Problem

Recall the CFG G_4 that we gave in Example 2.4. For convenience, let's rename its variables with single letters as follows.

$$E \to E + T \mid T$$
$$T \to T \times F \mid F$$
$$F \to (E) \mid \mathtt{a}$$

Give parse trees and derivations for each string.

a. a

b. a+a

c. a+a+a

d. ((a))

Step-by-step solution

Step 1 of 8

Given Grammar G_4 is

$E \to E+T \mid T$
$T \to T \times F \mid F$
$F \to (E) \mid a$

Derivation: The sequence of substitutions to obtain a string is called a *derivation*.

Parse Tree: The pictorial representation of derivation of a string is a *parse tree*.

Comment

Step 2 of 8

a)

The parse tree to generate string a is as follows:

Comment

Step 3 of 8

The derivation for the string a is as follows:

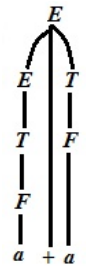
$E \Rightarrow T$
$E \Rightarrow F$
$E \Rightarrow a$

Comment

Step 4 of 8

b)

The parse tree to generate string $a+a$ is as follows:



Comment

Step 5 of 8

The derivation for the string $a+a$ is as follows:

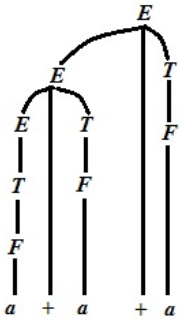
$E \Rightarrow E+T$
$E \Rightarrow T+T$
$E \Rightarrow F+T$
$E \Rightarrow a+T$
$E \Rightarrow a+F$
$E \Rightarrow a+a$

Comment

Step 6 of 8

c)

The parse tree to generate string $a+a+a$ is as follows:



The derivation for the string $a+a+a$ is as follows:

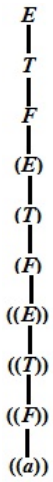
$E \Rightarrow E+T$
$E \Rightarrow E+T+T$
$E \Rightarrow T+T+T$
$E \Rightarrow F+T+T$
$E \Rightarrow a+T+T$
$E \Rightarrow a+F+T$
$E \Rightarrow a+a+T$
$E \Rightarrow a+a+F$
$E \Rightarrow a+a+a$

Comment

Step 7 of 8

d)

The parse tree to generate string $((a))$ is as follows:



Comment

Step 8 of 8

The derivation for the string $((a))$ is as follows:

$E \Rightarrow T$
$E \Rightarrow F$
$E \Rightarrow (E)$
$E \Rightarrow (T)$
$E \Rightarrow (F)$
$E \Rightarrow ((E))$
$E \Rightarrow ((T))$
$E \Rightarrow ((F))$
$E \Rightarrow ((a))$

Comments (2)

Problem

a. Use the languages

$$A = \{\mathtt{a}^m\mathtt{b}^n\mathtt{c}^n \mid m, n \geq 0\} \text{ and } B = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^m \mid m, n \geq 0\}$$

together with Example 2.36 to show that the class of context-free languages is not closed under intersection.

b. Use part (a) and DeMorgan's law (Theorem 0.20) to show that the class of context-free languages is not closed under complementation.

THEOREM 0.20

For any two sets A and B, $\overline{A \cup B} = \overline{A} \cap \overline{B}$.

Step-by-step solution

Step 1 of 4

Context Free languages

a) Given the languages are

$A = \{a^m b^n c^n \mid m, n \geq 0\}$ and

$B = \{a^n b^n c^m \mid m, n \geq 0\}$

Now we will show that both A and B are context-free languages.

In order to show, let us construct grammar that recognizes A.

$S \to UT$

$U \to aU \mid \varepsilon$

$T \to bTc \mid \varepsilon$

Observing the above grammar we can say that the language A is a context-free language.

Let us construct grammar that recognizes B.

$S \to TU$

$T \to aTb \mid \varepsilon$

$U \to cU \mid \varepsilon$

Observing the above grammar we can say that the language B is a context-free language.

Hence both A and B are context-free languages.

Comment

Step 2 of 4

Consider $A \cap B = \{a^n b^n c^n \mid n \geq 0\}$.

Now check whether the language $A \cap B$ is a context-free or not using pumping lemma.

Let us assume that $A \cap B$ is a context-free language.

Pumping lemma states that every context-free language has a special value called *pumping length* such that all longer strings in the language can be "pumped",

let p be the pumping length for $A \cap B$.

Consider a string $s = a^p b^p c^p$.

Clearly s is a member of $A \cap B$ and of length at least p.

Now we prove that one condition of pumping lemma violated by proving s cannot be pumped.

If we divide s into $wxyz$, condition 2 stipulates that either v or y is non-empty.

Now consider one of the two cases, depending on whether substring v and y contains more than one type of alphabet symbol.

1. If both v and y contain only one type of symbol, v doesn't contain both a's and b's or both b's and c's, and the same holds for y. Here the string uv^2xy^2z cannot contain equal number of a's, b's and c's. Therefore it cannot be a member of $A \cap B$ which violates the first condition of the pumping lemma and thus is a contradiction to our hypothesis.

2. If either v or y contain more than one type of symbol uv^2xy^2z may contain equal number of the three alphabet symbols but not in the correct order. Hence it cannot be a member of $A \cap B$ and thus is a contradiction to our hypothesis.

One of the above two case must occur. However, both the cases raised contradiction. This is because of our assumption $A \cap B$ is a context-free language.

Hence our assumption is false and $A \cap B$ is not a context-free language.

Comments (4)

Step 3 of 4

Hence, we have A and B are context-free languages and $A \cap B$ is not a context-free language. So we can say that the language obtained by intersection of two context-free languages A and B is not a context a context-free language.

Therefore, the languages A and B are not closed under intersection.

Comment

Step 4 of 4

b) Using DeMorgan's law we will show that the languages A and B is not closed under complementation.

DeMorgan's law states that for any two sets A and B, $\overline{A \cup B} = \overline{A} \cap \overline{B}$.

We have A and B are two arbitrary context-free languages.

Let these languages are represented in 4-tuple form as $A = (V_1, \Sigma, R_1, S_1)$ and $B = (V_2, \Sigma, R_2, S_2)$ where

- V_1, V_2 are finite set of variables of A and B respectively.
- Σ is finite set, disjoint from V_1, V_2 are terminals of A and B respectively.
- R_1, R_2 are finite set of rules of A and B respectively.
- $S_1 \in V_1, S_2 \in V_2$ are the start variables of A and B respectively.

Now construct a grammar G that recognizes $A \cup B$.

So $G = (V, \Sigma, R, S)$ where

- $V = V_1 \cup V_2$
- $R = R_1 \cup R_2 \cup \{S \to S_1, S \to S_2\}$. Here, R_1 and R_2 are disjoint.

Now we have to show that A and B are not closed under complementation.

Let us assume that A and B are closed under complementation.

Since, A and B are context-free languages, then $\overline{A}$ and $\overline{B}$ are also context-free-languages. We know that the context-free-languages are closed under union.

So, $\overline{A} \cup \overline{B}$ is closed. Hence $\overline{A} \cup \overline{B}$ is a context-free-language.

Since, $\overline{A} \cup \overline{B}$ is a context-free-language, we have $\overline{\overline{A} \cup \overline{B}}$ is a context-free-language.

Applying DeMorgan's law we get $\overline{\overline{A} \cup \overline{B}} = A \cap B$.

Hence $A \cap B$ is a context-free-language which is a contradiction to part(a).

This contradiction occurred because our assumption is wrong.

Hence A and B are not closed under complementation.

Therefore, class of context-free-languages is not closed under complementation.

Comments (2)

Problem

Answer each part for the following context-free grammar G.

$$
\begin{aligned}
R &\to XRX \mid S \\
S &\to \mathtt{a}T\mathtt{b} \mid \mathtt{b}T\mathtt{a} \\
T &\to XTX \mid X \mid \varepsilon \\
X &\to \mathtt{a} \mid \mathtt{b}
\end{aligned}
$$

a. What are the variables of G?
b. What are the terminals of G?
c. Which is the start variable of G?
d. Give three strings in $L(G)$.
e. Give three strings *not* in $L(G)$.
f. True or False: $T \Rightarrow \mathtt{aba}$.
g. True or False: $T \overset{*}{\Rightarrow} \mathtt{aba}$.
h. True or False: $T \Rightarrow T$.
i. True or False: $T \overset{*}{\Rightarrow} T$.
j. True or False: $XXX \overset{*}{\Rightarrow} \mathtt{aba}$.
k. True or False: $X \overset{*}{\Rightarrow} \mathtt{aba}$.
l. True or False: $T \overset{*}{\Rightarrow} XX$.
m. True or False: $T \overset{*}{\Rightarrow} XXX$.
n. True or False: $S \overset{*}{\Rightarrow} \varepsilon$.
o. Give a description in English of $L(G)$.

Step-by-step solution

Step 1 of 16

Given the context – free grammar G is

$R \to XRX \mid S$
$S \to aTb \mid bTa$
$T \to XTX \mid X \mid \varepsilon$
$X \to a \mid b$

Comment

Step 2 of 16

a. The variables of G are R, S, T and X

Variables are the non-terminal symbols that appear in the rules of the grammar.

Comment

Step 3 of 16

b. The terminals of G are a, b.

Terminals are the terminal symbols that appear in the rules of the grammar.

Comments (2)

Step 4 of 16

c. The start variable of G is R.

$R \to XRX \mid S$

Start variable is a variable usually occurs on the left – hand side of the topmost rule.

Comments (3)

Step 5 of 16

d. Case1: Consider the rule $R \to S$

Substitute S with rule $S \to aTb$

$R \to aTb$

Substitute T with rule $T \to \varepsilon$.

$R \to a \in b$

$R \to ab$

Case2: Consider the rule $R \to S$

Substitute S with rule $S \to bTa$

$R \to bTa$

Substitute T with rule $T \to \varepsilon$.

$R \to b \in a$

$R \to ba$

Case3: Consider the rule $R \to S$

Substitute S with rule $S \to aTb$

$R \to aTb$

Substitute T with rule $T \to X$.

$R \to aXb$

Substitute X with rule $X \to a$.

$R \to aab$

Therefore, the 3 strings in $L(G)$ are ab, ba and aab.

Comment

Step 6 of 16

e. The three strings not in $L(G)$ are aba, b and ε. Since these strings cannot be derived from the given grammar *G*.

Comments (6)

Step 7 of 16

f. False

$T \Rightarrow aba$, the string cannot be derived using G.

Comments (1)

Step 8 of 16

g. True

$T \overset{*}{\Rightarrow} aba$, the string can be derived using G.

Comments (16)

Step 9 of 16

h. False

$T \Rightarrow T$, the string cannot be derived using G.

Comment

Step 10 of 16

i. True

$T \overset{*}{\Rightarrow} T$, the string can be derived using G.

Comments (15)

Step 11 of 16

j. True

$XXX \overset{*}{\Rightarrow} aba$, the string can be derived using G.

Comment

Step 12 of 16

k. False

$X \overset{*}{\Rightarrow} aba$, the string cannot be derived using G.

Comments (2)

Step 13 of 16

l. True

$T \overset{*}{\Rightarrow} XX$, the string can be derived using G.

Comments (2)

Step 14 of 16

m. True

$T \overset{*}{\Rightarrow} XXX$, the string can be derived using G.

Comments (1)

Step 15 of 16

n. False

$S \overset{*}{\Rightarrow} \varepsilon$, the string cannot be derived using G.

Comment

Step 16 of 16

o. $L(G)$ consists of all strings that are not palindromes and are formed over terminal symbols a and b.

Comments (3)

Problem

Give context-free grammars that generate the following languages. In all parts, the alphabet ∑ is {0,1}.

A**a.** {wl w contains at least three 1s}

b. {wl w starts and ends with the same symbol}

c. {wl the length of w is odd}

A**d.** {wl the length of w is odd and its middle symbol is a 0}

e. {wl w = w^R, that is, w is a palindrome}

f. The empty set

Step-by-step solution

Step 1 of 6

The alphabet $\sum$ is given by $\{0, 1\}$

(a)

The context free grammar that generates the language

$\{\omega \mid \omega \text{contains at least three 1s}\}$ is given by

$S \to P1P1P1P$

$P \to 0P \mid 1P \mid \varepsilon$

Comments (4)

Step 2 of 6

(b)

The context free grammar that generates the language $\{\omega \mid \omega \text{starts and ends with the same symbol}\}$ is given by

$S \to 0P0 \mid 1P1 \mid 0 \mid 1$

$P \to 0P \mid 1P \mid \varepsilon$

Comments (3)

Step 3 of 6

(c)

The context free grammar that generates the language

$\{\omega \mid \text{the length of } \omega \text{is odd}\}$ is given by

$S \to 0 \mid 1 \mid 00S \mid 01S \mid 10S \mid 11S$

(or)

$S \to 0 \mid 1 \mid 0S0 \mid 0S1 \mid 1S0 \mid 1S1$

Comments (14)

Step 4 of 6

(d)

The context free grammar that generates the language $\{\omega \mid \text{the length of } \omega \text{ is odd and its middle symbol is a } 0\}$ is given by

$S \to 0 \mid 0S0 \mid 0S1 \mid 1S0 \mid 1S1$

Comments (5)

Step 5 of 6

(e)

The context free grammar that generates the language $\{\omega \mid \omega = \omega^R, \text{that is, } \omega \text{ is a palindrome}\}$ is given by

$S \to 0 \mid 1 \mid 0S0 \mid 1S1 \mid \varepsilon$

Comments (12)

Step 6 of 6

(f)

The context free grammar that generates the language

$\{\ \}$ is given by

$S \to S$

Comments (2)

Problem

Give informal descriptions and state diagrams of pushdown automata for the languages in Exercise 2.4.

Step-by-step solution

Step 1 of 10

Given:

$L = \{w \mid w\}$, which contains minimum three 1's

Context free grammar of the above language is as shown:

$S \to R1R1R1R$

$R \to 0R \mid 1R \mid \in$

Informal Description:

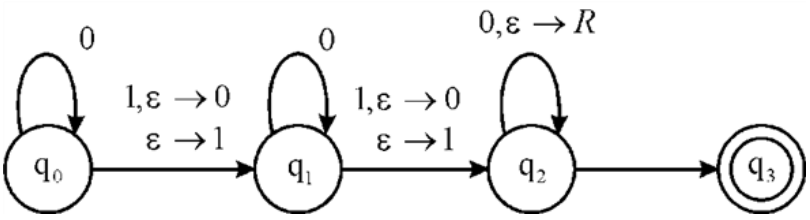
In the above grammar there is no need of stack. So, in the above grammar only the input is being read and that input is accepted only when there is an atleast three 1's. If atleast three 1's are present in any string then push down automata will accept that grammar otherwise it will reject.

Pushdown Automata:

Pushdown automata of the informal description is as shown:

Comment

Step 2 of 10



Comments (6)

Step 3 of 10

In the above pushdown automata, the state q_0 is the initial state and q_3 is the final state. Transition is done from q_0 to q_3 and string is accepted by the push down automata if there is three 1's otherwise string will not be accepted.

Comment

Step 4 of 10

Given:

$L = \{w \mid w\}$, whose starting and ending symbol are same

Context free grammar of the above language is as shown:

$S \to 0A0 \mid 1A1 \mid 0 \mid 1$

$A \to 0A \mid 1A \mid \in$

Informal Description:

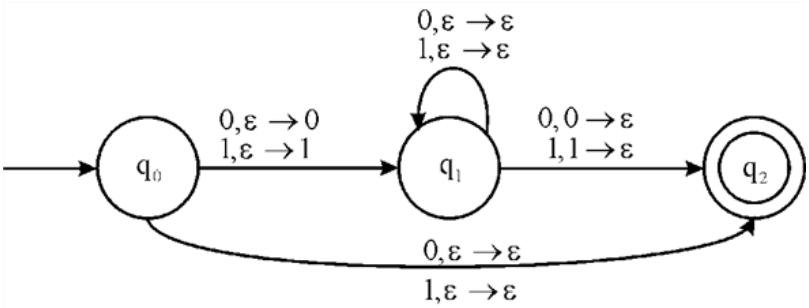
• In the above grammar user made an assumption that there is only one symbol in the string with can be accepted without the use of the stack.

• If there is more than one symbol in the string then all the symbol are sequentially put into the stack.

• Every time, the user identifies whether it is the last symbol which is to be read. Suppose it is the last symbol which is to be read and this symbol matches with the symbol which is present on the stack and if all the input has been retrieved and no input is being left then the user accepts the input otherwise pushdown automata will reject that input.

Pushdown Automata:

Pushdown automata of the informal description is as shown:



In the above push down automata, the state q_0 is the starting state and q_2 is the final state. In the first step user push the element into the push down automata after user pop the symbol from the stack and check whether it match with the final state or not. If it matches with the final stage then string is accepted.

Comments (6)

Step 5 of 10

Given: q_0

Context free grammar of the above language is as shown:

$S \to 0A \mid 1A$

$A \to 0S \mid 1S \mid \in$

Informal Description:

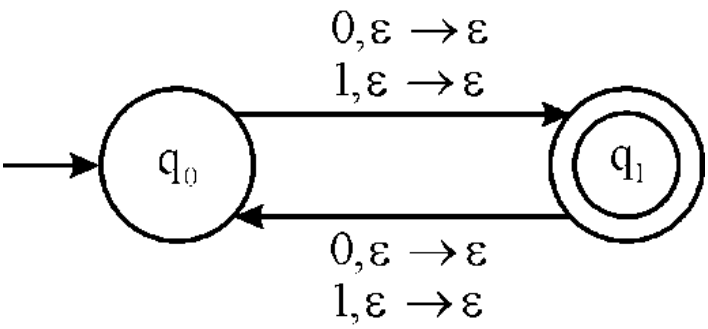
In the above grammar there is no need of stack. So, in the above grammar only the input is being read and that input is accepted only when there is an odd length. If the first symbol is being read or other symbol is being read then it is the final symbol read.

Pushdown Automata:

Pushdown automata of the informal description is as shown:

Comment

Step 6 of 10



In the above pushdown automata, the state q_0 is the initial stage and q_1 is the final stage.

Comment

Step 7 of 10

Given:

$L = \{w \mid \text{the length of } w \text{ is odd and its middle symbol is } 0\}$

Context free grammar of the above language is as shown:

$S \to 0S0 \mid 0S1 \mid 1S0 \mid 1S1 \mid 0$

Informal Description:

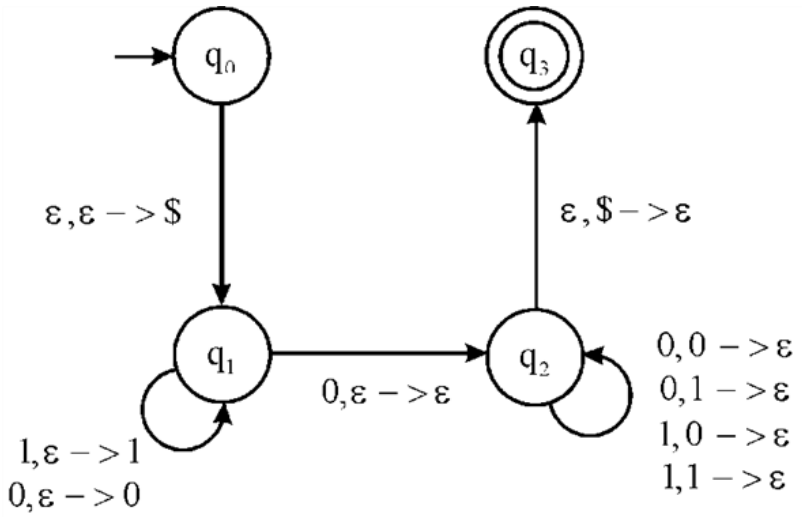
• In the above grammar input is scanned by the push down automata and symbol is placed into the stack. There is some point in the push down automata where it guesses the input which is placed in the middle.

• It checks whether the symbol which is present in the middle contains 0 or not. If 0 is found as a middle element then all the string are scan sequentially and every time character is pop from the stack which has been scanned.

• If the stack after poping is empty and there is no input which is to be read by the stack then it checks whether 0 is the middle symbol in the string. If it is then the string is accepted otherwise the pushdown automata will reject the string.

Pushdown Automata:

Pushdown automata of the informal description is as shown:



In the above push down automata, the state q_0 is the initial state and q_3 is the final state. In each step it checks whether the middle symbol is 0. If the stack is empty and middle symbol is 0 then the string is accepted by the push down automata otherwise pushdown automata will reject the input.

Comments (1)

Step 8 of 10

Given:

$L = \{w \mid w = w^R\}$, and w is the palindrome.

