

Bottom-up Chart Parsing: the CKY algorithm

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

Çağrı Çöltekin
ccoltekin@ifa.uni-tuebingen.de

University of Tübingen
Seminar für Sprachwissenschaft

Winter Semester 2024/25

version: 007020-00020101-02

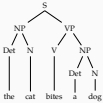
Parsing so far

- Parsing is the task of automatic syntactic analysis
- For most practical purposes, context-free grammars are the most useful formalism for parsing
 - 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

C. Çöltekin, ISL | University of Tübingen

Winter Semester 2024/25 3 / 3

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

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

C. Çöltekin, ISL | University of Tübingen

Winter Semester 2024/25 3 / 3

C. Çöltekin, ISL | University of Tübingen

Winter Semester 2024/25 3 / 3

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

C. Çöltekin, ISL | University of Tübingen

Winter Semester 2024/25 3 / 3

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

C. Çöltekin, ISL | University of Tübingen

Winter Semester 2024/25 3 / 3

Dealing with ambiguity

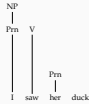


S → NP VP
NP → Prn N
NP → Prn
VP → V NP
VP → V
VP → V S
N → duck
V → duck
V → saw
Prn → I
Prn → she
Prn → her

C. Çöltekin, ISL | University of Tübingen

Winter Semester 2024/25 3 / 3

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

C. Çöltekin, ISL | University of Tübingen

Winter Semester 2024/25 3 / 3

Dealing with ambiguity

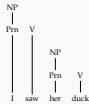


S → NP VP
NP → Prn N
NP → Prn
VP → V NP
VP → V
VP → V S
N → duck
V → duck
V → saw
Prn → I
Prn → she
Prn → her

C. Çöltekin, ISL | University of Tübingen

Winter Semester 2024/25 3 / 3

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

C. Çöltekin, ISL | University of Tübingen

Winter Semester 2024/25 3 / 3

Dealing with ambiguity

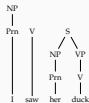


S → NP VP
NP → Prn N
NP → Prn
VP → V NP
VP → V
VP → V S
N → duck
V → duck
V → saw
Prn → I
Prn → she
Prn → her

C. Çöltekin, ISL | University of Tübingen

Winter Semester 2024/25 3 / 3

Dealing with ambiguity

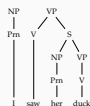


S → NP VP
NP → Prn N
NP → Prn
VP → V NP
VP → V
VP → V S
N → duck
V → duck
V → saw
Prn → I
Prn → she
Prn → her

C. Çöltekin, ISL | University of Tübingen

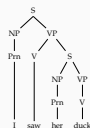
Winter Semester 2024/25 3 / 3

Dealing with ambiguity



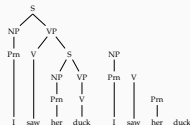
$S \rightarrow NP VP$
 $NP \rightarrow Pm N$
 $NP \rightarrow Pm$
 $VP \rightarrow V NP$
 $VP \rightarrow V$
 $VP \rightarrow V S$
 $N \rightarrow duck$
 $V \rightarrow duck$
 $V \rightarrow saw$
 $Pm \rightarrow I$
 $Pm \rightarrow she$
 $Pm \rightarrow her$

Dealing with ambiguity



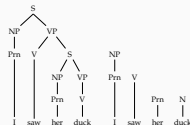
$S \rightarrow NP VP$
 $NP \rightarrow Pm N$
 $NP \rightarrow Pm$
 $VP \rightarrow V NP$
 $VP \rightarrow V$
 $VP \rightarrow V S$
 $N \rightarrow duck$
 $V \rightarrow duck$
 $V \rightarrow saw$
 $Pm \rightarrow I$
 $Pm \rightarrow she$
 $Pm \rightarrow her$

Dealing with ambiguity



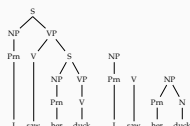
$S \rightarrow NP VP$
 $NP \rightarrow Pm N$
 $NP \rightarrow Pm$
 $VP \rightarrow V NP$
 $VP \rightarrow V$
 $VP \rightarrow V S$
 $N \rightarrow duck$
 $V \rightarrow duck$
 $V \rightarrow saw$
 $Pm \rightarrow I$
 $Pm \rightarrow she$
 $Pm \rightarrow her$

Dealing with ambiguity



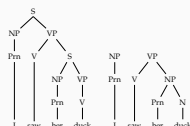
$S \rightarrow NP VP$
 $NP \rightarrow Pm N$
 $NP \rightarrow Pm$
 $VP \rightarrow V NP$
 $VP \rightarrow V$
 $VP \rightarrow V S$
 $N \rightarrow duck$
 $V \rightarrow duck$
 $V \rightarrow saw$
 $Pm \rightarrow I$
 $Pm \rightarrow she$
 $Pm \rightarrow her$

