Automata Theory and Formal Languages

Summer Term 2026

11th Lecture

CYK Algorithm

Prof. Dr. Matthias Mnich *Institute for Algorithms and Complexity*

CYK Algorithm

Decision Problem

- The decision problem is decidable for type 1,2,3 grammar but the algorithm had an exponential runtime.

Decision Problem

- The decision problem is decidable for type 1,2,3 grammar but the algorithm had an exponential runtime.

- Consider that you have a C source file of length N and it requires 2^N ms time to decide if that program is a valid C program. Length here refers to the number of symbols / token (if / for ... are one symbol).

Decision Problem

- The decision problem is decidable for type 1,2,3 grammar but the algorithm had an exponential runtime.
- Consider that you have a C source file of length N and it requires 2^N ms time to decide if that program is a valid C program. Length here refers to the number of symbols / token (if / for ... are one symbol).
- If your program has 10 symbols, it would require 2^{10} ms ≈ 1 sec to solve the decision problem

Decision Problem

- The decision problem is decidable for type 1,2,3 grammar but the algorithm had an exponential runtime.
- Consider that you have a C source file of length N and it requires 2^N ms time to decide if that program is a valid C program. Length here refers to the number of symbols / token (if / for ... are one symbol).
- If your program has 10 symbols, it would require 2^{10} ms ≈ 1 sec to solve the decision problem
- How long would it take if the program has 30 symbols?

Decision Problem

- The decision problem is decidable for type 1,2,3 grammar but the algorithm had an exponential runtime.
- Consider that you have a C source file of length N and it requires 2^N ms time to decide if that program is a valid C program. Length here refers to the number of symbols / token (if / for ... are one symbol).
- If your program has 10 symbols, it would require 2^{10} ms ≈ 1 sec to solve the decision problem
- How long would it take if the program has 30 symbols? 2^{30} ms ≈ 12 days

CYK Algorithm

- The CYK algorithm solves the decision problem much faster for a context-free grammar.

CYK Algorithm

- The CYK algorithm solves the decision problem much faster for a context-free grammar.
- It is named after its inventors: Cocke, Younger, Kasami

CYK Algorithm

- The CYK algorithm solves the decision problem much faster for a context-free grammar.
- It is named after its inventors: Cocke, Younger, Kasami
- It expects the grammar to be in Chomsky normal form

Idea behind CYK Algorithm

- A string of length 1 ($x = a$) can only be derived by a rule $A \to a$.

Idea behind CYK Algorithm

- A string of length 1 ($x = a$) can only be derived by a rule $A \to a$.
- For

$$x = a_1 a_2 \dots a_n \quad n \geq 2$$

one can derive x only if at some point a rule $A \to BC$ has been applied:

$$x = \underbrace{a_1 a_2 \dots a_k}_{B} \underbrace{a_{k+1} \dots a_n}_{C}$$

Idea behind CYK Algorithm

- A string of length 1 ($x = a$) can only be derived by a rule $A \to a$.
- For

$$x = a_1 a_2 \dots a_n \quad n \geq 2$$

one can derive x only if at some point a rule $A \to BC$ has been applied:

$$x = \underbrace{a_1 a_2 \dots a_k}_{B} \underbrace{a_{k+1} \dots a_n}_{C}$$

- Thus we can break down the decision problem of length n into two decision problems of lengths k and $n - k$.

Idea behind CYK Algorithm

Issue

k is not known

Idea behind CYK Algorithm

Issue

k is not known

Solution

Systematically try out all possible k. This approach is called *dynamic programming*.

CYK Algorithm

Notation

$x_{i,j}$ should be the substring of x that starts at index i and has length j.

CYK Algorithm

Notation

$x_{i,j}$ should be the substring of x that starts at index i and has length j.

The string x that that is split after the k-th character can thus be written as

$$x = x_{1,k}x_{k+1,n-k}$$

CYK Algorithm

Notation

$x_{i,j}$ should be the substring of x that starts at index i and has length j.

The string x that that is split after the k-th character can thus be written as

$$x = x_{1,k}x_{k+1,n-k}$$

Remark

The algorithm uses a table $T[1 \ldots n, 1 \ldots n]$ where just one triangle is used.

CYK Algorithm

Input: $x = a_1 a_2 \ldots a_n$ and grammar G

for $i := 1$ to n **do**
  $T[i,1] = \{A \in V \mid (A \to a_i) \in P\}$
end for
for $j := 2$ to n **do**
  for $i := 1$ to $n + 1 - j$ **do**
    $T[i,j] = \emptyset$
    for $k := 1$ to $j - 1$ **do**
      $T[i,j] = T[i,j] \cup \{A \in V \mid (A \to BC) \in P \wedge$
        $B \in T[i,k] \wedge C \in T[i+k, j-k]\}$
    end for
  end for
end for
$\vdots$

CYK Algorithm

$\vdots$

if $S \in T[1, n]$ **then**
  return true
else
  return false
end if

CYK Algorithm

Runtime

The runtime of the CYK algorithm is $\mathcal{O}(n^3)$ since we have three nested for loops of which each requires $\mathcal{O}(n)$ passes. This is much faster then the brute-force algorithm that we discussed for type 1 languages, which required $\mathcal{O}(\exp(n))$ steps.

Example – CYK Algorithm

Language

$L = \{a^n b^n c^m \mid n, m \geq 1\}$

Grammar

$$S \to AB$$
$$A \to CD \mid CF$$
$$B \to c \mid EB$$
$$C \to a$$
$$D \to b$$
$$E \to c$$
$$F \to AD$$

Example – CYK Algorithm

Input: $x = aaabbbcc$

Example – CYK Algorithm

Input: $x = aaabbbcc$

a	a	a	b	b	b	c	c
C	C	C	D	D	D	E, B	E, B
-	-	A	-	-	-	B	
-	-	F	-	-	-		
-	A	-	-	-			
-	F	-	-				
A	-	-					
S	-						
S							

Example – CYK Algorithm

Input: $x = aaabbbcc$

a	a	a	b	b	b	c	c
C	C	C	D	D	D	E, B	E, B
-	-	A	-	-	-	B	
-	-	F	-	-	-		
-	A	-	-	-			
-	F	-	-				
A	-	-					
S	-						
S							

Since $S \in T[1, 8]$ the string x can be generated by the grammar G.