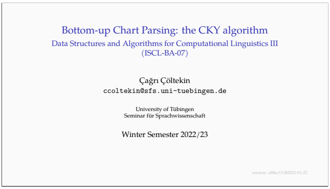


Bottom-up Chart Parsing: the CKY algorithm

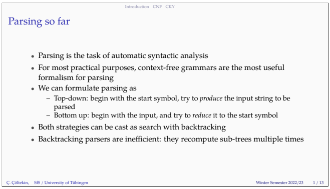
Data Structures and Algorithms for Computational Linguistics III
(ISCL-BA-07)

Çağrı Çöltekin
ccoltekin@sfs.uni-tuebingen.de

University of Tübingen
Seminar für Sprachwissenschaft

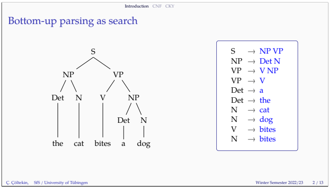
Winter Semester 2022/23

version: a88ac13 @2023-01-22



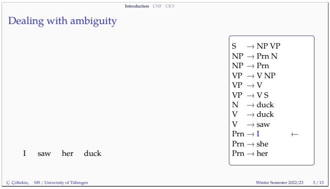
Parsing so far

- 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



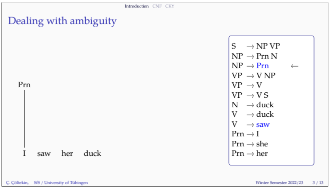
Bottom-up parsing as search

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



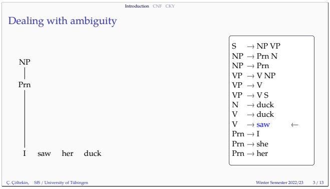
Dealing with ambiguity

S → 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



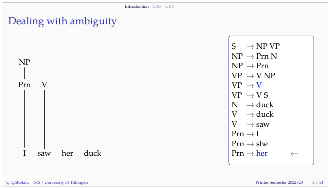
Dealing with ambiguity

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



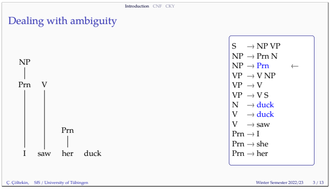
Dealing with ambiguity

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



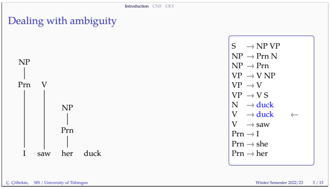
Dealing with ambiguity

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 ←



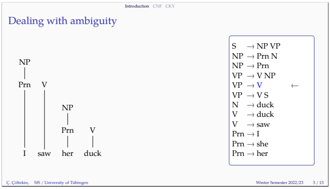
Dealing with ambiguity

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



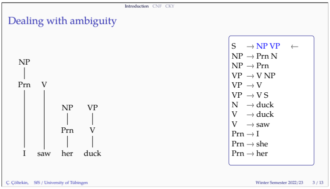
Dealing with ambiguity

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



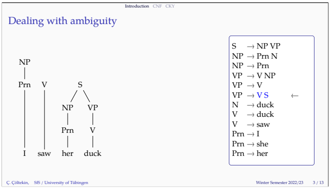
Dealing with ambiguity

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

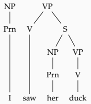
S → NP VP ←
NP → Prn N
NP → Prn
VP → V NP
VP → V
VP → V S
N → duck
V → duck
V → saw
Prn → I
Prn → she
Prn → her



Dealing with ambiguity

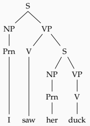
S → NP VP
NP → Prn N
NP → Prn
VP → V NP
VP → V
VP → V S ←
N → duck
V → duck
V → saw
Prn → I
Prn → she
Prn → her

Dealing with ambiguity



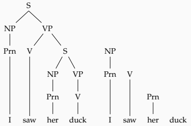
S → NP VP ←
NP → Prn N
NP → Prn
VP → V NP
VP → V
VP → V S
N → duck
V → duck
V → saw
Prn → I
Prn → she
Prn → her

Dealing with ambiguity



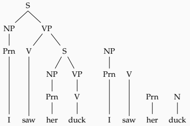
S → NP VP
NP → Prn N
NP → Prn
VP → V NP
VP → V
VP → V S
N → duck
V → duck
V → saw
Prn → I
Prn → she
Prn → her

Dealing with ambiguity



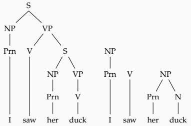
S → NP VP
NP → Prn N
NP → Prn
VP → V NP
VP → V
VP → V S
N → duck ←
V → duck
V → saw
Prn → I
Prn → she
Prn → her

Dealing with ambiguity



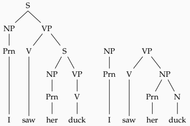
S → NP VP
NP → Prn N ←
NP → Prn
VP → V NP
VP → V
VP → V S
N → duck
V → duck
V → saw
Prn → I
Prn → she
Prn → her

Dealing with ambiguity



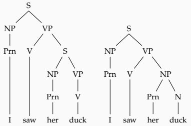
S → NP VP
NP → Prn N
NP → Prn
VP → V NP ←
VP → V
VP → V S
N → duck
V → duck
V → saw
Prn → I
Prn → she
Prn → her

