COMS3261: Computer Science Theory

Fall 2013

Mihalis Yannakakis

Lecture 16, 10/30/13

MEMBERSHIP PROBLEM (Parsing)

- Membership problem for a CFG G
- Input: String w
- Question: w $\in$L(G)?

  (and if yes, construct a derivation or a parse tree)
- We measure the time complexity primarily as a function of the length n=|w| of w.
- Size of grammar G regarded as fixed.

- Not obvious
- "Brute force" attempts: Try to generate all derivations, all parse trees - too many, infinite
- Construct PDA, try to generate systematically all computations – same problems

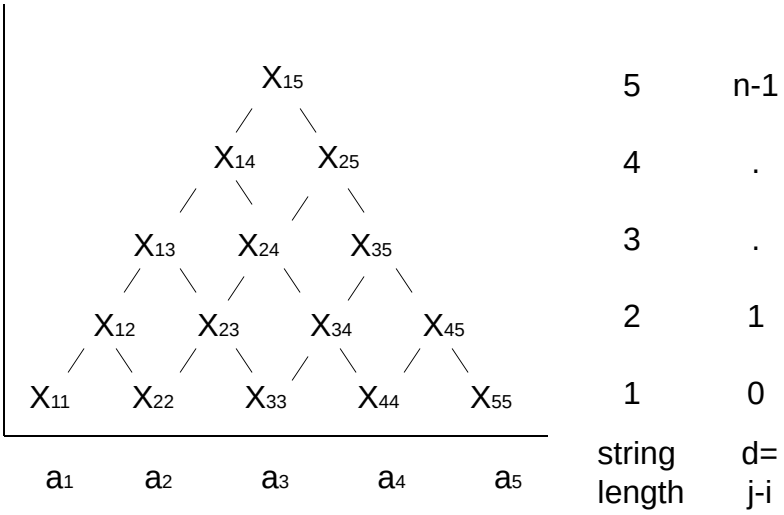
CYK (Cocke-Younger-Kasami) algorithm

- CFG in Chomsky Normal Form G=(V,T,P,S)
- Dynamic Programming algorithm

- Input: string $w = a_1 \ldots a_n$

- For each pair of indices i,j with $1 \le i < j \le n$, compute set $X_{ij} = \{ A \in V \mid A \Rightarrow^* w[i,j] = a_i \ldots a_j \}$
- Compute them in order of increasing length of the substring w[i,j], i.e. increasing difference d= j-i and fill table
- $w \in L$ iff $S \in X_{1n}$

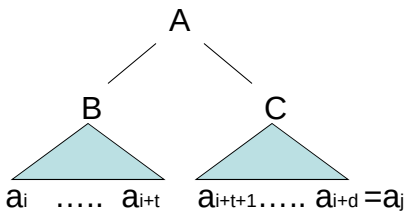
CYK: Fill a table



CYK Algorithm

- for i=1 to n $X_{ii} = \{ A \mid A \rightarrow a_i \}$
- for d=1 to n-1
  for i=1 to n-d
  { $X_{i,i+d} := \varnothing$
  for t=0 to d-1
  $X_{i,i+d} := X_{i,i+d} \cup \{ A \mid A \rightarrow BC, B \in X_{i,i+t} , C \in X_{i+t+1,i+d} \}$
  }
- if ($S \in X_{1,n}$) then return Yes else No

• Can record additional information why each variable was included in each set X_{ij} to construct a parse tree at the end



- Correctness proof: By induction on d =j-i
- Basis: d=0. By initialization, If $A \Rightarrow^* a$ it must be a single production because of CNF
- Induction step: By picture above. If $A \Rightarrow^* a_i \ldots a_{i+d}$, then must start with a production $A \rightarrow BC$, and $B \Rightarrow^* a_i \ldots a_{i+t}$ for some t=0,…,d-1, and $C \Rightarrow^* a_{i+t+1} \ldots a_{i+d}$

