

Algorithms for CFLs

Thursday, February 25, 2021 11:41 AM

$$\Sigma = \{a, b\}$$

$$L = \{ww \mid w \in \Sigma^*\} \rightarrow \underline{\text{not}} \text{ a CFL}$$

\bar{L} all words that are not in L

examples : $abba \in \bar{L}$, $aaa \in \bar{L}$
 $\epsilon \in L$, $abab \in L$, $abaaba \in L$

I will describe a PDA informally

A word is in \bar{L} if

(i) it has odd length \rightarrow can be checked with a DFA

OR (ii) it has even length and there is a pair of corresponding positions with different letters.

$$\underbrace{x_1 x_2 \dots x_{i-1}}_{(i-1) \text{ letters}} \quad \boxed{x_i} \quad \underbrace{x_{i+1} \dots x_n}_{(n-i)} \quad \underbrace{y_1 \dots y_{i-1}}_{(i-1)} \quad \boxed{y_i} \quad \underbrace{y_{i+1} \dots y_n}_{n-i}$$

x_i & y_i are both in position i of the two halves of the word.

The PDA will guess which pair of positions is mismatched.

It reads x_1, x_2, \dots, x_{i-1} and pushes them on the stack. Then it memorises x_i . Then it has to find y_i ; keep reading x_{i+1}, x_{i+2}, \dots upto x_n and pop the stack. When the stack is empty start pushing symbols onto the stack. Now guess where y_i is : compare y_i with the memorized $x_i \rightarrow$ if they are the same reject. If $x_i \neq y_i$ we need to verify that they were really at corresponding positions. To check this we read the rest of the word and pop the stack as we go. If the word ends exactly when the stack becomes empty x_i & y_i really were at corresponding positions.

Is there a CFG for \bar{L} ?

$$V = \{ \underline{S}, A, B, c \} \quad T = \Sigma = \{ a, b \}$$

$$S \rightarrow AB \mid BA \mid \underline{A} \mid \underline{B}$$

$$A \rightarrow \underline{CAC} | a \quad B \rightarrow C B C | b$$

$$C \rightarrow a | b$$

A produces all strings of odd length with 'a' as the middle letter

B produces - - - - with 'b' as the middle letter

$S \rightarrow AB$ and $S \rightarrow BA$ produce even length strings

$$A \xrightarrow{*} x a y \quad |x| = |y| = n$$

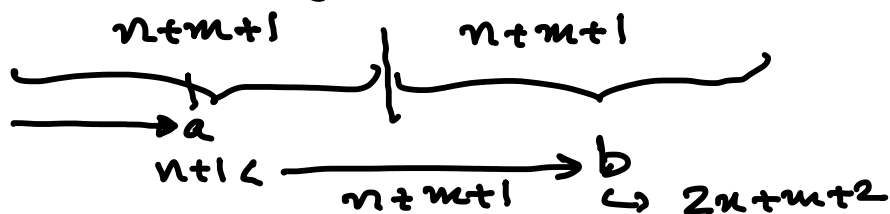
$$B \xrightarrow{*} u b v \quad |u| = |v| = m$$

$$S \xrightarrow{*} A B \xrightarrow{*} x a y u b v$$

a is at position $n+1$

b is at position $\underline{2n+m+2}$

Total length $2n+2m+2$



Hence the a & b are guaranteed to be at corresponding positions.

The two halves are guaranteed to be different so the word $\in L$.

An algorithm to determine whether

$$w \in L(G)$$

$w \rightarrow$ given a word

$G \rightarrow$ a given context free grammar

We will assume G is in Chomsky normal form

$$A \rightarrow BC \text{ or } A \rightarrow a$$

Alg is $O(n^3)$ where n is $|w|$.

Assignment Project Exam Help

DYNAMIC PROGRAMMING

Given $G = (V, \Sigma, P, S)$

Input $w = a_1 \dots a_n$, each $a_i \in \Sigma$

Work bottom-up to construct partial derivations of pieces of w & then eventually all of w .

Inductively define a 2-indexed family of subsets of V .

$$j \geq i \quad X_{ij} := \{A \in V \mid A \xrightarrow{*} a_i \dots a_j\}$$

$$\text{BASE } X_{ii} = \{A \in V \mid A \rightarrow a_i\} \quad X_{11}, X_{22}, X_{33}, \dots$$

Then I construct $X_{12}, X_{23}, X_{34}, \dots$ one row

Then I construct $X_{13}, X_{24}, X_{35}, \dots$ next row

We will put all these sets in a table
each of these will be in a row

X_{ij} will be in row $(j-i)+1$

When I want to compute X_{ij} I will
know $X_{ik} \& X_{kj}$ for all $i \leq k \leq j$

If $B \in X_{ik} \& C \in X_{(k+1)j}$

and $A \rightarrow BC$ is a production rule

Then $A \in X_{ij}$

$B \xrightarrow{*} a_i \dots a_k \quad C \xrightarrow{*} a_{(k+1)} \dots a_j$

$A \rightarrow BC \xrightarrow{*} a_i \dots a_j$

In the end we want to know

$S \in X_{1n} ?$

$S \rightarrow AB \mid BC \quad A \rightarrow BA \mid a \quad \underline{B \rightarrow CC \mid b}$

$C \rightarrow AB \mid a$

Want to know $baab \in L(G) ?$

5	\emptyset	$\{S, \Lambda\}$
4	\emptyset	$\{S, \Lambda\}$

3	\emptyset	$\{B\}$ ^{x=4}	$\{B\}$	
2	$\{A, S\}$	$\{B\}$	$\{S, C\}$	$\{A, S\}$
1	$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$
	b	a	a	b
				a

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder