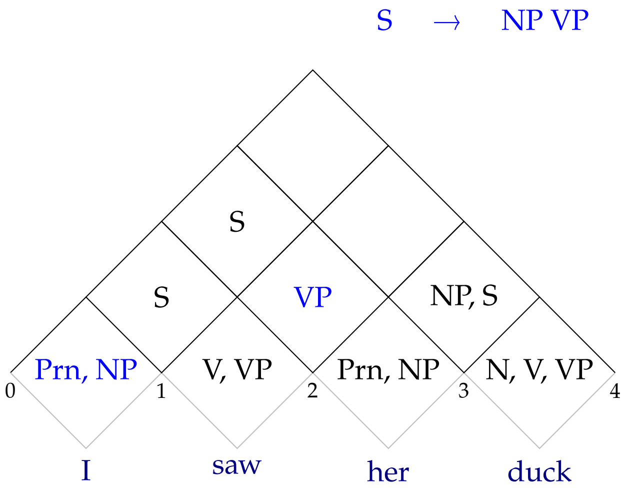
an ambiguous example



$$
\begin{aligned}
S &\rightarrow NP\ VP \\
NP &\rightarrow Prn\ N \\
VP &\rightarrow V\ NP \\
VP &\rightarrow V\ S \\
N &\rightarrow duck \\
VP &\rightarrow duck \mid saw \\
V &\rightarrow duck \mid saw \\
Prn &\rightarrow I \mid she \mid her \\
NP &\rightarrow I \mid she \mid her
\end{aligned}
$$

CKY demonstration

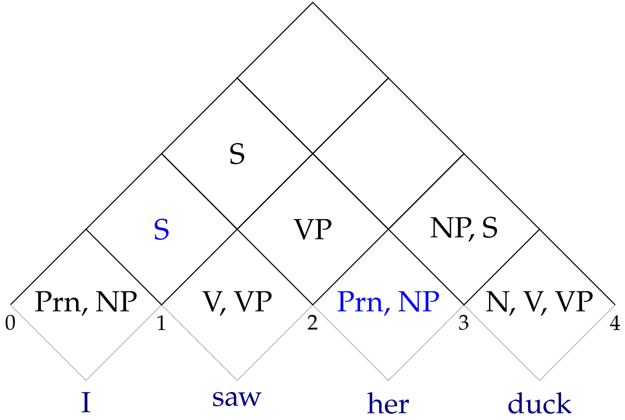
an ambiguous example



S	→	NP VP
NP	→	Prn N
VP	→	V NP
VP	→	V S
N	→	duck
VP	→	duck | saw
V	→	duck | saw
Prn	→	I | she | her
NP	→	I | she | her

CKY demonstration

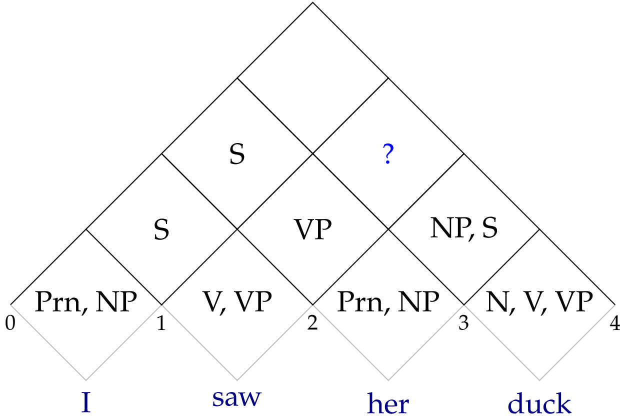
an ambiguous example



S	→	NP VP
NP	→	Prn N
VP	→	V NP
VP	→	V S
N	→	duck
VP	→	duck | saw
V	→	duck | saw
Prn	→	I | she | her
NP	→	I | she | her