Informal Description:

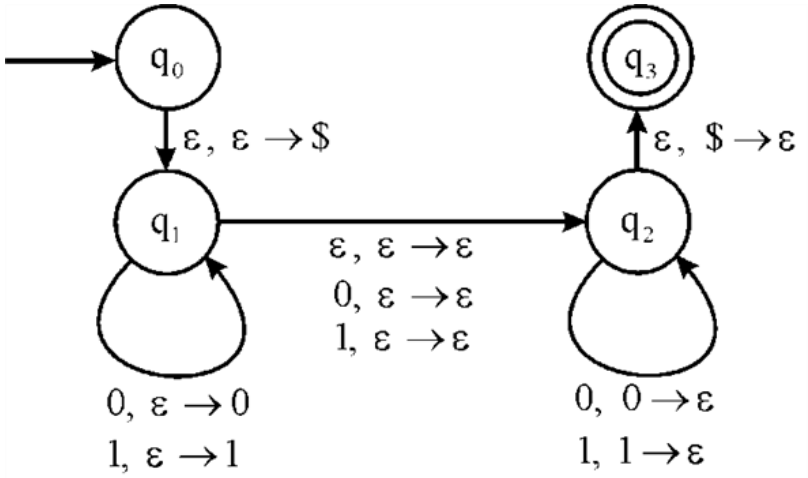
• In the above grammar first half of the palindrome is pushed into the stack and after that underministically middle is being guessed.After that poping of the element is done from the second half of the palindrone.

• The first half of the input and the second half of the input should match with each other.

• If it matches then stack is made empty and after that push down automata accepts that particular string. If it does not matches then pushdown automata then reject that particular string.

Pushdown Automata:

Pushdown automata of the informal description is as shown:



• In the above push down automata q_0 is the starting stage and q_3 is the ending stage. In the stage q_1 first half is pushed into the stack excluding the middle symbol if the palindrome is odd.

• If the string is not odd then user guesses where the middle point of the input is non-deterministically.

• After determining the middle part of the input user goes to state q_2 without the consumption of any input if the string is found to be even.

• In the state q_2 popping of the symbol is done from the stack and it is being checked whether it matches with the input state in the input in the beginning.

•

Comments (2)

Step 9 of 10

If the first half and second half of the input is matched, then the palindrome is accepted by the push down automata otherwise palindrome is rejected by the push down automata.

Comment

Step 10 of 10

Given:

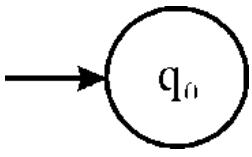
L is an empty set which implies $S \to S$

Informal Description:

In the above grammar as the language accept only an empty set so it implies it does not terminate any derivation. If non of the derivation is terminated then no any string is accepted by the CFG which also includes an empty string.

Pushdown Automata:

Pushdown automata of the informal description is as shown:



Comments (1)

Problem

Give context-free grammars generating the following languages.

a. The set of strings over the alphabet {a,b} with more a's than b's

b. The complement of the language $\{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$

c. $\{w\#x \mid w^{\mathcal{R}} \text{ is a substring of } x \text{ for } w, x \in \{0,1\}^*\}$

d. $\{x_1\#x_2\#\cdots\#x_k \mid k \geq 1, \text{each } x_i \in \{\mathtt{a},\mathtt{b}\}^*, \text{and for some } i \text{ and } j,\ x_i = x_j^{\mathcal{R}}\}$

Step-by-step solution

Step 1 of 2

Given language is

"The set of strings over the alphabet $\{a,b\}$ with more a's than b's"

The context – free grammar generating the given language is

$S \to Aa \mid BS \mid SBA$

$A \to Aa \mid \in$

$B \to \in \mid BB \mid bBa \mid aBb$

In the above grammar S will generate all strings with as many a's as b's. R Forces an extra a which gives the required strings of the language.

Given language is

"The compliment of the language $\{a^n b^n : n \geq 0\}$"

Let L be the language that is a compliment of given language. L can be obtained as $L = \{a^n b^m : n \neq m\} \cup \{(a \cup b)^* ba(a \cup b)^*\}$

Let us conside

$L_1 = \{a^n b^m : n \neq m\}$

$L_2 = \{(a \cup b)^* ba(a \cup b)^*\}$

The context – free grammar generating the language L_1 is

$S_1 \to aS_1b \mid T \mid U$

$T \to aT \mid a$

$U \to Ub \mid a$

The context – free grammar generating the language L_2 is

$S_2 \to RbaR$

$R \to RR \mid a \mid b \mid \varepsilon$

Therefore, the required context – free grammar generating the language L is given by

$L = L_1 \cup L_2$

$S \to S_1 \mid S_2$

$S_1 \to aS_1b \mid T \mid U$

$S_2 \to RbaR$

$T \to aT \mid a$

$U \to Ub \mid a$

$R \to RR \mid a \mid b \mid \varepsilon$

Given language is

$\{w\#x : w^R \text{ is a substring of } x \text{ for } w, x \in \{0,1\}^*\}$

The context – free grammar generating the given language is

$R \to SX$
$S \to 0S0 \mid 1S1 \mid \#X$
$X \to XX \mid 1 \mid 0 \mid \varepsilon$

The nonterminal S ends only with $\#X$, S must generate a string whose beginning and end are mirror images. Since X generates $(0 \cup 1)^*$, the symbol S generates all strings of the form $w\#(0 \cup 1)^* w^R$. The above grammar generates all the substrings of x for $w, x \in \{0,1\}^*$.

Comments (4)

Step 2 of 2

Given language is

$\{x_1 \# x_2 \# \cdots \# x_k \mid k \geq 1, \text{ each } x_i \in \{a,b\}^*, \text{ and for some } i \text{ and } j, x_i = x_j^R\}$

The context – free grammar generating the given language is

$R \to S \mid J\#S\#J \mid J\#S \mid S\#J$
$S \to aSa \mid bSb \mid \# \mid \#J\#$
$J \to aJ \mid bJ \mid \#J \mid \varepsilon$

The strings in the language contain matching pair of strings with at least one # between them. Before, after and between the matching pairs there can be any number of strings of a's and b's separated by #. Because the strings can be of any length, the strech of the strings of a's,b's and #'s can be of any length. The symbol S generates a matching pair, with strings of a's,b's and #'s optionally inserted in the middle. The symbol J generates strings of a's,b's and #'s. The string generated by J may start or end with a or b, so rules for M and S must ensure that the symbol J is always separated properly from the two matching strings.

Comments (1)

Problem

Give informal English descriptions of PDAs for the languages in Exercise 2.6.

Step-by-step solution

Step 1 of 4

a.

Consider the following language:

"The set of strings over the alphabet $\{a,b\}$ with more a's than b's"

This is the language where the number of a's is greater then the number of b's. Construct the PDA (Push Down Automata), that accepts the above language. It uses its stack to count the number of a's minus the number of b's. It enters an accepting state whenever this count is positive or top of the stack is a.

In more detail, it operates as follows:

• The PDA scans across the input.

• If it sees a a and its top of the stack symbol is either a or empty than, push a into the stack.

• If it scans a a and its top stack symbol is a b, it pops the stack.

• If it sees a b and its top of the stack symbol is either b or empty than, push b into the stack.

• If it sees a b and its top stack symbol is a, it pops the stack.

After the PDA finishes the input, if a is on top of the stack, it accepts. Otherwise, it rejects.

Therefore, given language is recognized by the above PDA.

Comment

Step 2 of 4

b.

Consider the following language:

"The compliment of the language $\{a^n b^n : n \geq 0\}$"

First, we will construct the compliment of the given language. That is number of a's is not equal to the number of b's. The compliment of this language can obtain by breaking the given language into the union of several simpler languages.

The PDA uses its stack to count the number of a's and the number of b's. It enters an accepting state whenever this count is not equal.

In more detail, it operates as follows:

• The PDA scans across the input.

• If it sees a a and its top of the stack symbol is either a or empty than, push a into the stack.

• If it sees a b and its top stack symbol is a, it pops the stack.

• If it sees a b and its top of the stack symbol is either b or empty than, push b into the stack.

After the PDA finishes the input, if a or b is on top of the stack, it accepts. Otherwise, it rejects.

Therefore, if the end of the input is reached and stack is not empty than, it accepts. Otherwise it rejects because number of a's and the number of b's in the input are equal.

Comments (2)

Step 3 of 4

c.

Consider the following language:

$\left\{w\#x : w^R \text{ is a substring of } x \text{ for } w,x \in \{0,1\}^*\right\}$

The language is obtained by constructing the strings w^R which is a substring x. The PDA scans across the input string and finds the input is a substring of x.

In more detail, it operates as follows:

• The PDA scans across the input string and pushes every symbol it reads until it reads until it reads a $\#$.

• If $\#$ is never encountered, it rejects.

• Then the PDA skips over part of the input, nondeterministically deciding when to stop skipping.

• At that point, it compares the next input finishes while the stack is nonempty, this branch of the computations rejects.

After the PDA finishes the input, if the stack becomes empty, the machine reads the rest of the input and accepts.

Therefore, given language is recognized by the above PDA.

Comment

Step 4 of 4

d.

Consider the following language:

$$\left\{x_1 \# x_2 \# \cdots \# x_k \mid k \geq 1, \text{ each } x_i \in \{a,b\}^*, \text{ and for some } i \text{ and } j, x_i = x_j^R\right\}$$

The language can obtain by constructing matching pair of strings with at least one $\#$ between them. Before, after and between the matching pairs there can be any number of strings of a's and b's separated by $\#$. The PDA scans across the input string and finds the input is with at least one $\#$ is between them.

In more detail, it operates as follows:

• The PDA scans across the input string and pushes every symbol it reads until it reads until it reads a $\#$.

• If # is never encountered, it rejects.

• If an input $\#$ is read, the nondeterministically skips to another # in the input and continue reading input symbols, now comparing them with symbols popped off the stack. If all agree and stack becomes empty at the same time as either end of the input of # symbol is reached, accept, otherwise reject.

After the PDA finishes the input, if the stack becomes empty, the machine reads the rest of the input and accepts.

Therefore, given language is recognized by the above PDA.

Comment

Problem

Show that the string the girl touches the boy with the flower has two different leftmost derivations in grammar G_2 on page 103. Describe in English the two different meanings of this sentence.

Step-by-step solution

Step 1 of 2

Leftmost derivative is use for replacing the leftmost variable. Each and every variable is replaced on the consecutive step.

Given: the girl touches the boy with flower

Proof: The above given string has two leftmost derivative

Grammars describing the fragment of English language are as follows:

$$\begin{aligned}\langle\text{SENTENCE}\rangle &\to \langle\text{NOUN-PHRASE}\rangle\langle\text{VERB-PHRASE}\rangle\\ &\to \langle\text{CMPLX-NOUN}\rangle\langle\text{VERB-PHRASE}\rangle\\ &\to \langle\text{ARTICLE}\rangle\langle\text{NOUN}\rangle\langle\text{VERB-PHRASE}\rangle\\ &\to \text{the}\langle\text{NOUN}\rangle\langle\text{VERB-PHRASE}\rangle\end{aligned}$$

$\to$ the girl $\langle$VERB-PHRASE$\rangle$
$\to$ the girl $\langle$CMPLX-VERB$\rangle$
$\to$ the girl $\langle$VERB$\rangle\langle$NOUN-PHRASE$\rangle$
$\to$ the girl touches $\langle$NOUN-PHRASE$\rangle$
$\to$ the girl touches $\langle$CMPLX-NOUN$\rangle\langle$PREP-PHRASE$\rangle$
$\to$ the girl touches $\langle$ARTICLE$\rangle\langle$NOUN$\rangle\langle$PREP-PHRASE$\rangle$
$\to$ the girl touches the $\langle$NOUN$\rangle\langle$PREP-PHRASE$\rangle$
$\to$ the girl touches the boy $\langle$PREP-PHRASE$\rangle$
$\to$ the girl touches the boy $\langle$PREP$\rangle\langle$CMPLX-NOUN$\rangle$
$\to$ the girl touches the boy with $\langle$CMPLX-NOUN$\rangle$
$\to$ the girl touches the boy with $\langle$ARTICLE$\rangle\langle$NOUN$\rangle$
$\to$ the girl touches the boy with the $\langle$NOUN$\rangle$
$\to$ the girl touches the boy with the flower

The above phrase is the first derivative of the leftmost derivation.

Comment

Step 2 of 2

Grammars describing the fragment of English language are as follows:

$$\begin{aligned}\langle\text{SENTENCE}\rangle &\to \langle\text{NOUN-PHRASE}\rangle\langle\text{VERB-PHRASE}\rangle\\ &\to \langle\text{CMPLX-NOUN}\rangle\langle\text{VERB-PHRASE}\rangle\\ &\to \langle\text{ARTICLE}\rangle\langle\text{NOUN}\rangle\langle\text{VERB-PHRASE}\rangle\\ &\to \text{the}\langle\text{NOUN}\rangle\langle\text{VERB-PHRASE}\rangle\end{aligned}$$

$\to$ the girl $\langle$VERB-PHRASE$\rangle$
$\to$ the girl $\langle$CMPLX-VERB$\rangle\langle$PREP-PHRASE$\rangle$
$\to$ the girl $\langle$VERB$\rangle\langle$NOUN-PHRASE$\rangle\langle$PREP-PHRASE$\rangle$
$\to$ the girl touches $\langle$NOUN-PHRASE$\rangle\langle$PREP-PHRASE$\rangle$
$\to$ the girl touches $\langle$CMPLX-NOUN$\rangle\langle$PREP-PHRASE$\rangle$
$\to$ the girl touches $\langle$ARTICLE$\rangle\langle$NOUN$\rangle\langle$PREP-PHRASE$\rangle$
$\to$ the girl touches the $\langle$NOUN$\rangle\langle$PREP-PHRASE$\rangle$
$\to$ the girl touches the boy $\langle$PREP-PHRASE$\rangle$

→ the girl touches the boy ⟨PREP⟩⟨CMPLX-NOUN⟩

→ the girl touches the boy with ⟨CMPLX-NOUN⟩

→ the girl touches the boy with ⟨ARTICLE⟩⟨NOUN⟩

→ the girl touches the boy with the ⟨NOUN⟩

→ the girl touches the boy with the flower

The above phrase is the second derivative of the leftmost derivation.

Comments (1)

Problem

Give a context-free grammar that generates the language

$$A = \{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid i = j \text{ or } j = k \text{ where } i, j, k \geq 0\}.$$

Is your grammar ambiguous? Why or why not?

Step-by-step solution

Step 1 of 1

The language given in the problem is as follows:

$A = \left\{a^i b^j c^k \mid \text{i=j } or \text{ j} = k \text{ for i} \geq 0, \text{j} \geq 0,\ k \geq 0\right\}$

The language *A* can be split into two languages which are defined as follows:

$A_1 = \left\{a^i b^j c^k \mid i, j, k \geq 0, i = j\right\}$

and

$A_2 = \left\{a^i b^j c^k \mid i, j, k \geq 0, j = k\right\}$

Using the language *A1* and *A2* the user can construct a CFG for A_1 and A_2.

The grammar for language *A* is the union of grammar of two languages which is defined as follows:

$S \to S_1 \mid S_2$

In the language A_1 the values of i and j are equal so there must be equal number of a's and b's in the language A_1.

CGF for the language A_1 is as follows:

$S_1 \to S_1 c \mid E \mid \in$
$E \to aEb \mid \in$

Similarly, in the language A_2 the values of j and k are equal so there must be equal number of b's and c's in the language A_2.

CGF for the language A_2 is as follows:

$S_2 \to aS_2 \mid F \mid \in$
$F \to bFc \mid \in$

Since for generating a string $w = a^n b^n c^n$ using the language *A*, either S_1 or S_2 can be used.

Therefore, the context free grammar for the language *A* is ambiguous.

Comment

Problem

Give an informal description of a pushdown automaton that recognizes the language A in Exercise 2.9.

Step-by-step solution

Step 1 of 1

2287-2-10E AID: 824

RID: 944

Given language is

$A = \{a^i b^j c^k \mid i = j \text{ or } j = k \text{ where } i, j, k \geq 0\}$

The language is union of two languages $\{a^i b^i c^k \mid i, k \geq 0\}$ and $\{a^i b^k c^k \mid i, k \geq 0\}$.

Let $A_1 = \{a^i b^i c^k \mid i, k \geq 0\}$ and $A_2 = \{a^i b^k c^k \mid i, k \geq 0\}$.

The informal description of the PDA that recognizes the language A_1.

In more detail, it operates as follows:

• Read and push $a's$.

• Read $b's$, while popping $a's$.

• If $b's$ finish when stack is empty, skip $c's$ on input and accept.

The informal description of the PDA that recognizes the language A_2.

In more detail, it operates as follows:

• Skip $a's$ on input.

• Read and push $b's$.

• Read $c's$, while popping $b's$.

• If $c's$ finish when stack is empty, accept.

The informal description of the PDA that recognizes the language A is the combinition of both the languages A_1 and A_2.

In more detail, it operates as follows:

1. Nondeterministically branck to either step 2 or step 6.

2. Read and push $a's$.

3. Read $b's$, while popping $a's$.

4. If $b's$ finish when stack is empty, skip $c's$ on input and accept.

5. Skip $a's$ on input.

6. Read and push $b's$.

7. Read $c's$, while popping $b's$.

8. If $c's$ finish when stack is empty, accept.

Comment

Problem

Convert the CFG G4 given in Exercise 2.1 to an equivalent PDA, using the procedure given in Theorem 2.20.

THEOREM **2.20**

A language is context free if and only if some pushdown automaton recognizes it.

Step-by-step solution

Step 1 of 2

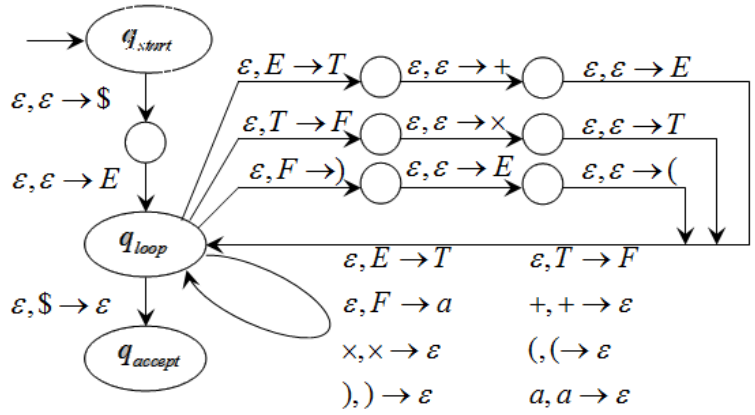
Given CFG (Context-free grammar) G_4 is

$E \to E+T \mid T$

$T \to T \times F \mid F$

$F \to (E) \mid a$

Equivalent PDA for the CFG G_4 is as follows:



Comments (5)

Step 2 of 2

Explanation:

1. A shorthand notation is used for pushing multiple symbols onto the stack.

2. Initially, at the start variable on the stack a marker symbol '$' is inserted. The start state is q_{start}. The transition function is $\delta(q_{start}, \in, \in) = \{(q_{loop}, S\$)\}$

3. If the stack top is a non-terminal variable E. Select one of the rules of E and substitutes its value on the right hand side of the rule. Repeat this process until the end of the string.

The transition function is $\delta(q_{loop}, \in, E) = \{(q_{loop}, w) \mid E \to w \text{ is a rule in } CFG\}$

Example:

• Consider the rule $E \to E+T$.

• Another rule for *E* is $E \to T$. Substitute the value of *E* in the above rule.

• Then, the equation becomes $E \to T+T$.

4. If the stack top is a terminal variable such as (,),a,+ and x the next symbol is read from the input rule. Repeat step-3 if again a non-terminal variable is encountered.

The transition function is $\delta(q_{loop}, a, a) = \{(q_{loop}, \in)\}$.

5. If the stack top is a '$' symbol, the accept state is entered because, the input is read completely.

The transition function is $\delta(q_{loop}, \in, \$) = \{(q_{accept}, \in)\}$.

Comment

Problem

Convert the CFG G given in Exercise 2.3 to an equivalent PDA, using the procedure given in Theorem 2.20.

THEOREM **2.20**

A language is context free if and only if some pushdown automaton recognizes it.

Step-by-step solution

Step 1 of 2

CFG stands for context free grammar. It is a set of recursive rules which are used to create a string pattern.

• A CFG contains set of terminals and no-terminals.

• Generally, Non terminals or variables are represented with capital alphabets whereas terminals are represented with small alphabets.

• The languages which use context free grammar are called as context free language.

• Machines which recognize the context free language are called as **push down automata**. It is used to provide additional power to CFG.