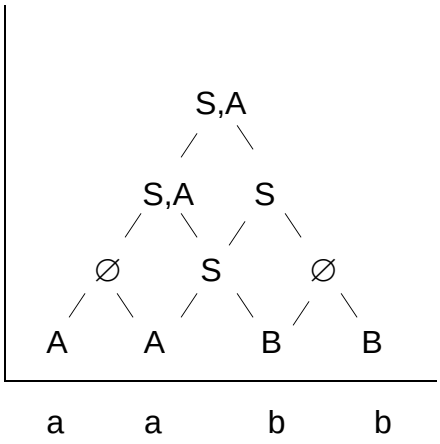
- Time Complexity: $O(n^3)$
  (Linear in size of CNF grammar)

Example

$S \rightarrow AS \mid SB \mid AB$

$A \rightarrow AS \mid a$

$B \rightarrow b$



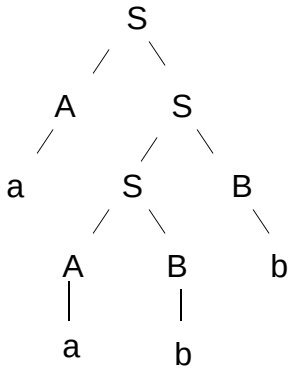
$L(G) = a^+b^+$

Example ctd: Parse Tree

$S \rightarrow AS \mid SB \mid AB$

$A \rightarrow AS \mid a$

$B \rightarrow b$



$L(G) = a^+b^+$

UNDECIDABLE PROBLEMS FOR CFLs

- $L = \Sigma^*$? (Universality problem)
- $L_1 = L_2$? (follows from 1)
- Decidable for DPDAs – very difficult result
- $L_1 \cap L_2 = \varnothing$?
- Is a given CFG G ambiguous ?
- Is L inherently ambiguous ?
- L given by CFG or PDA, it does not matter because we can convert between the two representations

Computability

- There are problems that computers cannot solve.
- Example: Is a program correct ?
- Can check for syntactic correctness (eg. valid C program), but functional? i.e., does it do what it is supposed to do ?
- Example: `hello world' program
- main()
  { printf("hello, world\n"); }

clear – ok, it prints hello world

- How about { ….. Complicated stuff
  printf("hello, world\n");
  ….. Complicated stuff }

Fermat theorem example

Fermat's last theorem: For $n \geq 3$, there are no positive integers x,y,z such that $x^n+y^n = z^n$

Program: For k=3,4,… try every value of n,x,y,z up to k;

if $x^n+y^n = z^n$ print "hello, world" and stop, else proceed to next value of k.

Will the program print "hello, world" or run forever?

$\Leftrightarrow$ Fermat's last theorem (so runs forever)

So, a program to check whether a program is `correct' (prints "hello, world", terminates ..) at least as hard as hard number theory questions.

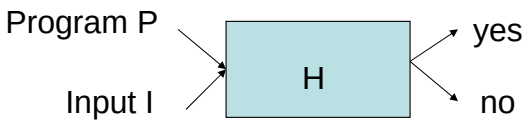
Self-referential paradoxes

(Liar) Paradox: "This sentence is false."

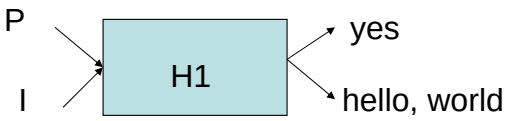
- Is this sentence true or false?
  - If true then false.
  - If false then true.

Self-reference argument for uncomputability

- Suppose there is a ‘hello, world’ tester program H that tests whether a given program P with input I prints “hello world” and prints accordingly `yes’ or `no’

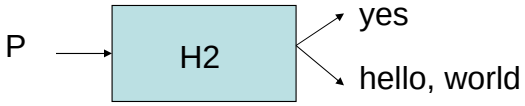


Modify H so it prints “hello world” instead of “no”

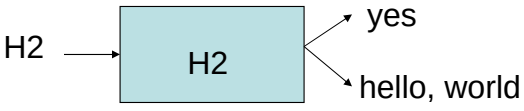


Self-reference argument ctd

- Program H2 on input P simulates H1 on input P, P
- Determines whether P on input P prints ‘hello, world’



- Feed input H2 to program H2



What will happen? If H2 prints yes, then it should have printed `hello world’, and vice-versa ⇒ H2 does not exist ⇒ H does not exist