CKY demonstration

an ambiguous example

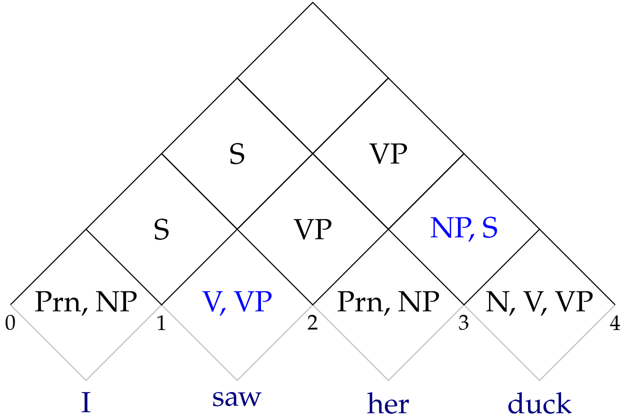


S $\rightarrow$ NP VP
NP $\rightarrow$ Prn N
VP $\rightarrow$ V NP
VP $\rightarrow$ V S
N $\rightarrow$ duck
VP $\rightarrow$ duck | saw
V $\rightarrow$ duck | saw
Prn $\rightarrow$ I | she | her
NP $\rightarrow$ I | she | her

CKY demonstration

an ambiguous example

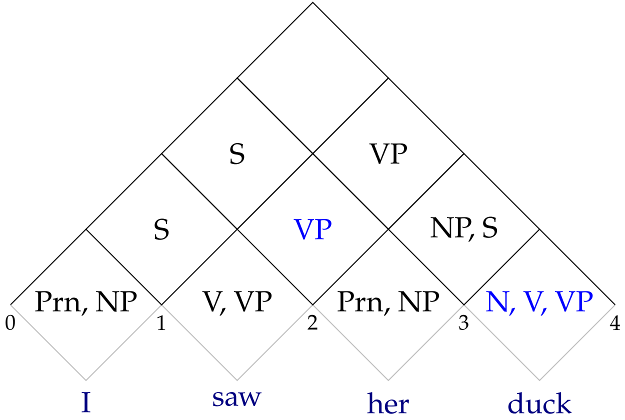
VP → V NP
VP → V S



S → NP VP
NP → Prn N
VP → V NP
VP → V S
N → duck
VP → duck | saw
V → duck | saw
Prn → I | she | her
NP → I | she | her

CKY demonstration

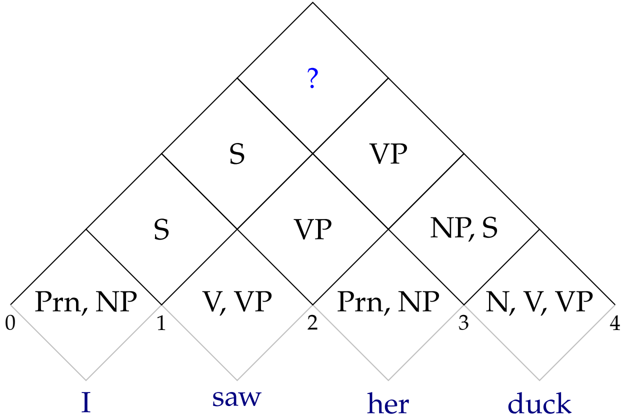
an ambiguous example



S	→	NP VP
NP	→	Prn N
VP	→	V NP
VP	→	V S
N	→	duck
VP	→	duck | saw
V	→	duck | saw
Prn	→	I | she | her
NP	→	I | she | her