Comment

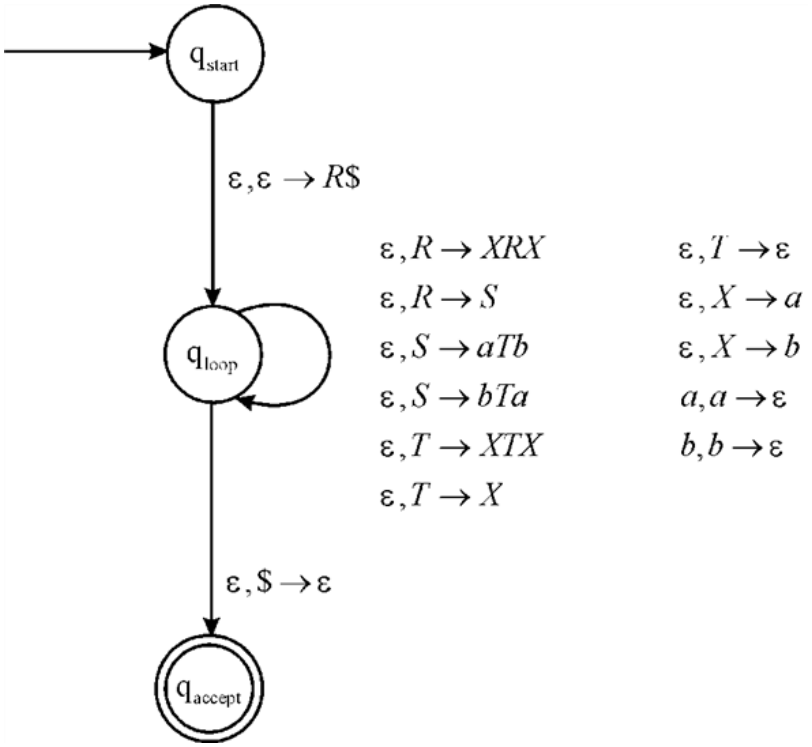
Step 2 of 2

Conversion of CFG to PDA:

Consider the following context free grammar:

$R \to XRX \mid S$
$S \to aTb \mid bTa$
$T \to XTX \mid X \mid \varepsilon$
$X \to a \mid b$

Consider the following diagram to represent the equivalent PDA of the above CFG:



• The above transition rule allows someone to reduce the grammar that is to replace non-terminals or variables to the right-hand side terminals.

• Transition for the terminal symbols such as (a, b) permits someone for matching the input symbol to the terminal symbol.

• A path of PDA of string *w* can only be accepted if an input string *w* can only be generated by grammar *G*.

Problem

Let G = (V,∑,R, S) be the following grammar. V = {S, T, U}; ∑ = {0, #}; and R is the set of rules:

$$S \rightarrow TT \mid U$$
$$T \rightarrow 0T \mid T0 \mid \#$$
$$U \rightarrow 0U00 \mid \#$$

a. Describe L(G) in English.

b. Prove that L(G) is not regular.

Step-by-step solution

Step 1 of 4

Let $G = (V, \Sigma, R, S)$ be the grammar. $V = \{S, T, U\}$; $\Sigma = \{0, \#\}$. R is the set of rules.

$S \rightarrow TT \mid U$
$T \rightarrow 0T \mid T0 \mid \#$
$U \rightarrow 0U00 \mid \#$

Comment

Step 2 of 4

Description of $L(G)$ in English is given as

$L(G)$ is the set of all strings containing $0's$ and $\#'s$ of the form(s)

(a) Strings containing exactly two $\#$'s and any number of $0's$.

(b) Strings containing exactly one $\#$ and the number of $0's$ on the left of $\#$ are half the number of $0's$ to the right.

Comments (1)

Step 3 of 4

To prove the language $L(G)$ is not regular, consider by contradiction $L(G)$ is regular.

Let $A = L(G) \cap 0^* \#0^*$.

From our consideration $L(G)$ is regular. So A is regular.

Since A is regular, using pumping lemma let p be the pumping length of the regular language.

Consider the sting $w = 0^p \#0^{2p}$.

Clearly length of the string $w \in A$ is greater the p. That is $|w| > p$.

So, by pumping lemma we have $w = xyz$ such that

$|xy| \leq p$, $y \neq \varepsilon$ and $xy^i z \in L(G)$ for all $i \geq 0$.

Comment

Step 4 of 4

Since A is a regular language, consider the possible ways of cutting the string w.

• If x contains the character $\#$, then y will be on the right side of $\#$. Pumping y down, increases the number of $0's$ on the left such that number of $0's$ on the left of $\#$ are not equal to half the number of $0's$ to the right.

• If y contains the character $\#$, then pumping y down, increases the number of $\#$'s in the string.

• If z contains the character $\#$, then y will be on the left side of $\#$. Pumping y down, decreases the number of $0's$ on the right such that number of $0's$ on the left of $\#$ are not equal to half the number of $0's$ to the right.

The resulting string of all the above cases does not belong to A. So, we can say that A does not satisfy pumping lemma. Hence A is not a regular language.

This is a contradiction to our statement that $L(G)$ regular language.

Therefore, $L(G)$ is not a regular language.

Comment

Problem

Convert the following CFG into an equivalent CFG in Chomsky normal form, using the procedure given in Theorem 2.9.

$$A \to BAB \mid B \mid \varepsilon$$
$$B \to 00 \mid \varepsilon$$

THEOREM **2.9**

Any context-free language is generated by a context-free grammar in Chomsky normal form.

Step-by-step solution

Step 1 of 7

Given CFG (Context-free Grammar) is

$A \to BAB \mid B \mid \varepsilon$
$B \to 00 \mid \varepsilon$

Now, construct an equivalent CFG (Context-free Grammar) in Chomsky normal form.

Chomsky normal form:

A context i– free grammar s in **Chomsky normal form** if every rule is of the form

$A \to BC$
$A \to a$

Here, a is terminal,

A, B and C are variables,

In addition, it permits the rule $S \to \varepsilon$, here S is the start variable.

convert the given CFG into an equivalent CFG in Chomsky normal form.

Comment

Step 2 of 7

Let's add a new start variable S_0 and the rule $S_0 \to A$.

Thus the obtained grammar is

$S_0 \to A$
$A \to BAB \mid B \mid \varepsilon$
$B \to 00 \mid \varepsilon$

In the addition of new start variable guarantees that the start variable doesn't occur on the right-hand side of a rule.

Comment

Step 3 of 7

Removing all rules that containing ε.

Removing $A \to \varepsilon$ and $B \to \varepsilon$ gives

$S_0 \to A \mid \varepsilon$
$A \to BAB \mid BA \mid AB \mid A \mid B \mid BB$
$B \to 00$

The rule $S_0 \to \varepsilon$ is accepted since S_0 is the start variable and that is allowed in Chomsky normal form.

Comments (1)

Step 4 of 7

Now remove the unit rules.

Removing $A \to A$ gives

$S_0 \to A \mid \varepsilon$
$A \to BAB \mid BA \mid AB \mid B \mid BB$
$B \to 00$

Removing $S \to B$ gives

$S_0 \to A \mid \varepsilon$
$A \to BAB \mid BA \mid AB \mid 00 \mid BB$
$B \to 00$

Removing $S_0 \to S$ gives

$S_0 \to BAB \mid BA \mid AB \mid 00 \mid BB \mid \varepsilon$
$A \to BAB \mid BA \mid AB \mid 00 \mid BB$
$B \to 00$

Comments (2)

Step 5 of 7

Now replace ill placed terminals 0 by variable U with new

$S_0 \to BAB \mid BA \mid AB \mid UU \mid BB \mid \varepsilon$
$A \to BAB \mid BA \mid AB \mid UU \mid BB$
$B \to UU$
$U \to 0$

Comment

Step 6 of 7

Shorten the right-hand side of rules with only 2 variables each.

To shorten the rules, replace $S_0 \to BAB$ with two rules $S_0 \to BA_1$ and $A_1 \to AB$.

The rule $A \to BAB$ is replaced by the two rules $A \to BA_2$ and $A_2 \to AB$.

After replacing these rules, the final Context-free grammar in Chomsky normal form is $G = (V, \Sigma, R, S_0)$,

Here the set of variables is $V = \{S_0, S, B, U, A_1, A_2\}$,

the start variable is S_0.

The set of terminals is $\Sigma = \{0\}$, and the rules R are given by

Comment

Step 7 of 7

$S_0 \to BA_1 \mid BA \mid SB \mid UU \mid BB \mid \varepsilon$
$A \to BA_1 \mid BA \mid SB \mid UU \mid BB$
$B \to UU$
$U \to 0$
$A_1 \to AB$

This is the final CFG in Chomsky normal form equivalent to the given CFG.

Comments (4)

Problem

Give a counterexample to show that the following construction fails to prove that the class of context-free languages is closed under star. Let A be a CFL that is generated by the CFG G = (V,∑,R, S). Add the new rule S → SS and call the resulting grammar G′. This grammar is supposed to generate A^*.

Step-by-step solution

Step 1 of 2

Example to show the failure of closure of Context free language under Star

Context free language is the language which is generated by CFG or the context free grammar. It is possible to get different types of context free languages from different types of context free grammar.

Given: Here, it is given that *A* is a context free language and the context free grammar which is responsible for its generation is given as $G=(V,\Sigma,R,S)$.

Here it is required to show the failure of the construction in proving the closure of the context free language is closed under star.

Comment

Step 2 of 2

Proof:

This can be done by adding a new rule as $S \to SS$ and the grammar which will be obtained as resultant will be called as G'. This grammar is expected to generate A^*.

Now, the counter example can be considered by supposing the CFL as A and the grammar corresponding to it along with its production rules as $G=\{\{S\},\{(,)\},\{S \Rightarrow (S), S \Rightarrow \in\},S\}$.

Now, as required and given add the new production rule as $S \to SS$. As the new rule is being added a new grammar represented as G'will be obtained.

This grammar will generate the language containing the string symbols as $((\ ),(\ ))$. And, thus, this grammar is not capable of generating the star closure of *A* represented by A^* .

Conclusion: Thus, it can be concluded that the new grammar G' fails to prove that the class of context free languages is closed under star.

This condition is true only if the A^* contains empty string else false. So, A^* must contains the empty string.

Comment

Problem

Show that the class of context-free languages is closed under the regular operations, union, concatenation, and star.

Step-by-step solution

Step 1 of 3

In order to show that context free language is closed under union operation considers two starts variable S_1 and S_2 for the two different languages L_1 and L_2.

Grammar for union operation is as shown:

$S \to S_1 \mid S_2$

If both the language belongs to the context free language then union of both the language should belong to context free language.

By the above definition if user generates S_1 and S_2 string or both then in that case union of both the language is generated.

Hence, $L_1 \cup L_2 \in CFL$.

This implies context free language is closed under union operation.

Comments (1)

Step 2 of 3

In order to show that context free language is closed under concatenation operation considers two starts variable S_1 and S_2 for the two different languages L_1 and L_2.

Grammar for union operation is as shown:

$S \to S_1 S_2$

If both the language belongs to the context free language then concatenation of both the language should belong to context free language.

$\forall L_1, L_2 \in CFL$

$\{w_1 w_2 : w_1 \in L_1 \wedge w_2 \in L_2\} \in CFL$

By the above definition if user generates S_1 string for language L_1 followed by S_2 string of language L_2. Then it concatenation of both the language is generated.

Hence, $\{w_1 w_2 : w_1 \in L_1 \wedge w_2 \in L_2\} \in CFL$

This implies context free language is closed under concatenation operation.

Comments (1)

Step 3 of 3

In order to show that context free language is closed under star operation.

Consider one start variable S_1 for the languages L_1

Grammar for union operation is as shown:

$S \to S_1 S \mid \in$

If the language belongs to the context free language then star of the language should belong to context free language.

$\forall L_1 \in CFL$

By the above definition, if user generates zero or many string which is the definition of the star.

This implies that context free language is closed under star operation.

Comments (1)

Problem

Use the results of Exercise 2.16 to give another proof that every regular language is context free, by showing how to convert a regular expression directly to an equivalent context-free grammar.

Step-by-step solution

Step 1 of 3

Consider the following data:

From Exercise 2.16, it is clear that Context free language is closed under union operation, concatenation operation and star operation.

Comment

Step 2 of 3

The definition of the regular expressions it can be inferred and known that regular expression consist of series of operations such as union, concatenation and star.

Statement: Every regular language is context free language

Proof by using induction:

Assume that *E* is a regular expression and there exists a CFG *G* such that $L(E) = L(G)$. The proof will be carried out by different number of operators in an expression *E*.

Step-1:

If $Operation(E) = 0$ then E is either $\phi, \in or\ a$ in $\sum$.

Step-2:

• If $E = \phi$: $G = (\{S\}, \{\ \}, P, S)$. Here $P = \{\ \}$.

• If $E = \in$: $G = (\{S\}, \{\ \}, P, S)$. Here $P = \{S \to \in\}$.

• If $E = a$: $G = (\{S\}, \{a\}, P, S)$. Here $P = \{S \to a\}$.

For every regular expression consisting of $\phi, \in or\ a$ the CFG can be written. So, every regular expression is context free.

Thus, every regular language is context free language.

Comment

Step 3 of 3

The context free languages are superset of regular languages. The rules for converting the regular languages consisting of union, concatenation and Kleene closure are as follows:

• If the regular expression E consists of E_1 and E_2 such that the concatenation $E = E_1E_2$, then the production can be easily expressed as $S \to E_1E_2$.

• If the regular expression E consists of E_1 and E_2 such that the union $E = E_1 \mid E_2$, then the production can be easily expressed as $S \to E_1 \mid E_2$.

• If the regular expression E consists of E_1 such that the Kleene closure $E = E_1^*$, then the production can be easily expressed as $E_1^* \to EE_1^* \mid \in$.

Comment

Problem

a. Let C be a context-free language and R be a regular language. Prove that the language C ∩ R is context free.

b. Let A = {wlw ? {a, b, c}* and w contains equal numbers of a's, b's, and c's}. Use part (a) to show that A is not a CFL.

Step-by-step solution

Step 1 of 2

2287-2-18P AID: 824

RID: 944

a) Let C be a context free language and R be a regular language.

Now we have to prove that the language $C \cap R$ is a context free.

In order to prove, let us consider

P be the PDA (Push Down Automata) recognizes C, and

D be the DFA (Deterministic Finite Automata) recognizes R.

Now we construct a PDA that recognizes $C \cap R$ with the set of states $Q \times Q'$

where

Q be the set of states of P and

Q' is the set of states of D

Here P' will do what P does and keep track of the states of D.

The PDA that recognizes $C \cap R$ accepts the string w if and only if it stops a state $q \in F_P \times F_D$

where

F_P is the states of accepts of P and

F_D is the states of accepts of D.

So $C \cap R$ is recognized by P'.

Therefore, $C \cap R$ is context free.

Comment

Step 2 of 2

b) Given the language is

$$A = \left\{ w \mid w \in \{a,b,c\}^* \text{ and contains equal number of } a\text{'s, } b\text{'s and } c\text{'s} \right\}$$

Now we have to prove A is not a CFL (Context Free Language) .

Let R be the regular language $a^*b^*c^*$.

If A were a CFL (Context Free Language) then $A \cap R$ would be a CFL (using the result proved above in part (a) of this problem).

Hence, inorder to prove that A is not a CFL it is enough to prove that $A \cap R$ is not a CFL.

We have $A \cap R = \left\{ a^n b^n c^n \mid n \geq 0 \right\}$.

We will prove $A \cap R$ is not a CFL by taking a contradiction.

Assume that $A \cap R$ is a CFL.

Using the pumping lemma, which states thet every context-free language has a special calue called *pumping length* such that all longer strings in the language can be "pumped", let p be the pumping length for $A \cap R$.

Consider a string $s = a^p b^p c^p$.

Clearly s is a member of $A \cap R$ and of length at least p.

Now we prove that one condition of pumping lemma violated by proving s cannot be pumped.

If we divide s into $wxyz$, condition 2 stipulates that either v or y is non-empty.

Now consider one of the two cases, depending on wheather substring v and y contains more than one type of alphabet symbol.

1. If both v and y contain only one type of symbol, v doesn't contain both a's and b's or both b's and c's, and the same holds for y. Here the string uv^2xy^2z cannot contain equal number of a's, b's and c's. Therefore it cannot be a member of $A \cap R$ which violates the first condition of the pumping lemma and thus is a contradiction to our hypothsis.

2. If either v or y contain more than one type of symbol uv^2xy^2z may contain equal number of the three alphabet symbols but not in the correct order. Hence it cannot be a member of $A \cap R$ and thus is a contradiction to our hypothsis.

One of the above two case must occur. However, both the cases raised contradiction. This is because of our assumption $A \cap R$ is a CFL.

Hence our assumption is false and $A \cap R$ is not a CFL

Therefore, A is not a CFL.

Comments (3)

Problem

Let CFG G be the following grammar.

$$S \to \mathtt{a}S\mathtt{b} \mid \mathtt{b}Y \mid Y\mathtt{a}$$
$$Y \to \mathtt{b}Y \mid \mathtt{a}Y \mid \varepsilon$$

Give a simple description of L(G) in English. Use that description to give a CFG for $\overline{L(G)},$ the complement of L(G).

Step-by-step solution

Step 1 of 2

Consider the context free grammar (CFG) G is as follows:

$S \to aSb \mid bY \mid Ya$
$Y \to bY \mid aY \mid \varepsilon$

Language *L* (*G*) for the G is as follows:

Consider the productions in the grammar

$S \to aSb$
$S \to bY$
$S \to Ya$
$Y \to bY$
$Y \to aY$
$Y \to \varepsilon$

Case 1:

Consider production $S \to Ya$ to derive the language.

Substitute Y with production $Y \to \varepsilon$ then

$S \to \in a$

$S \to a$

Case 2:

Consider production $S \to bY$ to derive the language.

Substitute Y with production $Y \to \varepsilon$ then

$S \to \in b$

$S \to b$

Case 3:

Consider production $S \to aSb$ to derive the language

Substitute S with production $S \to bY$ then

$S \to abYb$

Substitute Y with production $Y \to bY$ then

$S \to abbYb$

Substitute Y with production $Y \to \varepsilon$ then

$S \to abb \in b$

$S \to abbb$

Case 4:

Consider production $S \to bY$ to derive the language.

Substitute Y with production $Y \to bY$ then

$S \to bbY$

Substitute Y with production $Y \to \varepsilon$ then

$S \to bb \in$

$S \to bb$

Therefore form the Case 1, Case 2, Case 3 and Case 4 the language obtained is as follows:

L(*G*) = {*a, b, abbb, bb…*}

Using the grammar *G*, many more strings can be generated.

Comment

Step 2 of 2

Description of the *L* (*G*) is as follows:

The grammar G generates a language *L*(*G*) consists of the strings which are described as follows:

• Strings with consecutive number of ***a***'s with a length ranging from 1 to infinity.

• Strings with consecutive number of ***b*'s** with a length ranging from 1 to infinity.

• String with start symbol ***a*** followed by number of ***b*'s**.

• Strings with start symbol ***b*** followed by number of ***a*'s.**

• Strings with ***a*** as start symbol and ***b*** as end symbol.

• Strings with ***b*** as start symbol and ***a*** as end symbol.

• Strings that contains the same start and end symbols. For example, $aba,\ bab$ etc.

From the above description as *L*(*G*) is generating all the possible combination of *a*'s and *b*'s except $a^i b^i \ where \ i \geq 0$. The *L*(*G*) does not produce strings like $\in, ab,\ aabb,\ aaabbb \ldots$

The complements of $L(G)$ i.e. $\overline{L(G)}$ = $\{\in, ab,\ aabb,\ aaabbb \ldots\}$

The grammar for $\overline{L(G)}$ is $a^i b^i \ where \ i \geq 0$.

Therefore, the CFG G' for $\overline{L(G)}$ is as follows:

$S \to aSb \mid \in$

Comment

Problem

Let A/B = {wl wx ? A for some x ? B}. Show that if A is context free and B is regular, then A/B is context free.

Step-by-step solution

Step 1 of 1

From the problem statement assume a language A is context free and a language B is regular.

From the theorem 2.20: A language is a context free language if and only if some push down automaton recognizes it.

From the corollary 1.40: A language is regular if and only if some nondeterministic finite automaton (NFA) recognizes it.

To prove that A/B is context free then it must be recognized by some push down automaton.

The proof is as follows:

Firstly construct a push down automata P_A for the context free language A

Consider the following pushdown automata P_A:

$P_A=(Q_A,\Sigma,\Gamma,\delta_A,q_A,F_A)$

and a NFA M for the regular language B.

Consider the following NFA *M*:

$M=(Q_B,\Sigma,\delta_B,q_B,F_B)$

Assume $P_{A/B}$ is the push down automaton that recognizes the Context free language *A/B.*

$P_{A/B}=(Q^{A/B},\Sigma,\Gamma^{A/B},\delta_{A/B},q_{A/B},F_{A/B})$

• Now $P_{A/B}$ will read the prefix **w** of the input string.