Dealing with ambiguity



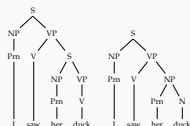
$S \rightarrow NP VP$
 $NP \rightarrow Pm N$
 $NP \rightarrow Pm$
 $VP \rightarrow V NP$
 $VP \rightarrow V$
 $VP \rightarrow V S$
 $N \rightarrow duck$
 $V \rightarrow duck$
 $V \rightarrow saw$
 $Pm \rightarrow I$
 $Pm \rightarrow she$
 $Pm \rightarrow her$

Dealing with ambiguity



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

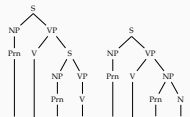
Dealing with ambiguity



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

How to represent multiple parses

parse forest grammar



$S_{0,4} \rightarrow NP_{0,1} VP_{1,4}$
 $NP_{0,1} \rightarrow Pm_{0,1}$
 $NP_{0,1} \rightarrow I_{0,1}$
 $V_{1,2} \rightarrow saw_{1,2}$
 $VP_{2,3} \rightarrow her_{2,3}$
 $VP_{1,4} \rightarrow V_{1,2} S_{2,4}$
 $S_{2,4} \rightarrow Pm_{2,3} V_{3,4}$
 $V_{3,4} \rightarrow duck_{3,4}$
 $VP_{1,4} \rightarrow V_{1,2} NP_{2,4}$
 $NP_{2,4} \rightarrow Pm_{2,3} N_{3,4}$
 $N_{3,4} \rightarrow duck_{3,4}$

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
 - $A \rightarrow BC$
 - $A \rightarrow a$
 where A, B, C are non-terminals and a is a terminal
- Any CFG can be converted to CNF
- Resulting grammar is *not* equivalent to the original grammar:
 - it generates/accepts the same language
 - but the derivations are different

Converting to CNF: example

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

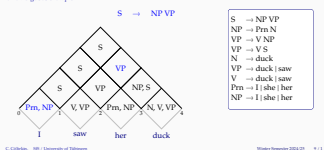
$S \rightarrow Aux NP VP$
 $S \rightarrow Aux NP VP \Rightarrow S \rightarrow Aux X$
 $NP \rightarrow the N$
 $NP \rightarrow the N \Rightarrow NP \rightarrow XN$
 $VP \rightarrow V$
 $VP \rightarrow V \Rightarrow VP \rightarrow bites$

Converting to CNF

- 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 $A' \rightarrow \epsilon$ for all A' (except ϵ) whose LHS is A
 - repeat the process for newly created ϵ rules
 - remove the rules with ϵ on the RHS (except $S \rightarrow \epsilon$)
- Eliminate unit rules: for a rule $A \rightarrow B$
 - Replace the rule with $A \rightarrow a_1 | \dots | a_n$, where $a_1, \dots, a_n$ are all RHS or rule B
 - Remove the rule $A \rightarrow B$
 - Repeat the process until no unit rules remain
- Binarize all the non-binary rules with non-terminal on the RHS: for a rule $A \rightarrow X_1 X_2 \dots X_n$:
 - Replace the rule with $A \rightarrow A_1 X_1 \dots X_{n-1}$, and add $A_1 \rightarrow X_2 X_3$
 - Repeat the process until all new rules are binary

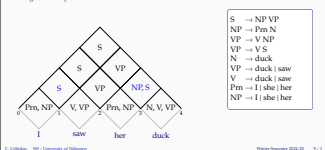
CKY demonstration

an ambiguous example



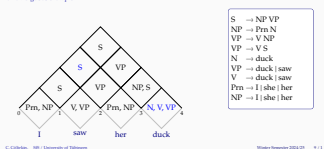
CKY demonstration

an ambiguous example



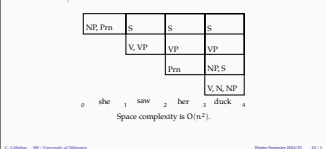
CKY demonstration

an ambiguous example



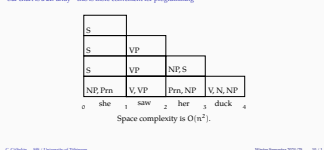
CKY demonstration: the chart

our chart is a 2D array



CKY demonstration: the chart

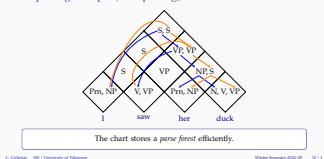
our chart is a 2D array – this is more convenient for programming



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)



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:
 - [grune2008](#)

Acknowledgments, references, additional reading material