CKY demonstration

an ambiguous example

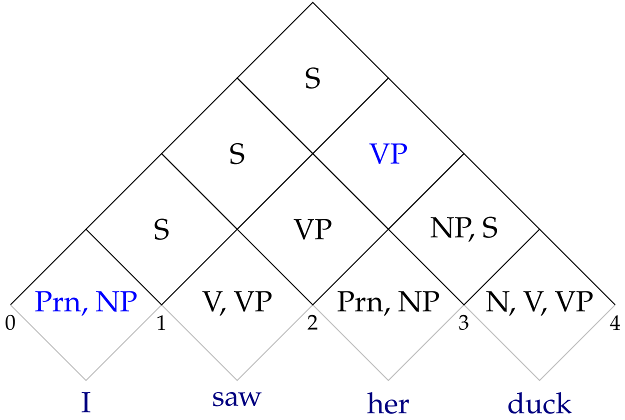


S	→	NP VP
NP	→	Prn N
VP	→	V NP
VP	→	V S
N	→	duck
VP	→	duck | saw
V	→	duck | saw
Prn	→	I | she | her
NP	→	I | she | her

CKY demonstration

an ambiguous example

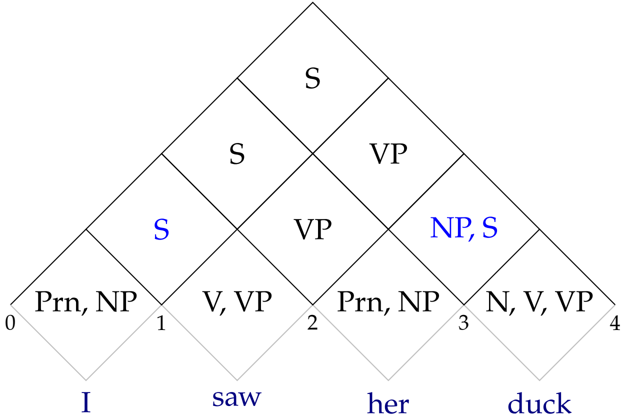
$S \rightarrow NP\ VP$



S	→	NP VP
NP	→	Prn N
VP	→	V NP
VP	→	V S
N	→	duck
VP	→	duck | saw
V	→	duck | saw
Prn	→	I | she | her
NP	→	I | she | her

CKY demonstration

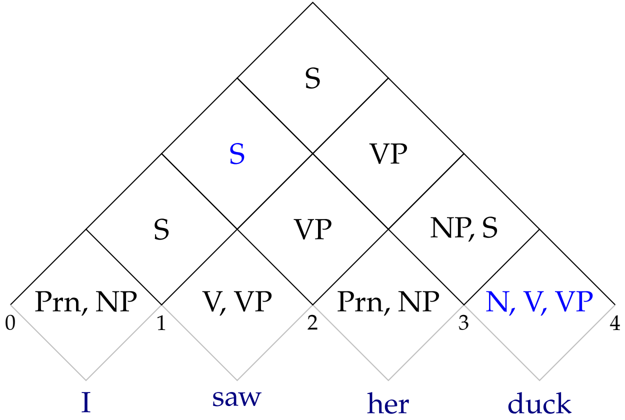
an ambiguous example



$S \rightarrow NP\ VP$
$NP \rightarrow Prn\ N$
$VP \rightarrow V\ NP$
$VP \rightarrow V\ S$
$N \rightarrow duck$
$VP \rightarrow duck \mid saw$
$V \rightarrow duck \mid saw$
$Prn \rightarrow I \mid she \mid her$
$NP \rightarrow I \mid she \mid her$

CKY demonstration

an ambiguous example



S → NP VP
NP → Prn N
VP → V NP
VP → V S
N → duck
VP → duck | saw
V → duck | saw
Prn → I | she | her
NP → I | she | her

CKY demonstration: the chart

our chart is a 2D array

NP, Prn	S	S	S
	V, VP	VP	VP
		Prn	NP, S
			V, N, NP

0 she 1 saw 2 her 3 duck 4

Space complexity is $O(n^2)$.

CKY demonstration: the chart

our chart is a 2D array – this is more convenient for programming

S			
S	VP		
S	VP	NP, S	
NP, Prn	V, VP	Prn, NP	V, N, NP
she	saw	her	duck

0 she 1 saw 2 her 3 duck 4

Space complexity is $O(n^2)$.