• $P_{A/B}$ will guess that it has reached to the end of input string **w** at a non-deterministically chosen point;

• $P_{A/B}$ will behave like P_A and *M* running concurrently, except that it will guess the input string **x,** rather than actually reading it as input.

• If it is feasible in this way to concurrently to reach an accepting state of both P_A and *M* then $P_{A/B}$ accepts.

• Note that there is no reason why the stack would have to be empty at the point where $P_{A/B}$ begins the guessing phase.

• So it is essential for $P_{A/B}$ to carry on *modeling* P_A in order to properly account for the stack contents.

$P_{A/B}$ is defined as follows:

• $Q^{A/B}=Q_A\ (Q_A \times Q_B)$

• $\Gamma^{A/B} = \Gamma$

• $q_{A/B}=q_0P_A$ where $q_0=q_A=q_B$

• $F_{A/B}=F_A\times F_B$

• $\delta_{A/B}$ is defined as follows: For $Q_A \in Q_A$ (*i.e. if* $P_{A/B}$ is the initial phase):

$$\delta_{A/B}(q_A,a,u)=\begin{cases}\delta_A(q_A,a,u), & \text{if a}\in\Sigma,\\ \delta_A(q_A,\in,u)\cup\{(q_A,q_{B,0}),\in\} & \text{if a}=\in.\end{cases}$$

For $(q_A,q_B)\in Q_A \times Q_B)$ (is the guessing phase):

$$\delta_{A/B}((q_A,q_B),a,u)=\begin{cases}\phi, & \text{if a}\in\Sigma,\\ \cup_{b\in\Sigma_\in}\{((r_A,r_B),v):(r_A,v)\in\delta_A(q_A,b,u)\text{ and r}_A\in\delta_B(q_B,b)\}, & \text{if a}=\in.\end{cases}$$

• Therefore it can be claimed that $P_{A/B}$ accepts **w** if and only if there occurs a string **x** such that P_A accepts **wx** and *M* accepts **x**.

• For instance an acceptance calculation of $P_{A/B}$ on input **w**, all of **w** must be read during the 1st stage.

• The input symbols b that are predicted through the 2nd stage determine a string **x** that is recognized M and is such that **wx** is recognized by P_A.

• Contrariwise, if w is a string with the property that **wx** belongs to A for some x belong to B, then there is an acceptance calculation of $P_{A/B}$ in which w is read through the 1st stage, and the input x is predicted in the 2nd stage.

• In this instance the P_A components of the states determine an acceptance calculation on P_A on input **wx** and the *M*- components of the states determine an acceptance calculation of B on input **x**.

Hence, it is proved A/B is context free language by using $P_{A/B}$ from the above discussion.

Problem

Let ∑ = {a,b}. Give a CFG generating the language of strings with twice as many a's as b's. Prove that your grammar is correct.

Step-by-step solution

Step 1 of 4

Consider the language *L* that generates strings with twice as many a's as b's over the input alphabet $\Sigma = \{a, b\}$. The language does not care about the order in which the symbols a's and b's occur.

Comment

Step 2 of 4

The CFG for the language *L* is as follows:

$T \to Saab \mid aSab \mid aaSb \mid aabS \mid Saba \mid aSba \mid abSa \mid abaS \mid Sbaa \mid bSaa \mid baSa \mid baaS$

$S \to T \mid \varepsilon$

Comments (5)

Step 3 of 4

Now, prove the grammar is correct using the induction.

The smallest possible strings that are generated by the grammar are $\{aab, aba, baa\}$. Let w be the string from the set of smallest possible strings, such that $f_a(w) = 2f_b(w)$, where $f_a(w)$ is the number of a's in the string w and $f_b(w)$ is the number of b's in the string w. Hence, all the smallest possible strings have twice as many a's as b's.

$f_a(w_n) = 2f_b(w_n)$ .....(1)

(where w_n represents the string of length n)

Comment

Step 4 of 4

Now, show that $f_a(w_{n+1}) = 2f_b(w_{n+1})$ holds.

Obtain the string w_{n+1} by inserting any of the strings $\{\varepsilon, aab, aba, baa\}$ in to w_n. The insertions may result in addition of 0 *a*'s and 0 *b*'s or 2 *a*'s and 1 *b*.

Case 1:

When ε is inserted, (inserting 0 a's and 0 b's)

$f_a(w_{n+1}) = f_a(w_n) + f_a(\varepsilon) = f_a(w_n) + 0 = f_a(w_n)$ .....(2)

$f_b(w_{n+1}) = f_b(w_n) + f_b(\varepsilon) = f_b(w_n) + 0 = f_b(w_n)$ .....(3)

Now, substitute (2) and (3) in (1).

$f_a(w_n) = 2f_b(w_n)$

$f_a(w_{n+1}) = 2f_b(w_{n+1})$

Case 2:

When aab or aba or baa is inserted, (inserting 2 a's and 1 b)

$f_a(w_{n+1}) = f_a(w_n) + f_a(w) = f_a(w_n) + 2$ .....(4)

$f_b(w_{n+1}) = f_b(w_n) + f_b(w) = f_b(w_n) + 1$ .....(5)

(where w is aab or aba or baa)

Using (4),

$$\begin{aligned} f_a(w_{n+1}) &= f_a(w_n)+2 && (\text{from } (4)) \\ &= 2f_b(w_n)+2 && (\text{from } (1)) \\ &= 2(f_b(w_n)+1) \\ f_a(w_{n+1}) &= 2f_b(w_{n+1}) && (\text{from } (5)) \end{aligned}$$

From both the cases, it is proved that $f_a(w_{n+1}) = 2f_b(w_{n+1})$.

Hence, from the principle of mathematical induction the grammar is correct.

Therefore, the CFG generates the language of strings with twice as many *a* 's as *b* 's.

Comments (1)

Problem

Let
$C = \{x\#y \mid x, y \in \{0,1\}^* \text{ and } x \neq y\}$. Show that C is a context-free language.

Step-by-step solution

Step 1 of 2

The grammar *C* is defined as follows:

$C = \{ x\#y \mid x, y \in \{0, 1\}^* \text{ and } x \neq y \}.$

Comment

Step 2 of 2

• Given that a string $x\#y$ is in language *C* if and only if $x \neq y$ or strings *x* and *y* vary at some specific position; Such as for *i*-index value of *x* is different from the character value of *y*.

• It is very easy to form a Context free grammar which produce all the strings of the form $x\#y$ with $x \neq y$.

The CFG grammar is as follows:

$S \rightarrow A\#B \mid B\#A$
$A \rightarrow TAT \mid 0$
$B \rightarrow TBT \mid 1$
$T \rightarrow 0 \mid 1$

As the grammar for *C* is defined in terms of CFG. The language produces a string that contains $x\#y$, and *x* and *y* are different character for same index position.

Hence, it is proved that *C* is Context Free Language.

Comments (4)

Problem

Let $D = \{xy | x, y \in \{0,1\}^* \text{ and } |x| = |y| \text{ but } x \neq y\}.$ Show that D is a context-free language.

Step-by-step solution

Step 1 of 2

By the definition of **Context free language,** for showing that the language D is a CFL i.e. context free language, generate a context free grammar CFG G.

Consider the following grammar G:

$S \to AB | BA$

$A \to 0 | 0A0 | 0A1 | 1A0 | 1A1$

$B \to 1 | 0B0 | 0B1 | 1B0 | 1B1$

The given grammar L (G) generates the language in the form $w_1xw_2v_1yv_2$, where $|w_1| = |w_2| = k, |v_1| = |v_2| = l, x \neq y$ for $\Sigma = (0,1)^*$.

Comments (2)

Step 2 of 2

• By the definition, any language which is generated by a context-free grammar is termed as a context-free language.

• The grammar generated above is a Context Free Grammar. The language D can be generated using the above context free grammar G as follows:

• A string is in D iff it can be written as xy with $|x| = |y|$ s.t. for some i, the ith character of x and y are different from one another. The above grammar can be used to obtain the required string by generating the ith characters and filling up with the remaining characters.

• The generated language $w_1xw_2v_1yv_2$ can be subjected to nested induction over k and l with case distinction over pairs (x, y).

• Now, w2 and v1 can exchange symbols because both carry symbols that are independent of the rest of the string.

• Therefore, x and y in their respective half can have the same position, which implies L (G) = L because G doesn't impose any restrictions on its language.

Hence, the given language **D** is **context free language**.

Comment

Problem

Let E = {$a^i b^j$ | i ≠ j and $2^i \neq j$}. Show that E is a context-free language.

Step-by-step solution

Step 1 of 4

Given language E is defined as follows:

$E = \{a^i b^j \mid \text{i} \neq \text{j} \text{ and } 2\text{i} \neq j\}$

In order to show that the language E as a context free language.

Consider the language E as the language of the following three languages:

$E_1 = \{a^i b^j \mid j < i\}$

$E_2 = \{a^i b^j \mid i < j < 2i\}$

$E_3 = \{a^i b^j \mid j > 2i\}$

Since context free languages are closed under the union operation. So for proving that the language E is context free user has to show that the all three languages which are written above are closed under union operation.

Comment

Step 2 of 4

For the language E_1 build the grammar as follows:

$S \to aAB$

$A \to aA \mid \in$

$B \to aBb \mid \in$

The non-terminal symbol B generates $a^i b^j$ for $j \geq 0$ and the non-terminal symbol A generates a^i for $i \geq 0$. The starting non terminal symbol S always includes an ***a*** so that user can conclude that any string which is generated by using above grammar has more $a's$ than $b's$.

Conversely, if the string w belongs to language E_1, then w can be written as $w = aa^{i-j-1}a^j b^j$ and assume that A generates a^{i-j-1} and B generates $a^j b^j$.

Comment

Step 3 of 4

For the language E_3 build the grammar as follows:

$S \to ABb$

$A \to aAbb \mid \in$

$B \to Bb \mid \in$

Now the non-terminal symbol A generates $a^i b^{2i}$ for $i \geq 0$ and the symbol B generates b^j for $j \geq 0$. Assume that this grammar derives the string w. Now suppose s is used for storing the total number of time user replaces A with $aAbb$ and t is used for storing the total number of times user replaces B with Bb. Then $w = a^s b^{2s+t+1}$ as $s,t \geq 0,\ 2s+t+1 > 2s$ (using *j>2i* assume *i=s* and *2s+t+1=j*) and w belongs to E_3.

Conversely, if the string w belongs to language E_3, then w can be written as $w = a^i b^{2i} b^{j-2i-1} b$ and assume that A derives $a^i b^{2i}$ and the non-terminal symbol B generate b^{j-2i-1}.

Comment

Step 4 of 4

For the language E_2 build the grammar as follows:

$S \to aAb$

$A \to aAb \mid aAbb \mid abb$

Assume this grammar generates the string w and assume that s is used for storing the total number of times the rule $A \to aAb$ is used and is used for storing the total number of times the rule $A \to aAbb$. But $S \to aAb$ and $A \to abb$ are used exactly once in the derivation of the string *w.* Then $w = a^{s+t+2}b^{s+2t+3}$ by assuming $i = s+t+2, j = s+2t+3$, here $s,t \geq 0$ and user has $i = s+t+2 < s+2t+3 = j$ and $2i = 2s+2t+4 > s+2t+3 = j$. Hence the string w belongs to E_2.

Conversely, if the string w belongs to language E_2 then w can be written as $w = a^i b^j$. Assume that $s = j-i-1$ and $t = 2i-j-1$. As $i < j < 2i$ and $s,t \geq 0$ this grammar generates the string w by using the rule $S \to aAb$ *s* times and $A \to aAbb$ t times.

Since E_1, E_2 and E_3 languages are closed under union operation.

Therefore, the language E is context free language.

Comment

Problem

For any language A, let *SUFFIX*(A) = {vl uv ∑ A for some string u}. Show that the class of context-free languages is closed under the *SUFFIX* operation.

Step-by-step solution

Step 1 of 2

For any language *A*, it's suffix is defined as, $SUFFIX(A) = \{v \mid uv \in A \text{ for some string u}\}$. In order to prove that the CFLs are closed under *SUFFIX* operation, the push down automata (PDA) can be constructed or context free grammar (CFG) can be written for *SUFFIX* operation.

Comment

Step 2 of 2

To prove the context free languages closed under context free languages, take a context free language *A*. There exists a PDA and CFG for the language *A* since it is context free. Construct the PDA for *SUFFIX* operation of *A*. Let the PDA for the language *A* be *P*. The PDA for *SUFFIX*(*A*) be *M*. Following is the procedure to construct a PDA *M*.

• Create a copy of the PDA *P* and name it as *Q*. The PDA *Q* has the same transitions as *P* as it is a replica of *P*. The PDAs *P* and *Q* combined to form the PDA *M*.

• Modify the input part of transition in *Q* to ε without changing the stack symbol. If the input transition has $0,1 \to \varepsilon$, modify it to $\varepsilon,1 \to \varepsilon$. The input in the transition $0,1 \to \varepsilon$ is 0 and it is changed $\varepsilon,1 \to \varepsilon$ where the stack symbol ε is unchanged. In this step, just change the input part of each transition irrespective of the stack symbol.

• For each state in PDA *Q*, add a new transition $\varepsilon,\varepsilon \to \varepsilon$ to the corresponding state in PDA *P*. This means, for the input ε and stack symbol ε, the top of the stack will be ε. This step simply connects two PDAs.

• The start state of PDA *Q* should be the start state of the whole PDA *M*. Thus, the PDA *M* is the combination of two PDAs *Q* and *P*.

The PDA *M* simply ignores the alphabet of *u* and starts functioning when it identifies the first alphabet of *v* from which the second part of the PDA *M* (i.e., *P*) accepts the substring *v* (i.e., suffix). Thus, all the suffixes of the string belong to language *A* will be accepted by the PDA *M*.

Therefore, the CFLs are closed under *SUFFIX* operation.

Comment

Problem

Show that if G is a CFG in Chomsky normal form, then for any string w ∑ L(G) of length n ≥ 1, exactly 2n – 1 steps are required for any derivation of w.

Step-by-step solution

Step 1 of 1

Given that *G* is a CFG in Chomsky Normal Form (CNF).

The length of the string $w \in L(G)$ is n ≥ 1 for the string *w.*

It is required to show that exactly 2*n*-1 steps are required for the derivation of string w.

It can be proved applying the **induction method** by on the string w of length *n.*

For $n=1$: Consider a string "**a**" of length 1 in Chomsky normal form, so the valid derivation for this will be $S \to a$, where $a \in \sum$ and S is starting symbol.

The number of steps can be obtained as follows:

$$\begin{aligned} 2n-1 &= 2(1)-1 \\ &= 2-1 \\ &= 1 \end{aligned}$$

Hence it is true that 2n-1 (for n=1) steps are required to derive a string **a**.

For $n=k$: Take a string of length $k \geq n$ in Chomsky normal form, so valid derivation for this will take $2k-1$ steps.

The number of steps can be obtained as follows:

$$\begin{aligned} 2n-1 &= 2(k)-1 \\ &= 2k-1 \end{aligned}$$

Assume a string of length at most k≥1 terminal symbols and it has a string of length

n= $k+1$ is in Chomsky Normal Form

Since n>1, Consider a language as follows in CNF where derivation starts with start symbols S:

$$\begin{aligned} S &\to BC \\ B &\to *x \\ C &\to *y \end{aligned}$$

So length of the string starting with start symbol S is l*w*l = xy where lxl > 0 and l y l > 0.

Using the inductive hypothesis, for the above language in CNF the length of any derivation of string *w* must be

$$1+(2|x|-1)+(2|y|-1)=2|x|+2|y|+1-1-1=2(|x|+|y|)-1$$

Here n=$|x|+|y|$.

Since $B \to *x$ has a length of *lxl* and $C \to *y$ has a length of *lyl.*

Hence, it is proved that it requires **2n-1** steps required for the derivation of string

$w \in L(G)$ in Chomsky Normal Form(CNF).

Comment

Let $G = (V, \Sigma, R, \langle \text{STMT} \rangle)$ be the following grammar.

$$\begin{aligned} \langle \text{STMT} \rangle &\to \langle \text{ASSIGN} \rangle \mid \langle \text{IF-THEN} \rangle \mid \langle \text{IF-THEN-ELSE} \rangle \\ \langle \text{IF-THEN} \rangle &\to \texttt{if condition then } \langle \text{STMT} \rangle \\ \langle \text{IF-THEN-ELSE} \rangle &\to \texttt{if condition then } \langle \text{STMT} \rangle \texttt{ else } \langle \text{STMT} \rangle \\ \langle \text{ASSIGN} \rangle &\to \texttt{a:=1} \end{aligned}$$

$$\Sigma = \{\texttt{if}, \texttt{condition}, \texttt{then}, \texttt{else}, \texttt{a:=1}\}$$

$$V = \{\langle \text{STMT} \rangle, \langle \text{IF-THEN} \rangle, \langle \text{IF-THEN-ELSE} \rangle, \langle \text{ASSIGN} \rangle\}$$

G is a natural-looking grammar for a fragment of a programming language, but G is ambiguous.

a. Show that G is ambiguous.

b. Give a new unambiguous grammar for the same language.

Step-by-step solution

Step 1 of 3

Ambiguous and unambiguous grammar

a.

Consider $G = (V, \Sigma, R, \langle \text{STMT} \rangle)$

To show that G is ambiguous

It is required to find out two leftmost derivations:

$\langle \text{Stmt} \rangle$
$\Rightarrow \langle \text{if-then} \rangle$
$\Rightarrow$ if condition then $\langle \text{Stmt} \rangle$

$\Rightarrow$ if condition then $\langle \text{if-then-else} \rangle$
$\Rightarrow$ if condition then if condition then $\langle \text{Stmt} \rangle$ else $\langle \text{Stmt} \rangle$
$\Rightarrow$ if condition then if condition then $\langle \text{Assign} \rangle$ else $\langle \text{Stmt} \rangle$

$\Rightarrow$ if condition then if condition then a:=1 else $\langle \text{Stmt} \rangle$
$\Rightarrow$ if condition then if condition then a:=1 else $\langle \text{Assign} \rangle$
$\Rightarrow$ if condition then if condition then a:=1 else a:=1

Now the second derivation will be:

$\langle \text{Stmt} \rangle$
$\Rightarrow \langle \text{if-then-else} \rangle$
$\Rightarrow$ if condition then $\langle \text{Stmt} \rangle$ else $\langle \text{Stmt} \rangle$

$\Rightarrow$ if condition then $\langle \text{if-then} \rangle$ else $\langle \text{Stmt} \rangle$
$\Rightarrow$ if condition then if condition then $\langle \text{Stmt} \rangle$ else $\langle \text{Stmt} \rangle$
$\Rightarrow$ if condition then if condition then $\langle \text{Assign} \rangle$ else $\langle \text{Stmt} \rangle$

$\Rightarrow$ if condition then if condition then a:=1 else $\langle \text{Stmt} \rangle$
$\Rightarrow$ if condition then if condition then a:=1 else $\langle \text{Assign} \rangle$
$\Rightarrow$ if condition then if condition then a:=1 else a:=1

Comment

Step 2 of 3

In both cases when it takes “if-then” or “if-then else” result is same. Hence both have same left derivation and it is ambiguous.

Comment

Step 3 of 3

For making unambiguous grammar, it is required to make correct interpretation of the above two. For that when "if-then-else" is derived, it should not allow then part for deriving "if-then”. So introduce a new variable $\langle \text{Stmt} \rangle$ the new grammar is:

$$\langle \text{Stmt} \rangle \Rightarrow \langle \text{if-then} \rangle \mid \langle \text{Assign} \rangle \mid \langle \text{if-then-else} \rangle$$

$$\langle \text{if-then-else} \rangle \Rightarrow \text{ if condition then } \langle \text{Stmt1} \rangle \text{ else } \langle \text{Stmt} \rangle$$

$$\langle \text{Stmt1} \rangle \Rightarrow \langle \text{if-then-else} \rangle \mid \langle \text{Assign} \rangle$$

$$\langle \text{if-then} \rangle \Rightarrow \text{if condition then } \langle \text{Stmt} \rangle$$

$$\langle \text{Assign} \rangle \Rightarrow \text{a:=1}$$

Comments (2)

Problem

Give unambiguous CFGs for the following languages.

a. {wl in every prefix of w the number of a's is at least the number of b's}

b. {wl the number of a's and the number of b's in w are equal}

c. {wl the number of a's is at least the number of b's in w}

Step-by-step solution

Step 1 of 4

First step is finding a grammar for finding an unambiguous CFG whether it is ambiguous or unambiguous.

• Grammar for the given problem is

$$S \to SS|(S)|\varepsilon$$
$$S \to a \mid b$$

• But the given grammar is ambiguous as there is more than one parse tree is possible for the above grammar.