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

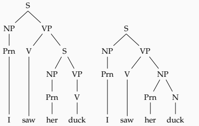
Dealing with ambiguity



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

How to represent multiple parses

parse forest grammar



$S_{0:4} \to NP_{0:1}\ VP_{1:4}$
$NP_{0:1} \to Prn_{0:1}$
$Prn_{0:1} \to I_{0:1}$
$VP_{1:4} \to V_{1:2}\ S_{2:4}$
$V_{1:2} \to saw_{1:2}$
$S_{2:4} \to Prn_{2:3}\ V_{3:4}$
$V_{3:4} \to duck_{3:4}$

$VP_{1:4} \to V_{1:2}\ NP_{2:4}$
$NP_{2:4} \to Prn_{2:3}\ N_{3:4}$

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 → B C
  - A → 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

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

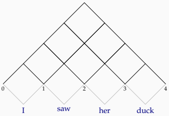
- S → Aux NP VP
  S → Aux NP VP ⇒ S → Aux X
  X → NP VP
- NP → the N
  NP → the N ⇒ NP → X N
  X → the
- VP → V
  VP → V ⇒ VP → bites

Converting to CNF

1. Eliminate the ϵ rules: if $A \to \epsilon$ is in the grammar
   - replace any rule $B \to \alpha A \beta$ with two rules
     $B \to \alpha \beta$
     $B \to \alpha A' \beta$
   - add $A' \to \alpha$ 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 $S \to \epsilon$)
2. Eliminate unit rules: for a rule $A \to B$
   - Replace the rule with $A \to \alpha_1 \mid \ldots \mid \alpha_n$, where $\alpha_1, \ldots, \alpha_n$ are all RHS or rule B
   - Remove the rule $A \to B$
   - Repeat the process until no unit rules remain
3. Binarize all the non-binary rules with non-terminal on the RHS: for a rule $A \to X_1\ X_2 \ldots X_n$:
   - Replace the rule with $A \to A_1\ X_2 \ldots X_n$, and add $A_1 \to X_1\ X_2$
   - Repeat the process until all new rules are binary

CKY demonstration

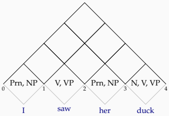
an ambiguous example



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

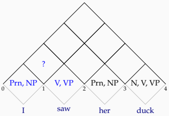


S	→	NP VP
NP	→	Prn N
VP	→	V NP
VP	→	V S
N	→	duck
VP	→	duck | saw
V	→	duck | saw
Prn	→	I | she | her
NP	→	I | she | her

CKY demonstration

an ambiguous example

S → NP VP

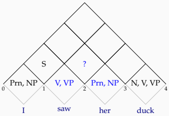


S	→	NP VP
NP	→	Prn N
VP	→	V NP
VP	→	V S
N	→	duck
VP	→	duck | saw
V	→	duck | saw
Prn	→	I | she | her
NP	→	I | she | her

CKY demonstration

an ambiguous example

VP → V NP

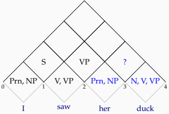


S	→	NP VP
NP	→	Prn N
VP	→	V NP
VP	→	V S
N	→	duck
VP	→	duck | saw
V	→	duck | saw
Prn	→	I | she | her
NP	→	I | she | her

CKY demonstration

an ambiguous example

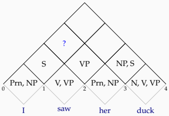
NP → Prn N
S → NP VP



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

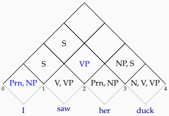


S	→	NP VP
NP	→	Prn N
VP	→	V NP
VP	→	V S
N	→	duck
VP	→	duck | saw
V	→	duck | saw
Prn	→	I | she | her
NP	→	I | she | her

CKY demonstration

an ambiguous example

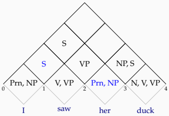
S → NP VP



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

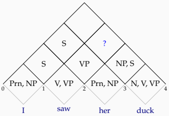
an ambiguous example



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

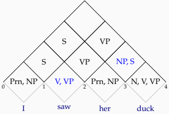


S	→	NP VP
NP	→	Prn N
VP	→	V NP
VP	→	V S
N	→	duck
VP	→	duck | saw
V	→	duck | saw
Prn	→	I | she | her
NP	→	I | she | her

CKY demonstration

an ambiguous example

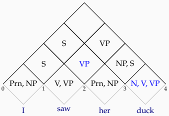
VP → V NP
VP → V S



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

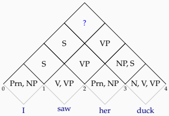
an ambiguous example



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

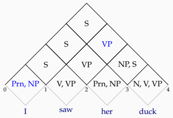


S	→	NP VP
NP	→	Prn N
VP	→	V NP
VP	→	V S
N	→	duck
VP	→	duck | saw
V	→	duck | saw
Prn	→	I | she | her
NP	→	I | she | her

CKY demonstration

an ambiguous example

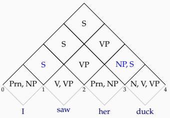
S → NP VP



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

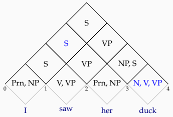
an ambiguous example



S → 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: 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

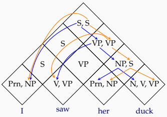
0 she 1 saw 2 her 3 duck 4

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)



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: **jurafsky2009**

Next:

- Top-down chart parsing: Earley algorithm
- Suggested reading:
  - **jurafsky2009**
  - **grune2008**

Acknowledgments, references, additional reading material