Parsing vs. recognition

- We went through a recognition example
- Note that the algorithm is not directional: it takes the complete input
- Recognition accepts or rejects a sentence based on a grammar
- For parsing, we want to know the derivations that yielded a correct parse
- To recover parse trees, we
  - follow the same procedure as recognition
  - add back links to keep track of the derivations

Chart parsing example (CKY parsing)

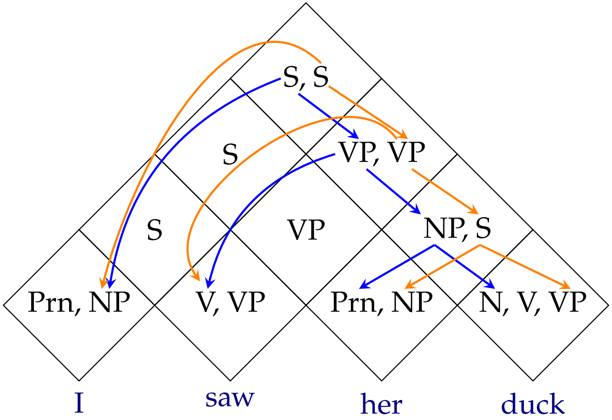
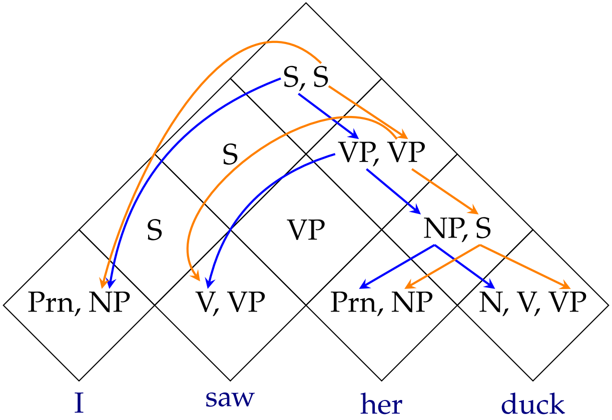


Chart parsing example (CKY parsing)



The chart stores a *parse forest* efficiently.

Summary

+ CKY avoids re-computing the analyses by storing the earlier analyses (of sub-spans) in a table
– It still computes lower level constituents that are not allowed by the grammar
– CKY requires the grammar to be in CNF

- CKY has $O(n^3)$ recognition complexity
- For parsing we need to keep track of backlinks
- CKY can efficiently store all possible parses in a chart
- Enumerating all possible parses have exponential complexity (worst case)
- Suggested reading: Jurafsky and Martin (2009, draft 3rd ed, section 13.2)

Summary

+ CKY avoids re-computing the analyses by storing the earlier analyses (of sub-spans) in a table
– It still computes lower level constituents that are not allowed by the grammar
– CKY requires the grammar to be in CNF

- CKY has $O(n^3)$ recognition complexity
- For parsing we need to keep track of backlinks
- CKY can efficiently store all possible parses in a chart
- Enumerating all possible parses have exponential complexity (worst case)
- Suggested reading: Jurafsky and Martin (2009, draft 3rd ed, section 13.2)

Next:

- Top-down chart parsing: Earley algorithm
- Suggested reading:
  - Jurafsky and Martin (2009, section 13.2.4)
  - Grune and Jacobs (2007, section 7.2)

Acknowledgments, references, additional reading material

- Grune, Dick and Ceriel J.H. Jacobs (2007). *Parsing Techniques: A Practical Guide*. second. Monographs in Computer Science. The first edition is available at `http://dickgrune.com/Books/PTAPG_1st_Edition/BookBody.pdf`. Springer New York. ISBN: 9780387689548.
- Jurafsky, Daniel and James H. Martin (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. second edition. Pearson Prentice Hall. ISBN: 978-0-13-504196-3.