• After finding the grammar finds that whether it is ambiguous or not.

• If the grammar is ambiguous then set some priority rules for selecting derivation tree for the grammar and after that set production rules for the grammar.

• Now after setting some priority rules the grammar can be:

$$S \to (S)S|\varepsilon$$
$$S \to a \mid b$$

• But this is also wrong grammar as in this grammar for every prefix of any input string w the number of a's does not at least the number of b's.

• Final grammar in which for every prefix of any input string *w* the number of a's is at least the number of b's is as shown:

$$S \to (S)S|(S|\varepsilon$$
$$S \to a \mid b$$

Comment

Step 2 of 4

a)

The input alphabet *a* is prefixes at least the number of b's that means if the length of the input string is one then that must be a.

For the two or more inputs the possible strings are *aa*, *ab*, *aaa*, *aab* and so on.

So, the unambiguous grammar (CFG) is given below:

$$S \to aS \mid a \mid b \mid \varepsilon$$

The production for a regular expression *aaab*.

$$\begin{aligned} S &\to aS \\ &\to aaS \\ &\to aaaS \\ &\to aaab \end{aligned}$$

This grammar is an unambiguous grammar because there is no possible way to get same regular expression.

Comments (1)

Step 3 of 4

b)

Consider the following unambiguous CFG:

$S \to aSb|bSa \mid \varepsilon$

In the above grammar, produ ion rules $S \to aSb$ and $S \to bSa$ generates the equal number of terminals a's and b's.

$$S \to aSb$$
$$\to a(aSb)b$$
$$\to aa(aSb)bb$$
$$\to aaabbb$$

Comments (5)

Step 4 of 4

c)

In the given question, the number *a's* is at least the number of *b's* that means for a single character input *a* must be the input alphabet.

Consider the following unambiguous CFG:

$S \to aSb|bSa|aS|\varepsilon$

• In the above grammar, production rules $S \to aSb$ and $S \to bSa$ generates the equal number of terminals *a's* and *b's*.

• The production rule $S \to aS$ is used for generating the terminal *a's* as many as user wants.

• The production rule $S \to \varepsilon$ is used for generating the equal number of terminals *a's* and *b's*.

$$S \to aSb|bSa|aS|\varepsilon$$
$$\to a(aSb)b$$
$$\to aa(bSa)bb$$
$$\to aababb$$

Comments (3)

Problem

Show that the language A in Exercise 2.9 is inherently ambiguous.

Step-by-step solution

Step 1 of 4

If a string w has two or more different derivations, it is ambiguous. The ambiguous grammar will generate some string ambiguously.

Some context free grammars can only be generated by the ambiguous grammars. Such languages are called ***inherently ambiguous***.

Comment

Step 2 of 4

Given context-free grammar that generates the language

$A = \left\{ a^i b^j c^k \mid i = j \text{ or } j = k \text{ where } i, j, k \geq o \right\}$

Let G be the context-free grammar generating the language A and G is an ambiguous grammar. Now, show that A is inherently ambiguous.

Let p be the pumping length of the context-free grammar G given by the pumping lemma.

Let k be an integer such that $k = p! = p(p-1)(p-2)...1$.

Let s be a string such that $s = a^k b^k c^k$.

To show that A is inherently ambiguous, it is enough to show that s has two or more different leftmost derivations. It is required to know that those parse trees are equivalent to derivations and s has two parse trees.

Comment

Step 3 of 4

Let $s_1 = a^k b^p c^p$ and $s_2 = a^p b^p c^k$ are two strings in A with parse trees τ_1 and τ_2 respectively.

Take τ_1 and remove all nodes that have only $a's$ below them. Since there are k $a's$, the sub tree will have only p $b's$ and p $c's$, in total having $2p$ leaves.

Because $2p > p$ it is known that the sub tree contains a path with repeated variable. Let R be the repeated variable. Divide s into $uvxyz$ using R and R can be chosen such that $|vxy| \leq p$. Each of the strings v and y can contain only one of the symbols, if not uv^2xy^2z will not belong to A.

Also y contain no $a's$, because R was on a path to a, b or c. Hence v must be $b's$ and y must be $c's$ and they must have an equal length l.

Consider $d = k/l$ (which must be an integer since $l \leq p$ and $k = p!$) the string $s = vu^d xy^d$ has a parse tree where most $b's$ have a parent node in common with $c's$ but not with $a's$.

Comment

Step 4 of 4

Similarly, consider τ_2, the parse tree is obtained for s $b's$ having a parent node in common with $a's$ but not with $c's$.

So s has two distinct parse trees. Hence s has two or more different leftmost derivations and G can be generated by the ambiguous grammars only.

Therefore, the language A is inherently ambiguous.

Comment

Problem

Use the pumping lemma to show that the following languages are not context free.

a. $\{0^n1^n0^n1^n \mid n \geq 0\}$

b. $\{0^n\#0^{2n}\#0^{3n} \mid n \geq 0\}$

c. $\{w\#t \mid w \text{ is a substring of } t, \text{ where } w, t \in \{\mathtt{a}, \mathtt{b}\}^*\}$

d. $\{t_1\#t_2\#\cdots\#t_k \mid k \geq 2, \text{ each } t_i \in \{\mathtt{a}, \mathtt{b}\}^*, \text{ and } t_i = t_j \text{ for some } i \neq j\}$

Step-by-step solution

Step 1 of 5

a) Consider the language $B = \{0^n1^n0^n1^n \mid n \geq 0\}$.

Let p be the pumping length of B given by the pumping lemma.

To show that B is not a CFL, it is enough to show that a string $s = 0^p1^p0^p1^p$ cannot be pumped.

Consider s is of the form $uvxyz$.

• If both v and y contain at most one type of alphabet symbol, the string will be of the form uv^2xy^2z runs of 0's and 1's of unequal length. Hence the string s cannot be a member of B.

• If either v or y contains more than one type of alphabet symbol, the string will be of the form uv^2xy^2z which does not contain the symbols in correct order. Hence the string s cannot be a member of B.

Since the string s cannot be pumped without violating the pumping lemma condition, B is not a CFL (context-free language).

Comment

Step 2 of 5

b) Consider the language $B = \{0^n\#0^{2n}\#0^{3n} \mid n \geq 0\}$.

Let p be the pumping length of B given by the pumping lemma.

To show that B is not a CFL, it enough to show that a string $s = 0^p\#0^{2p}\#0^{3p}$ cannot be pumped.

Consider s is of the form $uvxyz$.

Neither v nor y can contains $\#$, otherwise uv^2xy^2z contains more than two $\#$s. If the string s is divided into three segments by $\#$'s at least one of the segments $0^p, 0^{2p}$ and 0^{3p} is not contained within either v or y.

Because the length ratio of the segments is not maintained as 1:2:3, xv^2wy^2z is not in B.

Hence the string s cannot be a member of B.

Since the string s cannot be pumped without violating the pumping lemma condition, B is not a CFL (context-free language).

Comment

Step 3 of 5

c) Consider the language $B = \{w\#t \mid w \text{ is a substring of } t, \text{ where } w, t \in \{a, b\}^*\}$.

Let p be the pumping length of B given by the pumping lemma.

To show that B is not a CFL, it enough to show that a string $s = a^pb^p\#a^pb^p$ cannot be pumped.

Consider s is of the form $uvxyz$.

• Neither v nor y can contair $\#$, otherwise uv^0xy^0z does not contain $\#$. Hence the string s cannot be a member of B.

• If both v and y are nonempty and occur on the left-hand side of $\#$, the string uv^2xy^2z is longer on the left-hand side of $\#$. Hence the string s cannot be a member of B.

• Similarly, if both v and y are nonempty and occure on the right-hand side of $\#$, the string uv^0xy^0z is longer on the rifht hand side of $\#$. Hence the string s cannot be a member of B.

• If only one of v and y is nonempty we can treat them as if both occurred on the same side of $\#$. Hence the string s cannot be a member of B.

• In the remaining case if both v and y are nonempty and include the $\#$, then by the third pumping lemma condition $|vxy| \leq p$, we have v consists of b's and y consists of a's. Hence uv^2xy^2z contains more b's on the left-hand side of the $\#$. Hence the string s cannot be a member of B.

Since the string s cannot be pumped without violating the pumping lemma condition, B is not a CFL (context-free language).

Comments (3)

Step 4 of 5

d) Consider the language

$$B = \left\{ t_1 \# t_2 \# \cdots \# t_k \mid k \geq 2,\ t_i \in \{a,b\}^*, \text{ and } t_i = t_j \text{ for some } i \neq j \right\}.$$

Let p be the pumping length of B given by the pumping lemma.

The t_i can be equal to t_j for different *i* and *j* values. Hence, the same terms will be appeared in the string *s* separated by *#*.

For example, if the *k* value is 2 the string *s* can be *ab#ab.* The string *s* has the same term *ab* for different *k* values separated by *#*. The language generates the strings that contains same terms comprised of *a, b* separated by *#*. The strings that can be generated from the language *B* are *ab#ab, b#b#b, aba#aba#aba#aba, …* etc.

To show that B is not a CFL, it enough to show that a string $s = a^p b^p \# a^p b^p$ cannot be pumped.

Consider s is of the form $uvxyz$.

• Neither v nor y can contains $\#$, otherwise uv^0xy^0z does not contain $\#$. Hence the string s cannot be a member of B.

• If only one of v and y is nonempty we can treat them as if both occurred on the same side of $\#$. Hence the string s cannot be a member of B.

• If both v and y are nonempty and occur on the left-hand side of $\#$, the string uv^2xy^2z is longer on the left-hand side of $\#$. Hence the string s cannot be a member of B.

• If both v and y are nonempty and occur on the right-hand side of $\#$, the string uv^0xy^0z is longer on the rifht hand side of $\#$. Hence the string s cannot be a member of B.

•

Comment

Step 5 of 5

In the remaining case if both v and y are nonempty and include the $\#$, then by the third pumping lemma condition $|vxy| \leq p$, we have v consists of b's and y consists of a's. Hence uv^2xy^2z contains more b's on the left-hand side of the $\#$. Hence the string s cannot be a member of B.

Since the string s cannot be pumped without violating the pumping lemma condition, B is not a CFL (context-free language).

Comment

Problem

Let B be the language of all palindromes over {0,1} containing equal numbers of 0s and 1s. Show that B is not context free.

Step-by-step solution

Step 1 of 3

Let B be the language of all palindromes over $\{0, 1\}$ containing the equal numbers of 0's and 1's . To prove B is not a context free language by taking a contradiction.
Assume that B is a context free language.

Since B is a context free language, then by pumping lemma, there is a number p (the pumping length) where, if s is any string in B of length at least p , then s may be divided into five pieces $s = uvxyz$ satisfying the conditions

1. for each $i \geq 0, uv^i xy^i z \in A$,
2. $|vy| > 0$, and
3. $|vxy| \leq p$.

Comment

Step 2 of 3

Case 1:

Now select a string $s = 0^n 1^{2n} 0^n$.

Clearly s is a member of B of length at least p.

Assume the value of n=2 for the string *s*.

$s = 0^2 1^4 0^2$

$s = 00111100$

The string s can be divided into *uvxyz* as follows:

$$\underbrace{00}_{u}\ \underbrace{11}_{v}\ \underbrace{1}_{x}\ \underbrace{10}_{y}\ \underbrace{0}_{z}$$

Now apply the first condition of the pumping lemma.

$$\text{for each } i \geq 0, uv^i xy^i z \in A$$

For i = 2:

$$\underbrace{00}_{u}\left(\underbrace{11}_{v}\right)^2\ \underbrace{1}_{x}\left(\underbrace{10}_{y}\right)^2\ \underbrace{0}_{z}$$

$$\underbrace{00}_{u}\left(\underbrace{1111}_{v}\right)\ \underbrace{1}_{x}\left(\underbrace{1010}_{y}\right)\ \underbrace{0}_{z}$$

Assume the obtained string 00 1111 1 1010 0 as s'.

The obtained string s' is not a palindrome after applying the first condition of pumping lemma and $s' \notin B$. In the pumped string, the number of 0's and 1's is not equal.

So, the language B is not following the condition1 of the pumping lemma.

Comment

Step 3 of 3

Case 2:

The same string is selected as $s = 0^n 1^{2n} 0^n$

Clearly s is a member of B of length at least p.

Assume the value of n=2 for the string s.

$s=0^2 1^4 0^2$

s=00111100

The string s can be divided into *uvxyz* as follows:

$$\frac{0}{u}\ \frac{01}{v}\ \frac{11}{x}\ \frac{1}{y}\ \frac{00}{z}$$

Now apply the first condition of the pumping lemma.

$$\text{for each } i \geq 0, uv^i xy^i z \in A$$

For i=2:

$$\frac{0}{u}\left(\frac{01}{v}\right)^2\ \frac{11}{x}\left(\frac{1}{y}\right)^2\ \frac{00}{z}$$

$$\frac{0}{u}\left(\frac{0101}{v}\right)\ \frac{11}{x}\left(\frac{11}{y}\right)\ \frac{00}{z}$$

Assume the obtained string 0 0101 11 11 00 as s'.

The obtained string s' is not a palindrome after applying the first condition of pumping lemma and $s' \notin B$. In the pumped string, the number of 0's and 1's is not equal.

Hence, the assumption B is a context free language is wrong.

Therefore, by the two cases it can be proved that B is not a context free language.

Comments (3)

Problem

Let ∑ = {1, 2, 3, 4} and C = {w ? ∑*| in w, the number of 1s equals the number of 2s, and the number of 3s equals the number of 4s}. Show that C is not context free.

Step-by-step solution

Step 1 of 1

Context Free Language

Consider the language:

$$C=\left\{\begin{matrix} w\in\{0,1,2,3,4\}^{*} \mid \text{in w the number of 1s and the number of 2s} \\ \text{are equal,the number of 3s and the number of 4s are equal.} \end{matrix}\right\}$$

On the contrary consider C is context free. So, C has a pumping length p.

Take $s=1^p3^p2^p4^p\in C$ with $|s|>p$.

Therefore, there exist $uvxyz$ such that

(a) $uv^ixy^iz\in C$ for all $i\geq 0$ .....(1)

(b) $vy>0$ .....(2)

(c) $vxy\leq p$ .....(3)

Now it has to prove all the cases by contradiction, no matter what the value of $uvxyz$.

Case 1: If vxy contains a 1. Then $uv^2xy^2z\notin C$, since it cannot be same number of 1s and 2s. Hence due to equation (3), vxy cannot contain any 2s.

Case 2: If vxy contains a 2. Then $uv^2xy^2z\notin C$, since it cannot be same number of 1s and 2s. Hence due to equation (3), vxy cannot contain any 1s.

Case 3: If vxy contains a 3. Then $uv^2xy^2z\notin C$, since it cannot be same number of 3s and 4s. Hence due to equation (3), vxy cannot contain any 4s.

Case 4: If vxy contains a 4. Then $uv^2xy^2z\notin C$, since it cannot be same number of 3s and 4s. Hence due to equation (3), vxy cannot contain any 3s.

Hence from equation 2, it contradicts equation 1 in all the cases which shows *C* is not context free language.

Comment

Problem

Show that F = {$a^i b^j$ I i = k_j for some positive integer k} is not context free.

Step-by-step solution

Step 1 of 3

The language given in the question is as follows:

Language $F = \left\{ a^i b^j \middle| i = kj \text{ for some positive integer } k \right\}$.

A context free language (CFL) is generated by a context free grammar. **In order to prove that a language is not a CFL, Pumping Lemma is used.**

Comment

Step 2 of 3

Proof that Language L is not a Context Free Language:

Assume that *F* is CFL. Obtain a contradiction using pumping lemma to prove that the assumed statement is false.

• Let *p* be the pumping length for *F* that is guaranteed to exit by pumping lemma.

• Select string $s = a^p b^{2p} \in F$ where $k = 2$ and divide the string *s* into *uvxyz*.

• According to pumping lemma, *v* and *y* in string cannot be empty sets.

• Now, consider these two case, depending on whether substring *v* and *y* contain more than one type of alphabet symbol:

1. **Both *v* and *y* contain only one type of alphabet symbol:** In this case, both $v \text{ and } y$ does not contain mixed a's and b's. Thus, the string uv^2xy^2z cannot contain equal number of a's and b's. Also, a pattern for a's and b's can be obtained that contains a relation between number of a's and b's. So, none of the conditions of lemma violates and thus it does not contradict.

For example:

Consider $uv^i xy^i z$ such that $v = a, y = bb$ and $u = x = z = \phi$. Thus, the strings generated will be $s = abb, aabbbb, aaabbbbbb...$

All the strings *s* are a member of *F*. Hence, no contradiction is obtained.

2. **Either of *v* or *y* contains more than one type of alphabet symbols**: In this case, both $v \text{ and } y$ contain mixed a's and b's. Thus, the string uv^2xy^2z will contain strings with some order of ab followed by some order of ab again. Thus, it produces a wrong order of strings thereby producing a contradiction

For example:

Consider $uv^i xy^i z$ such that $v = ab \text{ and } y = b$ and $u = x = z = \phi$. Thus the strings generated will be $s = abb, ababbb...$

The string s = $ababbb$ is not member of *F*. This violates our assumption and thus, a contradiction is obtained.

Comments (1)

Step 3 of 3

The second case results in a contradiction. Hence, the assumption that *F* is context free language is false and therefore, F is not a context free language.

Comments (2)

Problem

Consider the language B = L(G), where G is the grammar given in Exercise 2.13. The pumping lemma for context free languages, Theorem 2.34, states the existence of a pumping length p for B. What is the minimum value of p that works in the pumping lemma? Justify your answer.

THEOREM **2.34**

Pumping lemma for context-free languages If A is a context-free language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided into five pieces s = uvxyz satisfying the conditions

$$\textbf{1. for each } i \geq 0,\ uv^i xy^i z \in A,$$
$$\textbf{2. } |vy| > 0, \text{ and}$$
$$\textbf{3. } |vxy| \leq p.$$

When s is being divided into uvxyz, condition 2 says that either v or y is not the empty string. Otherwise the theorem would be trivially true. Condition 3 states that the pieces v, x, and y together have length at most p. This technical condition sometimes is useful in proving that certain languages are not context free.

Step-by-step solution

Step 1 of 3

Given:

The minimum value of the pumping length p for the language $B = L(G)$

Comment

Step 2 of 3

Finding minimum length of $\mathbf{p}$:

The language $B = L(G)$ is defined by the

Grammar $G = (V, \Sigma, R, S)$ where $V = \{S, T, U\}$; $\Sigma = \{0, \#\}$ and the given set of rules R :

$S \to TT \mid U$
$T \to 0T \mid T0 \mid \#$
$U \to 0U00 \mid \#$

Theorem 2.34 states that the pumping lemma p, for a context-free language A, is the minimum length of any string s in A such that it may be split into five pieces $s = uvxyz$. The string s will also fulfill the following three conditions:

1. For all $i \geq 0$, the string $uv^i xy^i z$ is a part of the context-free language A.

2. The strings which are pumped, which are v and y, cannot both be the empty string ε, that is $|vy| > 0$.

3. The combined length of the strings lying inside u and z must not be greater that the pumping length p. In other words $|vxy| \leq p$.

The string $s = uvxyz$ can be taken as $\#\#0$, with the substrings being as follows:

$u = v = z = \varepsilon$
$x = \#\#$
$y = 0$

Thus the pumping length p is $|vxy| = |\#\#0| = 3$

Since, $|vy| = |\varepsilon 0| = |0| = 1 > 0$, condition 2 of the theorem holds.

Condition 3 of Theorem 2.34 is also valid as $|vxy| = |\varepsilon \#\#0| = |\#\#0| = 3 \leq p = 3$.

To meet condition 1 it has to be proven that $uv^i xy^i z \in B$ for $i \geq 0$.

The string $uv^i xy^i z$, where $u = \varepsilon, v = \varepsilon, x = ##, y = 0, z = \varepsilon$, can be expressed as the regular expression $##0^i$.

The derivation for the case when $i = 0$ is:

$$S \Rightarrow TT \Rightarrow \#T \Rightarrow ##$$

So the string $s = uv^0 xy^0 z$ lies in the language B.

When $i > 0$ the string $s = uv^i xy^i z$ will also lie in B as the derivation will be:

$$S \Rightarrow TT \Rightarrow \#T \Rightarrow \#T0 \overset{*}{\Rightarrow} \#T0^+ \Rightarrow ##0^+$$

Condition 1 has been proven true as $uv^i xy^i z \in B \forall i \geq 0$.

Comment

Step 3 of 3

Conclusion:

Thus the string satisfies all three conditions of Theorem 2.34 for a context-free language. Therefore, **the minimum value is 3** for the pumping lemma p.

Comment

Problem

Let G be a CFG in Chomsky normal form that contains b variables. Show that if G generates some string with a derivation having at least 2^b steps, L(G) is infinite.

Step-by-step solution

Step 1 of 3

CFG:

• A finite set of grammar rules is known as **CFG (context free grammar)**. It consisting of that is quadruple (*N*, *T*, *P*, *S*).

• Where, set of non-terminal symbol represented by *N*.

• **T** is group of terminal $N \bigcap T = \text{NULL}$.

• **P** is group of rule, $P : N \to (N \bigcup T)^*$.

• **S** is start symbol.

Comment

Step 2 of 3

CNF:

• In **CNF (Chomsky normal form)** we have a restriction on the length of RHS, which is a CFG (context free grammar) is in Chomsky normal form if the productions are in the following terms:

$U \to u$
$U \to VW$

• Where *U*, *V* and *W* are non-terminals and *u* is a terminal.

Comment

Step 3 of 3

• At most two terminal can generate in every deviation.

• In any parse string with use of *G*.

• An internal node can have at most two children.

• Parse tree with height k has at most o $2^k - 1$ internal node.

• If several string generated by *G* with a derivation taking at least 2^b steps.

• At least 2^b inner node takes parse tree of that string.

• At least consuming $b+1$ height in Parse tree.

• It exists a path from root to leaf containing $b+1$ variable.

• In this one variable occurring twice.

Hence, user can use the technique in proof of pumping lemma to construct infinity many string which are all in L(G).

Comments (1)

Problem

Give an example of a language that is not context free but that acts like a CFL in the pumping lemma. Prove that your example works. (See the analogous example for regular languages in Problem 1.54.)

Problem 1.54

Consider the language $$F = \{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid i, j, k \geq 0 \text{ and if } i = 1 \text{ then } j = k\}.$$

a. Show that F is not regular.

b. Show that F acts like a regular language in the pumping lemma. In other words, give a pumping length p and demonstrate that F satisfies the three conditions of the pumping lemma for this value of p.

c. Explain why parts (a) and (b) do not contradict the pumping lemma.

Step-by-step solution

Step 1 of 6

Consider the following details:

Let $F = \{a^i b^j c^k d^m \mid i, j, k, m \geq 0 \text{ and if } i = 1 \text{ then } j = k = m\}$.

Comment

Step 2 of 6

Proof that F is not a context free Language (by contradiction):

• Suppose F is context free, then $F \cap \{ab^i c^j d^k \mid i, j, k \geq 0\} = \{ab^i c^i d^i \mid i \geq 0\} = G$ is context free since $\{ab^i c^j d^k \mid i, j, k \geq 0\}$ is the language of the regular expression $ab^* c^* d^*$ which is regular, and also the intersection of a context free language with a regular language is context free.

• By showing that G cannot be context free using pumping lemma, it will contradict the fact that F is context free.

Comment

Step 3 of 6

Suppose the pumping length of G is p and take $s = ab^p c^p d^p \in G$ with $|s| > p$.

There exists *u*, *v*, *x*, *y*, *z* such that $s = uvxyz$ and,

(1) $uv^n xy^n z \in G$ for all n ≥ 0,

(2) $|vy| > 0$ and

(3) $|vxy| \leq$ p.

Comment

Step 4 of 6

For any valid choice of *uvxyz* that $uv^2xy^2z \notin G$, take $i = 0$. Then,

Case 1: If *v* or *y* contains *a*, then uv^2xy^2z will have more than one a and thus is not in *G*.

Case 2: If *v* and *y* do not contain a, then from $|vxy| \leq p$, *vxy* can have at most two other symbols from *b*, *c* or *d*.

Therefore, uv^2xy^2z will not have the same number of the three symbols.

Thus, G is not a CFL and therefore F is also not a CFL.

Comment

Step 5 of 6

Proof that F is a context free language using the pumping lemma:

Let $p = 2$. Now for any string $s \in F$, with $|s| \geq 2$, can be written as $uvxyz$ such that

(1) $uv^n xy^n z \in G$ for all $n \geq 0$,

(2) $|vy| > 0$

(3) $|vxy| \leq p$

Now,

Case 1: $s = a^i b^j c^k d^m$ with $i \neq 2$ and $i + j + k + m \geq 2$.

In this case, let $u = v = x = \in$, *y* is the first symbol in *s* and *z* be the remaining symbols.

Then (2) and (3) hold, and $uv^n xy^n z$ will have either zero as (*i* = 0 or *i* = 1 and *n* = 0) or more than one a followed by a string of the form $b^j c^k d^m$, so it will remain in *F* and therefore (1) holds.

Case 2: s = $a^2 b^j c^k d^m \in F$ for some *j, k,* $m \geq 0$ (so $|s| \geq 2$). Take $u = v = x = \in$, $y = a^2$ (in this case *a* would not work if any *j, k, m* > 0 because there is pumping down, but if a would be taken as the first symbol then it will work), and $z = b^j c^k d^m$. Then (2) and (3) hold and $xy^i z = a^{2+2(i-1)} b^j c^k d^m \in F$, therefore, (1) also holds.

Thus, in either cases, the conditions of the pumping lemma hold.

Comment

Step 6 of 6

Therefore, F , which is not a Context Free Language, satisfies the pumping lemma.

Comment

Problem

Prove the following stronger form of the pumping lemma, wherein *both* pieces v and y must be nonempty when the string s is broken up.

If A is a context-free language, then there is a number k where, if s is any string in A of length at least k, then s may be divided into five pieces, s = uvxyz, satisfying the conditions:

a. for each $i \geq 0$, $uv^i xy^i z \in A$,

b. $v \neq \varepsilon$ and $y \neq \varepsilon$, and

c. $|vxy| \leq k$.

Step-by-step solution

Step 1 of 8

It has to be proven that if A is a context-free language, there is a number k where is a string s such that $s \in A$ and $|s| \geq k$, then the string s can be split into five pieces $s = uvxyz$ such that the following conditions hold:

a. For each $i \geq 0, uv^i xy^i z \in A$,

b. $v \neq \varepsilon$ and $y \neq \varepsilon$, and

c. $|vxy| \leq k$.

Comment

Step 2 of 8

Now, define a context-free grammar $G = (V, \Sigma, R, T)$ for the context-free language A such that the right-hand side of a rule b has maximum number of symbols, with b being at least 3 symbols large.

• When the height of parse tree is k then the length of the string s generated is at least b^k. This is as there will be at most b leaves 1 step from S, at most b^2 at a 2 steps and at-most b^k leaves at a depth of k steps.

• Alternatively if the length of string is b^k then all smallest possible parse trees for the string must be at least k deep. Thus any path in a parse tree for the string s will have a path that is at least k terminals long. The path will start with $k-1$ variables and terminate with a single terminal.

Comment

Step 3 of 8

The number of variables in the grammar G is $|V|$. A string s is taken whose length is at least $|V|+1$. Hence any of the smallest possible parse trees will have a depth that is at least $b^{|V|+1}$ steps. Or in other words all paths will be at least $|V|+1$ long, which will start with $|V|+1$ variables and end with a terminal.

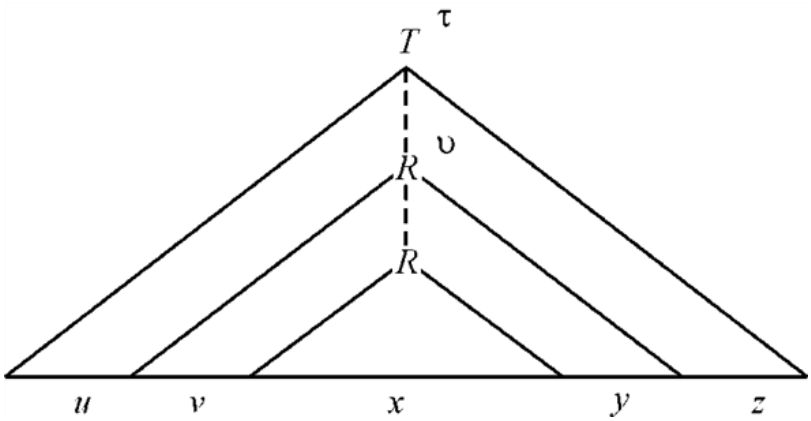
• There are only $|V|$ variables in the grammar G. The pigeonhole principle is applied with the variables (or non-terminal nodes) being the holes and with the pigeons being the variables encountered on any path in the tree.

• There are $|V|+1$ pigeons going into $|V|$ holes, so at least one hole must contain more than one pigeon. That is at least one variable occurs more once in the longest path of the smallest possible parse tree τ.

Comment

Step 4 of 8

Let this variable be R. Use the parse tree τ from the start symbol T to divide the string s into five pieces, of the form $uvxyz$. Consider the figure which is given below:

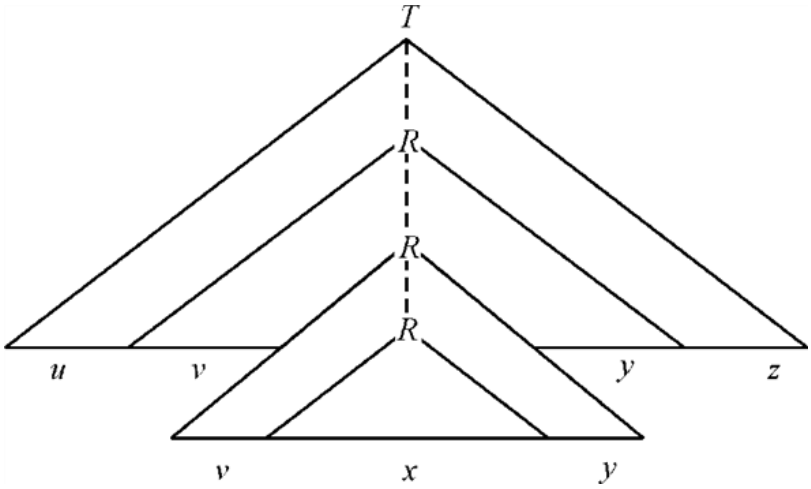


The parse tree υ at the first R vertex hit by the start symbol T breaks down the substring vxy into three strings of terminals. Therefore this tree υ can be repeated any number of times. This is shown by considering a couple of examples.

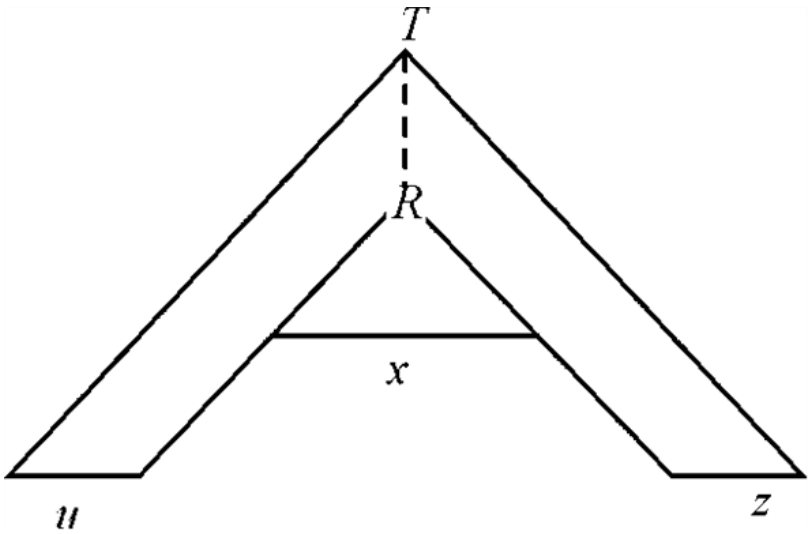
Comment

Step 5 of 8

Consider the figure which is given below. It shows a tree for the string uv^2xy^2z is:



Now, the figure which is given below, shows a tree for the string uxz is:



In this way, it can be said that "The first condition $uv^ixy^iz \in A, i \geq 0$ has been shown to be true".

Comment

Step 6 of 8

In the second condition, it is stated that **either of** v **and** y **cannot be the empty string** ε. The parse tree τ is the smallest possible tree.

• If $v = \varepsilon$ or if $y = \varepsilon$, then one or more vertices can be removed from the sub-trees ending in ε in the parse tree τ.

•

Comments (1)

Step 7 of 8

This is a contradiction as it is not possible to remove vertices from the tree τ as it is the smallest possible parse tree.

It has been shown that $v \neq \varepsilon$ **and** $y \neq \varepsilon$ **.**

Comment

Step 8 of 8

Lastly it has to be proven that $|vxy| \leq k$. Alternatively, the depth of the tree υ has to be at least k. Since the root node R repeats and the path has $|V|+1$ variables. **Thus depth of the tree** υ **is at least** $|V|+1$ **and its yield will be at most** $b^{|V|+1} = k$**.**

Comment

Problem

Refer to Problem 1.41 for the definition of the perfect shuffle operation. Show that the class of context-free languages is not closed under perfect shuffle.

Problem 1.41

For languages A and B, let the ***perfect shuffle*** of A and B be the language

$$\{w \mid w = a_1 b_1 \cdots a_k b_k, \text{ where } a_1 \cdots a_k \in A \text{ and } b_1 \cdots b_k \in B, \text{ each } a_i, b_i \in \Sigma\}.$$

Show that the class of regular languages is closed under perfect shuffle.

Step-by-step solution

Step 1 of 1

Context free language is the language which is generated by CFG or the context free grammar. It is possible to get different types of context free languages from different types of context free grammar.

Consider the languages A and B as given below:

Language *A* $\{0^k 1^k | k \geq 0\}$

Language *B* $\{a^k b^{3k} | k \geq 0\}$

After the perfect shuffle, other language (*C*) obtained will be $\{(0a)^k (0b)^k (1b)^{2k} \mid k >= 0\}$

It is clear that languages *A* and *B* are context free languages but language *C* is not a context-free language. It can be proved by contradiction follow:

• Suppose that the language defined for the shuffle of *A* and *B* is context free and let *p* be the length of its pumping lemma and *s* is string $(0a)^p (0b)^p (1b)^{2p}$.

• As string is longer as compared with pumping lemma and string is a part of language *C*, divide $s = uvxyz$.

• In language *C*, string is equal to one-fourth part of 1s and one-eighth part of *a*'s. For uv^2xy^2z, to have the same property, it should contain *a*'s and 1's. But it is not possible because both are separated by another symbol 2*p*. Also $|vxy| \leq p$.

Thus, it can be concluded and deduced that context free languages or CFL's are not closed under perfect shuffle.

Comment

Problem

Refer to Problem 1.42 for the definition of the shuffle operation. Show that the class of context-free languages is not closed under shuffle.

Problem 1.42

For languages A and B, let the ***shuffle*** of A and B be the language

$$\{w \mid w = a_1 b_1 \cdots a_k b_k, \text{ where } a_1 \cdots a_k \in A \text{ and } b_1 \cdots b_k \in B, \text{ each } a_i, b_i \in \Sigma^*\}.$$

Show that the class of regular languages is closed under shuffle.

Step-by-step solution

Step 1 of 2

Context free language is the language which is generated by CFG or the context free grammar. It is possible to get different types of context free languages from different types of context free grammar.

Consider two languages as *P* and *Q* defined over the alphabet and let the shuffle operation defined for these two languages is:

$$\left\{ w \mid w = p_1 q_1 \ldots\ldots p_k q_k, \text{where } p_1 \ldots\ldots p_k \in P, q_1 \ldots\ldots q_k \in q \text{ and each } a_j b_j \in \Sigma^* \right\}$$

Now, the two languages under consideration can be defined as follows:

$$P = \left\{ w \in \{0,1\}^* \mid n_0(w) = n_1(w) \right\}$$

$$Q = \left\{ w \in \{p,q\}^* \mid n_p(w) = n_q(w) \right\}$$

Where, in the above grammar $n_1(w)$ is the number of one's in *w* and $n_0(w)$ is the number of zeros present in the string *w*. Also, $n_p(w)$ and $n_q(w)$ denotes the number of *p's* and number of *q's* in *w* respectively.

Comment

Step 2 of 2

The language belonging to the shuffle of *P* and *Q* can be given as:

$$S = \left\{ w \in \{0,1,p,q\}^* \mid n_0(w) = n_1(w) \text{and } n_p(w) = n_q(w) \right\}$$

The language shown above for the shuffle of *P* and *Q* is not a context free language and it can be proved by using the proof by contradiction.

Therefore, suppose that the language defined for the shuffle of P and Q is context free and let *t* be the length of its pumping lemma.

• Now, also suppose that $ab^2de^2f \neq S$ and by using the approach of pumping lemma shuffle can be represented or written as *abdef* where $|ae| > 1$ and $|\text{abd}| \leq r$. Now, $|\text{abd}|$ cannot contain both zeros and ones and both *p's* and *q's* as $|\text{abd}| \leq r$.

• Thus, if the principle of pumping lemma is applied to ab^2de^2f once then the resultant string that will be obtained will have unequal number of zeros and ones or unbalanced ratio of *p's* and *q's*.

• This implies that $ab^2de^2f \neq S$ and thus this also signifies that the language S belonging to the shuffle of *P* and *Q* is not a context free language or CFL.

Now, the two languages *P* and *Q* which are under consideration are context free but the shuffle of these two languages is not context free.

Thus, it can be concluded and deduced that context free languages or CFL's are not closed under shuffle operation.

Comments (2)

Problem

Say that a language is ***prefix-closed*** if all prefixes of every string in the language are also in the language. Let C be an infinite, prefix-closed, context-free language. Show that C contains an infinite regular subset.

Step-by-step solution

Step 1 of 2

Given: A language which is prefix closed. Here, it is required to prove that an infinite regular subset is contained in every context free language which is prefix-closed.

Comment

Step 2 of 2

Proof: Consider a language L which is context free and prefix closed. As the language L is context free and therefore, the principle of pumping lemma can be applied on it.

Let the length of the pumping lemma be P and the string that needs to be considered in the language is s.

The length of the pumping lemma P is lesser or smaller than length of the string. Now, it is possible to break the string into 'abcde' such that $ab^kcd^ke \in L$ for all $k \geq 0$ and $|bd| \geq 1$.

Now, it is already given that the language L is prefix closed and all the prefixes of the string s are also present in language L, therefore, $ab^k \in L$ for all $k \geq 0$.

Thus, it can be inferred that the language formed from the ab^* is regular and it is a subset of L.

Also, if $b \neq \in$ then it implies ab^* is an infinite regular subset of Land thus it proves the required statement.

Comments (2)

Problem

Read the definitions of NOPREFIX(A) and NOEXTEND(A) in Problem 1.40.

a. Show that the class of CFLs is not closed under NOPREFIX.

b. Show that the class of CFLs is not closed under NOEXTEND.

Problem 1.40

Recall that string x is a ***prefix*** of string y if a string z exists where xz = y, and that x is a ***proper prefix*** of y if in addition x ≠ y. In each of the following parts, we define an operation on a language A. Show that the class of regular languages is closed under that operation.

A**a.** NOPREFIX(A) = {w ? Al no proper prefix of w is a member of A}.

b. NOEXTEND(A) = {w ? Alw is not the proper prefix of any string in A}.

Step-by-step solution

Step 1 of 3

a)

Consider the $NOPREFIX$ operation. For a language A, the $NOPREFIX$ operation is defined as:

$NOPREFIX(A) = \{w \in A \mid \text{no proper prefix of } w \text{ is a member of } A\}$

• Now consider the language P, defined as $P = P_1 \cup P_2$ where $P_1 = \{x^a y^b z \mid a \neq b, a, b \geq 1\}$ and $P_2 = \{x^a y^b z^b \mid a, b \geq 1\}$.

• If a string $x^a y^b z$ of P_1 is considered, then the proper prefix of it is the string that consists only x and y and all the string in P_1 and P_2 consists minimum one z. Therefore, all strings in P_1 is in $NOPREFIX(P)$.

Comment

Step 2 of 3

Now, if a string $x^a y^b z^b$ in P_2 is considered. It is not in $NOPREFIX(P)$, if and only if there is proper prefix of it that is in P.

• As no proper prefix exists in P_2, the proper prefix will have to come from P_1 and hence the $a \neq b$. Thus, the string in P_2 which are in $NOPREFIX(P)$ are $\{x^a y^a z^a \mid a \geq 1\}$. Therefore, $NOPREFIX(P) = P_1 \cup \{x^a y^a z^a \mid a \geq 1\}$

• P is a context free language since P_1 and P_2 are both context-free and context-free languages are closed under union. However, $NOPREFIX(P)$ is not context-free.

• In other way, context-free behavior is shown by $NOPREFIX(P) \cap P(x^* y^* z z z^*) = \{x^a y^a z^a \mid a \geq 2\}$ **that is a contradiction**. Therefore, **a context-free language** P **exists in such a way that** $NOPREFIX(P)$ **is not context-free.**

Hence, from the above discussion, it can be said that **context-free languages are not closed under** $NOPREFIX$ **operation.**

Comment

Step 3 of 3

b)

Consider the $NOEXTEND$ operation. For a language A, the $NOEXTEND$ operation is defined as:

$NOEXTEND(P) = \{w \in A \mid w \text{ is not a proper prefix of any string in } A\}$

Now consider the language $P = P_1 \cup P_2$ where $P_1 = \{x^a y^b z^c \mid a \neq b, a, b, c \geq 1\}$ and $P_2 = \{x^a y^b z^b \mid a, b \geq 1\}$.

• Consider the string $x^a y^b z^c \in P_1$, the given string is not in $NOEXTEND(P)$ since $x^a y^b z^{c+1}$, which is an extension of the string is in P.

• Now, the string $x^a y^b z^b$ is considered. Now any extension of this string in P should belong to P_1. Hence this string will not exist in $NOEXTEND(P)$, if and only if an extension of it belongs to P_1 if $a \neq b$. Therefore**, the string of the form $x^a y^a z^a$ belongs to $NOEXTEND(P)$.**

• Hence, $NOEXTEND(A) = \{x^a y^a z^a \mid a \geq 1\}$. As it is known **that P is context-free but $NOEXTEND(P)$ is not context-free**.

Hence from the above explanation, it can be said that "**the context-free language are not closed under $NOEXTEND$ operation".**

Comment

Problem

Let

$$Y = \{w \mid w = t_1 \#t_2 \# \cdots \#t_k \text{ for } k \geq 0, \text{ each } t_i \in 1^*, \text{ and } t_i \neq t_j \text{ whenever } i \neq j\}.$$

Here ∑ = {1, #}. Prove that Y is not context free.

Step-by-step solution

Step 1 of 6

CFG:

• A fixed set of grammar rules is known as **CFG (context free grammar)**. It consisting of that is augment (N, T, P, S).

• Where, **N** is set of non-terminal symbol.

• **T** is set of terminal $N \bigcap T = NULL$.

• **P** is set of rule, $P : N \to (N \bigcup T)^*$.

• **S** is start symbol.

Comment

Step 2 of 6

Consider the following details:

The language is $Y = \{w \mid w = t_1 \#t_2 \#....\#t_k \text{ where } k \geq 0, t_i \in 1^* \text{and } t_i \neq t_j \text{ when } i \neq j\}$ with the terminals being $\Sigma = \{1, \#\}$.

Comment

Step 3 of 6

Proof:

Theorem 2.34: Any string s in A, the pumping lemma p is the minimum length such that It could make part under five ends $s = uvxyz$. The string s also satisfies the following conditions for a context-free language A:

1. The string what's to come for $uv^i xy^i z$ only those context-free dialect A, the point when every $i \geq 0$.

$uv^i xy^i z \in A$

2. Those strings that need aid pumped, v furthermore y, can't both make the void string ε.

$|vy| > 0$

3. The joined together period of the strings lying inside what's to come for u Also z must not a chance to be more stupendous that those pumping length p.

$|vxy| \leq p$

Comment

Step 4 of 6

This problem is solved by the proof of contradiction.

• The language Y is supposed to be a context-free language.

• Theorem 2.34 is shown not to hold for the language.

• This makes the assumption, which is that Y is a CFL, invalid.

• Assume language Y is a context-free language.

• Now it can be seen that either x or y cannot have any $\#$'s.

• This is as when user pump the string then user will get strings of the form $s = t_1 \#t_2 \#....\#t_k$ where $t_i = t_j$ when $i \neq j$.

• Such strings do not lie in A. Consequently, to get $v, y = 1^*$.

Comment

Step 5 of 6

Construction:

Consider the string $s = uv^i xy^i z$ with $v = 1^*, y = 1^*$. The two cases possible for the substring vxy are:

• The substring contains $\#$: as the $\#$ symbol cannot lie in either v or y, it must lie in x.

• When the string $s = uv^i xy^i z$ is pumped with $x = 1^* \#1^*$, the case $t_i = t_j$ when $i \neq j$ can occur.

• Thus, pumping this string does not necessarily produce strings that lie in Y.

• It does not contain $\#$: the substring xyz will be just be a sequence of 1s. As was argued for the previous case, on applying the condition 1 of theorem 2.34 to pump the string will result in strings wherein $t_i = t_j$ in cases when $i \neq j$. As has been seen these strings are not part of language Y.

Comments (1)

Step 6 of 6

Conclusion:

The language Y does not satisfy the pumping lemma. Consequently, it is not a CFL.

Comment

Problem

Next

For strings w and t, write $w \triangleq t$ if t and w have the same symbols in the same quantities, but possibly in a different order.

For any string w, define $SCRAMBLE(w) = \{t \mid t \triangleq w\}$. For any language A, let $SCRAMBLE(A) = \{t \mid t \in SCRAMBLE(w) \text{ for some } w \in A\}$.

a. Show that if ∑ = {0,1}, then the *SCRAMBLE* of a regular language is context free.

b. What happens in part (a) if ∑ contains three or more symbols? Prove your answer.

Step-by-step solution

Step 1 of 4

For any two strings *w* and *t*, *SCRAMBLE* is defined as, $SCRAMBLE(w) = \{t \mid t \square w\}$. Consider a language A, $SCRAMBLE(A) = \{t \mid t \in SCRAMBLE(w) \text{ for some } w \in A\}$.

Comment

Step 2 of 4

a.

Consider the input alphabet $\Sigma = \{0,1\}$. Assume the language $L = (10)^*$ over the input alphabet $\Sigma = \{0,1\}$. The *SCRAMBLE*(*w*) can have all the word permutations of *w*. The states in FA for the language *L* must be rearranged to accept the *SCRAMBLE* of the language.

The intersection of *SCRAMBLE*(*L*) and a regular language need to be regular if *SCRAMBLE*(*L*) is regular. Consider a regular language $M = 1^*0^*$. The intersection of *SCRAMBLE*(*L*) and *M* is, $SCRAMBLE(L) \cap 1^*0^* = \{1^n 0^n : n \geq 0\}$ which is context free language. Thus, the *SCRAMBLE*(*L*) cannot be a regular language.

Comment

Step 3 of 4

Now, using push down automata *P* to show that scramble $SCRAMBLE(A)$ of regular language *A* is context free. The Push down automata *P* contains set of states similar to DFA *D* which accepts *A*. Consider the input $w = w_1 w_2 w_3 \cdots w_n$, the push down automata *P* non-deterministically works to guess the permutation u of w in A regular language. When it guesses $w_1 = u_1$ then transition occurs by reading the input w_1 like *D*. When it guesses $w_1 \neq u_1$ then null transition occurs. The input symbol u_1 stored in stack for the future use.

When number of elements in the stack is greater than 1 then the machine works as follows:

If the input symbol is same as the top element of the stack, then it consumes both the symbol but does not make any transition. If the input symbol does not matches with the top element of the stack then it guesses that may be it is the next right element in *u* or guesses that current input symbol is wrong and make null transition while push the complement of the symbol in the stack. The symbol that is inserted in the stack will pay back as soon as possible. For language *A*, if *w* is a permutation for string *u* then there must be a path that lead to the accept state with empty state. This will be the acceptance conditions for PDA *P*.

Therefore, the SCRAMBLE of a regular language is context free.

Comment

Step 4 of 4

b.

Consider the input alphabet $\Sigma = \{0,1,2\}$. Assume the language $L = (210)^*$ over the input alphabet $\Sigma = \{0,1,2\}$.

The intersection of *SCRAMBLE*(*L*) and a regular language need to be regular if *SCRAMBLE*(*L*) is regular. Consider a regular language $M = 2^*1^*0^*$.

The intersection of *SCRAMBLE*(*L*) and *M* is, $SCRAMBLE(L) \cap 2^*1^*0^* = \{2^n1^n0^n : n \geq 0\}$ which is not a context free language.

The pushdown automata has a single stack which can be used to compare the occurrences of two symbols, but for 3 or more symbols PDA cannot be drawn.

Therefore, the SCRAMBLE of a regular language with 3 or more symbols is not context free.

Next

Comment

Problem

If A and B are languages, define $A \diamond B = \{xy \mid x \in A \text{ and } y \in B \text{ and } |x| = |y|\}.$

Step-by-step solution

Step 1 of 3

Consider the two regular languages *A* and *B* over the input alphabet Σ. The language $A \diamond B$ is defined as, $A \diamond B = \{xy \mid x \in A \text{ and } y \in B \text{ and } |x| = |y|\}$. For the language $A \diamond B$, if PDA is constructed then it can be said that $A \diamond B$ is in CFL.

Comment

Step 2 of 3

Consider the DFA $D_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ and $D_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ for the languages *A* and *B* respectively. The strings of the language $A \diamond B$ are formed by concatenating equal length strings from *A* and *B*. Construct the PDA $M = (Q, \Sigma, \Gamma, \delta, q_{start}, F)$ for the language $A \diamond B$ as,

• From the start state q_{start}, push the symbol $ into the stack to know the bottom of the stack.

• For every symbol from the language *A*, push 1 into the stack. It guesses the end of the string that belongs to *A* when it reaches to the final state of D_A.

• Then, for every symbol from the language *B*, pop 1 from the stack. When the string reaches the final state of D_B, it moves to the final state *F* if and only if the top of the stack is $.

The PDA can be constructed informally as above described.

Comment

Step 3 of 3

The formal description of the PDA $M = (Q, \Sigma, \Gamma, \delta, q_{start}, F)$ is as follows:

- $Q = Q_A \cup Q_B \cup \{q_{start}, F\}$
- Σ is the input alphabet for *A* and *B*
- $\Gamma = \{\$, 1\}$ where $ is the symbol used to know the bottom of the stack and 1 is pushed every time into the stack when the symbol read from the language *A*.

• The transition function is defined as,

$$\delta(q_{start}, \varepsilon, \varepsilon) = \{q_A, \$\}$$
$$\delta(q, a, \varepsilon) = \{\delta_A(q, a), 1\} \quad \text{if } q \in Q_A, \ a \in \Sigma$$
$$\delta(q, \varepsilon, \varepsilon) = \{q_B, \varepsilon\} \quad \text{if } q \in F_A$$
$$\delta(q, a, 1) = \{\delta_B(q, a), \varepsilon\} \quad \text{if } q \in Q_B, \ a \in \Sigma$$
$$\delta(q, \varepsilon, \$) = \{F, \varepsilon\} \quad \text{if } q \in F_B$$

Any other transitions apart from this will not be accepted.

- q_{start} is the start state.
- *F* is the final state.

The PDA non deterministically guesses the end of the string from D_A and transitions to the start symbol of D_B if it is in a final state of D_A. The PDA *M* accepts the string when it is in accepting state of D_B while hitting the empty stack. This shows that $L(M) = A \diamond B$.

Therefore, for any two regular languages *A* and *B*, $A \diamond B$ is in CFL.

Problem

Let $A = \{wtw^{\mathcal{R}} \mid w, t \in \{0,1\}^* \text{ and } |w| = |t|\}.$ Prove that A is not a CFL.

Step-by-step solution

Step 1 of 2

In the given function $A = \left\{ wtw^R \middle| w, t \in \{0,1\}^* \text{ and } |w| = |t| \right\}$, every string $s = wtw^R \in A$ where $|w| = |t| = |w^R|$ and $|s|$ is a multiple of three. Let us assume that *A* is context free and reach to a contradiction.

• Let *p* be the pumping length for A that is guaranteed to exit by pumping lemma.

• Select string $s = 0^{2p}0^p1^p0^{2p} \in A$ with $|s| > p$.

• Therefore, there exits $uvxyz$ such that

1) $uv^ixy^iz \in B$ for all $i \geq 0,$

2) $|uy| > 0$,

3) $|vxy| \leq p$.

• Consider these cases for pumping lemma:

Case 1: $|vy|$ is not a multiple of 3. Then $s' = uv^2xy^2z \notin A$ since $|s'|$ is no longer a multiple of 3.

Case 2: vxy consist of only 0s from the prefix set of 0s and $|vy| = 3r$ for some *r*. Then, $uv^2xy^2z = 0^{3p+3r}1^p0^{2p} = 0^{2p+r}0^{p+2r}1^{p-r}1^r0^{2p} \notin A$, since $w = 0^{2p+r}$ and $w^R \neq 1^r0^{2p}$, the w^R of the string s.

Case 3: uxy consists of only 1s and $|vy| = 3r$ for some *r*. Then, the string $uv^2xy^2z = 0^{3p}1^{p+3r}0^{2p} = 0^{2p+r}0^{p-r}1^{p+2r}1^r0^{2p} \notin A$, since $w = 0^{2p+r}$ and $w^R \neq 1^r0^{2p}$, the w^R of the string s.

Case 4: vxy consist of only 0s from the suffix set of 0s and $|vy| = 3r$ from some *r*. Then $uv^0xy^0z = 0^{3p}1^p0^{2p-2r} \notin A,$ since $w = 0^{2p-r}$ and $w^R \neq 1^{2r}0^{2p-3r}$, the w^R of the string s.

.

Case 5: $uy = 0^m1^n$ with $m, n > 0$ and $m + n = 3r$ from some *r*. Then, the string $uv^2xy^2z = 0^{3p+m}1^{p+n}0^{2p} = 0^{2p+r}0^{p+m-r}1^{p+n-r}1^r0^{2p} \notin A,$ since $w = 0^{2p+r}$ and $w^R \neq 1^r0^{2p}$, the w^R of the string s.

Case 6: $vy = 1^m0^n$ with $m, n > 0$ and $m + n = 3r$ for some *r*.

• **Sub-case 6.1**: $n < r$. Then $uv^2xy^2z = 0^{3p}1^{p+m}0^{2p+n} = 0^{2p+r}0^{p-r}1^{p+m+n-r}1^{r-n}0^{2p+n} \notin A$, since $w = 0^{2p+r}$ and $w^R \neq 1^{r-n}0^{2p+n}$, the w^R of the string s.

• **Sub-case 6.2**: $n > r$. Then $uv^0xy^0z = 0^{3p}1^{p-m}0^{2p-n} = 0^{2p-r}0^{p+r}1^{p+r-m-n}1^{n-r}0^{2p-n} \notin A$, since $w = 0^{2p-r}$ and $w^R \neq 1^{n-r}0^{2p-n}$, the w^R of the string s.

• **Sub-case 6.3**: $n = r$. Then $uv^{p+2}xy^{p+2}z = 0^{3p}1^{p+2r(p+2-1)}0^{2p+r(p+2-1)n} = 0^{3p}1^{rp+r-p}1^{2r+rp+r}0^{2p+rp+r} \notin A$, since $w = 0^{2p+r}$ and $w \neq 0^{3p}1^{rp+r-p}$ and $w^R \neq 0^{2p+rp+r}$, the w^R of the string s.

If $i < p + 2,$ then take $r = 1,$ $uv^ixy^iz = 0^{3p}1^{p+2(i-1)}0^{2p+(i-1)}$, $= 0^{2p+(i-1)}0^{p-(i-1)}1^{p+2(i-1)}0^{2p+(i-1)} \in A$. As there are not enough 1's to push or pump into *w,* $p + 2$ is the first time that is guaranteed.

Comment

Step 2 of 2

Thus, in all the cases, the 1) of pumping lemma results in a contradiction. Therefore, the assumption that *A* is context free language, is false.

Comment

Problem

Consider the following CFG G:

$$S \rightarrow SS \mid T$$
$$T \rightarrow \mathtt{a}T\mathtt{b} \mid \mathtt{ab}$$

Describe L(G) and show that G is ambiguous. Give an unambiguous grammar H where L(H) = L(G) and sketch a proof that H is unambiguous.

Step-by-step solution

Step 1 of 10

CFG:

• A Context Free Grammar (CFG) is an arrangement of recursive rewriting principles (or productions) used to create pattern of strings.

• A CFG comprises of the following components: An arrangement of terminal symbols.

• Which are the characters of the letter set that show up in the strings created by the grammar.

Comment

Step 2 of 10

Ambiguous Grammar:

• This is a context free grammar to which there exists a string that can have in excess of left most derivation or parse tree.

Comment

Step 3 of 10

The language of a CFG G has to be described and shown to be ambiguous.

An unambiguous grammar H has to obtain from the grammar.

The rules for the CFG G are:

• First, consider the strings produced by the variable T. Let the CFG $I = (V, \Sigma, R, T)$ be:

$T \rightarrow aTb \mid ab$

• As it is either a string of two terminals ab or it is placed between the same two terminals aTb.

• Therefore, the language generated will consist of a sequence of two or more a's followed by the same number of b's.

$L(I) = \{a^i b^i \mid i > 1\}$

• All the strings lying in this language will be of even length as $\left|a^i b^i\right| = i + i = 2i$.

Comment

Step 4 of 10

• The start symbol S either can be replaced by two start symbols SS or by the variable T.

• It can be seen that a string in the language $L(G)$ will be the concatenation of one or more occurrences of strings of $L(I)$. So, the language $L(G)$ will be given by:

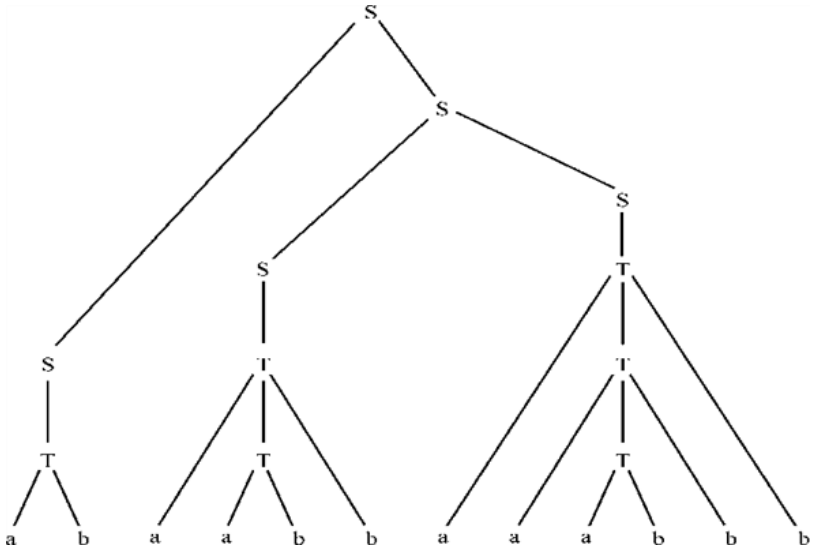
$$L(G) = \{a^{i_1}b^{i_1}a^{i_2}b^{i_2}....a^{i_k}b^{i_k} \mid i_1, i_2, ..., i_k \geq 1 \text{ and } k \geq 1\}$$

• Take the string $s = abaabbaaabbb$.

• A derivation for this string s is:

$$S \Rightarrow SS \Rightarrow TS \Rightarrow abS \Rightarrow abSS \Rightarrow abTS \Rightarrow abaTbS \Rightarrow abaabbS$$
$$\Rightarrow abaabbT \Rightarrow abaabbaTb \Rightarrow abaabbaaTbb \Rightarrow abaabbaaabbb$$

• This leads to parse tree:



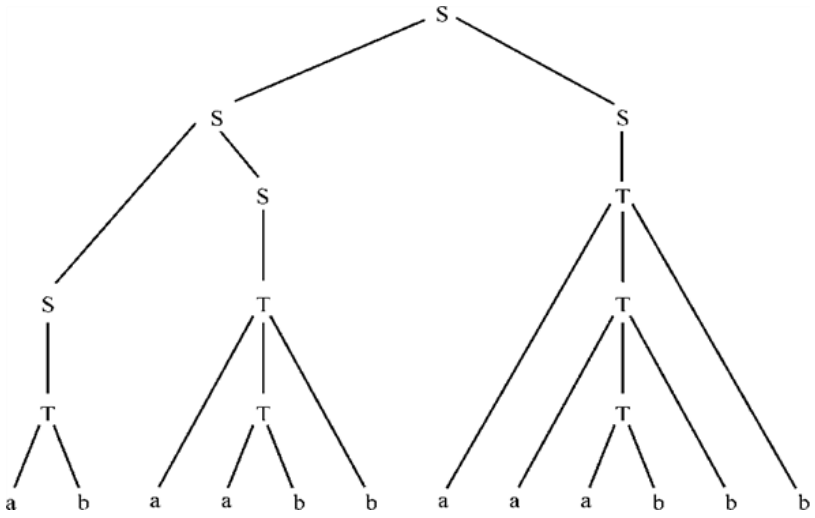
Comment

Step 5 of 10

The Second alternative derivation for this string s is:

$$S \Rightarrow SS \Rightarrow ST \Rightarrow SaTb \Rightarrow SaaTbb \Rightarrow Saaabbb \Rightarrow SSaaabbb \Rightarrow STaaabbb$$
$$\Rightarrow SaTbaaabbb \Rightarrow Saabbaaabbb \Rightarrow Taabbaaabbb \Rightarrow abaabbaaabbb$$

• This leads to parse tree:



• The string $s = abaabbaaabbb$ can be derived into two different parse trees. So, the grammar G is ambiguous. The rules for the start symbol S are:

$$S \rightarrow SS \mid T$$

• Due to this rule, it is possible to get different derivations for a string in the language $L(G)$.

Comment

Step 6 of 10

Unambiguous Grammar:

• This is also a context free grammar to which each substantial string has a one of a kind unique left most derivation or parse tree.

• The grammar G can be converted into unambiguous grammar $H = (V, \Sigma, R', S)$ by removing this ambiguity. The rules are modified to:

$S \to TS \mid T$
$T \to aTb \mid ab$

• It has to be checked if $L(G) = L(H)$.

• That is a string s lies in $L(G)$ if and only if it lies in $L(H)$. This result is proven via induction on the length of s.

•

Comment

Step 7 of 10

It has to be proven in both the directions – the 'if' direction and the 'only if' direction.

Comment

Step 8 of 10

Now, it can be proved that a string s lies in $L(G)$ if it lies in $L(H)$.

Basis:

• The string ab lies in $L(H)$.

• It also be derived in $L(G)$, that is $S \Rightarrow_G T \Rightarrow_G ab$

Inductive step:

• All strings of length at most n of $L(H)$ lie in the language $L(G)$. Consider an arbitrary w string of length $n+2$ lying in the language $L(H)$.

• There are three cases possible:

1. The string w starts with an ab. It can be derived in the language $L(G)$ to a string of length n as follows.

$S \Rightarrow_G SS \Rightarrow_G TS \Rightarrow_G abS \Rightarrow_G abx$

2. An ab in the middle of the string w. The derivation for this string is.

$S \Rightarrow_G SS \Rightarrow_G SSS \Rightarrow_G STS \Rightarrow_G SaTbS \Rightarrow_G xaybz$

3. The string w ends with an ab.

$S \Rightarrow_G SS \Rightarrow_G ST \Rightarrow_G Sab \Rightarrow_G xab$

• In all the above cases, the string can be derived in grammar G to a string(s) whose length $\leq n$. So, a string s lies in $L(G)$ if it lies in $L(H)$.

Comment

Step 9 of 10

Only-If:

• It can be proved that a string s lies in $L(H)$ if it lies in $L(G)$.

Basis:

• The string ab lies in $L(G)$. It also be derived in $L(H)$, that is $S \Rightarrow_H T \Rightarrow_H ab$.

Inductive step:

• All strings whose length is equal to or less than n of $L(G)$ lie in the language $L(H)$

Comment

Step 10 of 10

Consider an arbitrary w string of length $n+2$ lying in the language $L(G)$.

• The three cases possible are:

• The string w starts with an ab. It can be derived in the language $L(H)$ to a string of length n as follows.

$$S \Rightarrow_H SS \Rightarrow_H TS \Rightarrow_H abS \Rightarrow_H abx$$

• An ab in the middle of the string w. The derivation for this string is.

$$S \Rightarrow_H SS \Rightarrow_H SSS \Rightarrow_H STS \Rightarrow_H SaTbS \Rightarrow_H xaybz$$

• The string w ends with an ab.

$$S \Rightarrow_H SS \Rightarrow_H ST \Rightarrow_H Sab \Rightarrow_H xab$$

The string can be derived in all the cases into a string(s) whose length is at most.

• Therefore, it has been proven if a string lies in $L(H)$ then it also lies in $L(G)$.

• An outline of a proof showing that H **is unambiguous is to be given**.

• A grammar is unambiguous if there is only one leftmost derivation of a string in the grammar.

• An induction on the length of the string can be used to prove this and showing that is only one way to derive an arbitrary string into a string of smaller length.

Hence, it has been proven that the languages $L(H)$ and $L(G)$ are equivalent. In other words:

$$L(G) = L(H)$$

Comment

Problem

Let ∑ = {0,1} and let B be the collection of strings that contain at least one 1 in their second half. In other words,

$$B = \{uv |\ u \in \Sigma^*, v \in \Sigma^* 1 \Sigma^* \text{ and } |u| \geq |v|\}.$$

a. Give a PDA that recognizes B.

b. Give a CFG that generates B.

Step-by-step solution

Step 1 of 5

Consider the following information:

The alphabet set $\Sigma = \{0,1\}$.

$B = \{uv \mid u \in \Sigma^*, v \in \Sigma^* 1 \Sigma^* \ and \ |u| \geq |v|\}$.

Comment

Step 2 of 5

a.

Construction of PDA:

1. The PDA for this language works by first reading and pushing the string u onto the stack. The string u consists of either 0's or 1's.

2. After pushing the string *u* into stack is completed, the machine has all the letters of *u* on the stack. Now, match the letters in *u* with the letters in *v*.

3. Remember that the length of *u* is greater than or equal to *v*.

4. If the condition that number of 1's in the input is at least 1 in second half, then accept the entire string.

Problem:

• But here a problem takes place in finding the middle of the string *uv*.

• That is the position at which *u* ends and *v* starts.

• The problem can be solved by using **Non deterministic push down automata**.

• The PDA will be designed such that the empty string $\$$ can be accepted by the PDA.

Construction:

The PDA M that recognizes B is $(Q, \Sigma \frac{1}{2}, \Gamma, \delta, q_0, F)$, where:

$Q = \{q_0, q_1, q_2, q_3\}$,
$\Sigma = \{0,1\}$,
$\Gamma = \{x\}$,

$F = \{q_3\}$

Transition function δ of the represented in a tabular format:

Input :	0		1		ε	
Stack :	x	ε	x	ε	x	ε
q_0	$\{(q_0, xx)\}$	$\{(q_0, x)\}$	$\{(q_0, xx)\}$	$\{(q_0, x)\}$	$\{(q_1, x)\}$	
q_1	$\{(q_1, \varepsilon)\}$		$\{(q_2, \varepsilon)\}$			
q_2	$\{(q_2, \varepsilon)\}$		$\{(q_2, \varepsilon)\}$			

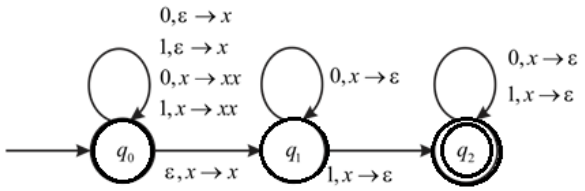
Comment

Step 3 of 5

The state diagram for the PDA M is given below:

Comment

Step 4 of 5



Comments (6)

Step 5 of 5

b.

The CFG for the language *B* is as follows:

$S \to UV$

$U \to AB$

$V \to A1A \mid A1B \mid AIU \mid B1U \mid U1U$

$A \to 00^* \mid \in$

$B \to 11^* \mid \in$

Explanation:

• The string *S* is concatenation of *U* and *V*.

• The string U may consist of any number of 0's and 1's.

• The string *V* may consist of atleast one 1 or only one 1.

• The string *A* may consists of atleast one zero or more than one zeros.

• The string *B* may consist of atleast one 1 or more than one 1's.

Comments (3)

Problem

Let ∑ = {0,1}. Let C_1 be the language of all strings that contain a 1 in their middle third. Let C_2 be the language of all strings that contain two 1s in their middle third.

$$\text{So } C_1 = \{xyz |\ x, z \in \Sigma^* \text{ and } y \in \Sigma^* 1 \Sigma^*, \text{where } |x| = |z| \geq |y|\}$$

$$\text{and } C_2 = \{xyz |\ x, z \in \Sigma^* \text{ and } y \in \Sigma^* 1 \Sigma^* 1 \Sigma^*, \text{where } |x| = |z| \geq |y|\}.$$

a. Show that C_1 is a CFL.

b. Show that C_2 is not a CFL.

Step-by-step solution

Step 1 of 7

A language L is said to be **context-free** if there exist some integer $q \geq 1$ (it is also known as pumping length) in such a way that all the string S in L which is equal or longer than q symbols or $|S| \geq q$.

It can be written as $S = abcde$ with substring $a, b, c, d \text{ and } e$ such that

1. $|bcd| \leq q$
2. $|bd| \geq 1$, and
3. $ab^x cd^x e$ is in L for all $x \geq 0$.

Comment

Step 2 of 7

a.

Consider the language C_1 which is given below:

$C_1 = \{xyz \mid x, z \in \Sigma^* \text{ and } y \in \Sigma^* 1 \Sigma^*, \text{where} |x| = |z| \geq y\}$, where $\Sigma = \{0,1\}$

Now, a Push down automata needs to construct which help in determining the language C_1. The PDA M that recognizes C_1 is $(Q, \Sigma \frac{1}{3}, \Gamma, \delta, q_0, F)$, where:

$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0,1\}$
$\Gamma = \{x\}$
$F = \{q_2\}$

Comment

Step 3 of 7

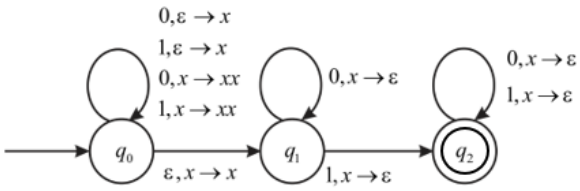
The transition functions δ of the represented in a tabular format:

Input :	0		1		ε	
Stack :	x	ε	x	ε	x	ε
q_0	$\{(q_0, xx)\}$	$\{(q_0, x)\}$	$\{(q_0, xx)\}$	$\{(q_0, x)\}$	$\{(q_1, x)\}$	
q_1	$\{(q_1, \varepsilon)\}$		$\{(q_2, \varepsilon)\}$			
q_2	$\{(q_2, \varepsilon)\}$		$\{(q_2, \varepsilon)\}$			

Comment

Step 4 of 7

The state diagram for the PDA M is given below:



Hence from the above explanation it can be said that there exists a Turing machine which accepts the given language.

Hence, C_1 is a context free language.

Comments (4)

Step 5 of 7

b.

Now consider the language C_2 that accepts all the string that contains two 1's in their middle of the string. The language C_2 can be defined as:

$$C_2 = \{xyz \mid x,z \in \Sigma^* \text{ and } y \in \Sigma^* 1 \Sigma^* 1 \Sigma^*, \text{where} \mid x \mid = \mid z \mid \geq y\}$$

Now, using a pumping lemma to show Language C_2 is not CFL.

Let as assume that C_2 is CFL and obtain a contradiction.

Let pumping length of the pumping lemma is *p*.

Let select a string $S = 0^{p+2}10^p10^{p+2}$ of given language.

Comment

Step 6 of 7

Let, divide S into five pieces $S = uvxyz$, it must satisfy the conditions according to the pumping lemma,

1. For each $i \geq 0, uv^ixy^iz \in C_2$,
2. $\mid vy \mid > 0$, and
3. $\mid vxy \mid \leq p$

Where

$u = 0^{p+2}$

$v = 1$

$x = 0^p$

$y = 1$

$z = 0^{p+2}$

Comments (1)

Step 7 of 7

Let $i = 1$

After pumping, string becomes $S = 0^{p+2}10^p10^{p+2}$.

According to the pumping lemma third condition, new string $\mid 10^p1 \mid \leq p$ becomes fails, that means, length of the *vxz* is greater than the pumping length *p*, i.e. $\mid vxy \mid \leq p$.

Hence, C_2 is not context free language.

Problem

We defined the rotational closure of language A to be *RC*(A) = {yxl xy ? A}. Show that the class of CFLs is closed under rotational closure.

Step-by-step solution

Step 1 of 1

CFLs is closed under rotational closure

The rotational closure of a language A is defined as $RC(A) = \{yx \mid xy \in A\}$.

The class of CFLs is closed under the concatenation operation. In other words if L_1 and L_2 are CFLs then the language $L = L_1 \bullet L_2$ is also a CFL. The converse is also true: if language $L = L_1 \bullet L_2$ is a CFL, then language L_1 and language L_2 are CFLs.

Consider a string $s = xy$ in the language A. It can be formed from the concatenation of two languages X and Y, such that $x \in X$ and $y \in Y$. As the language $A = X \bullet Y$ is a CFL, the languages X and Y will also be CFLs.

The rotational closure of the string $s = xy$ will be $RC(s) = RC(xy) = yx$. It can be formed by concatenating Y and X.

$RC(s) = Y \bullet X$

As both Y and X are context-free languages, the language $RC(s)$ is a context-free language for any string $s \in A$.

The rotational closure has been proven for the class of CFLs.

Comments (1)

Problem

We defined the *CUT* of language A to be *CUT*(A) = {yxzl xyz ? A}. Show that the class of CFLs is not closed under *CUT*.

Step-by-step solution

Step 1 of 4

A CFL language is closed under few operations. If L_1 and L_2 are two CFL languages then the following language will also be context free:

1. Cyclic shift of CFL
2. Union of both is CFL
3. The reversal of L_1
4. Concatenation of both languages
5. Kleen star of L_1
6. Image of L_1 under homomorphism
7. Image inverse of L_1 under inverse homomorphism

Comment

Step 2 of 4

Consider the language $CUT(A)$ is defined as $CUT(A) = \{yxz \mid xyz \in A\}$.

Consider $L_1 = x,\ L_2 = y,\ L_3 = z$.

If the CFL $CUT(A)$ is closed under above operation then all conditions must be true. For showing it is not closed it will be sufficient to prove that one condition does not hold.

Comment

Step 3 of 4

Cyclic shift of CFL:

Consider $L_1 \circ L_2$ exist in CFL, then it there must be Cyclic shift exist in the definition of language that is $L_2 \circ L_1$ must exist.

Consider a string $s = xyz$ in the language $CUT(A)$. It can be formed from the concatenation of the languages X and Y and Z, such that $x \in X$ and $y \in Y$ and $z \in Z$.

As per the definition of language $CUT(A)$, if $yxz \in A \text{ then } xyz \in A$. For given definition it will be closed if cyclic shift is also CFL. That is, if $xy \in A \text{ then } yx \in A$. But given condition is not true as definition of $CUT(A)$.

Comments (1)

Step 4 of 4

The Cyclic shift of CFL which is first condition does not follow. Hence, given CFL cannot be closed under $CUT(A) = \{yxz \mid xyz \in A\}$.

Comment

Problem

Show that every DCFG is an unambiguous CFG.

Step-by-step solution

Step 1 of 1

An **ambiguous grammar** is defined as "**a context free grammar** for which there subsists a string and that string may contains greater than one left-most derivation. **An unambiguous grammar** is defined as "a **context free grammar** for which all **justifiable string** has an individual leftmost derivation.

• As context free grammar (CFG's) is proper superset of deterministic context-free grammars (DCFG's). It can be derived from deterministic finite automata and it can be used to generate deterministic context free language.

• DCFG's (deterministic context-free grammars) are always shows an unambiguous behavior and an unambiguous context free grammar (CFG's) is an important super class of DCFG's.

The above statement can be proved by the following way:

• As it is known that **for every pushed down automata** M **there exist an equivalent context free grammar** G**.**

Therefore, M **Recognizes** $L\$ \Rightarrow G$ **generates** $L\$$. But, M **deterministic** $\Rightarrow G$ **is an unambiguous**. Hence, **replacing** $\$$ **by** ε **in** $G \Rightarrow G$ **generates** L.

• Thus**,** from the above explanation it can be said that **every DCFG is an unambiguous CFG.**

Comment

Problem

Show that every DCFG generates a prefix-free language.

Step-by-step solution

Step 1 of 1

DCFG stands for deterministic context free grammars. It is subset of context-free grammars. DCFGs are derived from the deterministic pushdown automata (variation of pushdown automata) for generating deterministic context free languages.

Prefix-free language refers to language in which any of its members does not have any prefix. By contradiction, it can be proved that DCFG always generates prefix-free language. It is given below:

Consider there are two strings w and wz in $L(G)$, where w and wz are unequal strings, $L(G)$ is language of grammar and G is DCFG. As both strings are valid, handles of them will exist. Both can be written as:

$w = xhy$ and

$$\begin{aligned} wz &= xhyz \\ &= xh\hat{y} \end{aligned}$$

'h' refers to handle of w.

Now consider u and uz valid strings as first reduced step of w and wz. The process is continued till S_1 and S_1z, where S_1 refers to start variable. Since S_1 never appears at the right side, so S_1z cannot be reduced. This leads to contradiction.

Comment

Problem

Show that the class of DCFLs is not closed under the following operations:

a. Union

b. Intersection

c. Concatenation

d. Star

e. Reversal

Step-by-step solution

Step 1 of 5

a) Suppose $L_1 = \{a^i b^j c^k : i, j, k \geq 0 \text{ and } i \neq j\}$ and $L_2 = \{a^i b^j c^k : i, j, k \geq 0 \text{ and } i \neq k\}$ are both deterministic context free languages **(DCFL's).** However, $L_1 \cup L_2$ which is defined as $L_1 \cup L_2 = \{a^i b^j c^k : i, j, k \geq 0 \text{ and } (i \neq k \text{ or } i \neq j)\}$ is not comes under DCFL's which can be seen as follows.

• Suppose $L_1 \cup L_2$ is a DCFL. T hen its complement is also a DCFL as discussed in theorem.
$(L_1 \cup L_2)' \cap \{a\}^* \{b\}^* \{c\}^* = \{a^i b^j c^k : i, j, k \geq 0 \text{ and } (i = k = j)\}$ is a (D)CFL. This shows a **contradiction**.

• Hence, it can be said that" **Deterministic Context Free Language's (DCFL's) is not closed under union"**.

Comments (1)

Step 2 of 5

b) Non-closure under intersection follows from the non-closure under union (as discussed above) and closure under complement: $K \cup L = (K' \cap L')'$.

• Thus, if the family of DCFLs would be closed under intersection, it follows with the help of closure under complement that it would also be closed under union. This shows a **contradiction.**

• Hence, it can be said that" **Deterministic Context Free Language's (DCFL's) is not closed under intersection"**.

Comment

Step 3 of 5

c) Suppose $L_1 = \{a^i b^j c^k : i, j, k \geq 0 \text{ and } i \neq j\}$ and $L_2 = \{a^i b^j c^k : i, j, k \geq 0 \text{ and } i \neq k\}$ be the languages. Now, suppose $L_3 = \{d\} L_1 \cup L_2$,where d is the symbol different from a, b, c. It is simple to see that $L_3 \text{ and } \{d\}^*$ are **DCFL.** Now, it has intended to show that $\{d\}^* L_3$ is not a DCFL.

• Consider $\{d\}^* L_3 \cap \{d\}\{a\}^* \{b\}^* \{c\}^* = \{d\} L_1 \cup \{d\} L_2$. If $\{d\}^* L_3$ would be in DCFL then also $\{d\} L_1 \cup \{d\} L_2 = \{d\}(L_1 \cup L_2)$ is a DCFL.

• Now, it is fairly easy to see that for all words w and all languages K that consist K are a DCFL whenever $\{w\} k$ is DCFL. Consequently, since $L_1 \cup L_2$ is not a DCFL, also $\{d\}(L_1 \cup L_2)$ is not a DCFL. This shows a **contradiction**.

• Hence, it can be said that" **Deterministic Context Free Language's (DCFL's) is not closed under concatenation"**.

Comment

Step 4 of 5

d) Suppose L_4 is defined as $L_4 = \{d\} \cup \{d\} L_1 \cup L_2$ where L_1 and L_2 is defined same as the above discussion.

• Consider $L_4^* \cap \{d\}^* \left(\{a\}^* \{b\}^* \{c\}^* - \{\wedge\}\right) = \{d\} L_1 \cup \{d\} L_2$. If L_4^* is DCFL then also $\{d\} L_1 \cup \{d\} L_2$ should be in DCFL, but here it is not followed in this case.

• Hence, it can be said that" **Deterministic Context Free Language's (DCFL's) is not closed under star (*)"**.

Comment

Step 5 of 5

e) Non-closure under difference follows from the closure under complement and the non-closure under intersection as discussed in part (b): $K \cap L = K - L^{'}$, **thus if the family of DCFL's would be closed under difference.**

• The above terms follows with the help of closure under complements that would also be closed under intersection. That shows a contradiction.

• Hence, it can be said that" **Deterministic Context Free Language's (DCFL's) is not closed under reversal"**.

Comment

Problem

Let G be the following grammar:

$$S \to T\dashv$$
$$T \to T\mathtt{a}T\mathtt{b} \mid T\mathtt{b}T\mathtt{a} \mid \varepsilon$$

a. Show that $L(G) = \{w\dashv \mid$ w contains equal numbers of a's and b's$\}$. Use a proof by induction on the length of w.

b. Use the *DK*-test to show that G is a DCFG.

c. Describe a DPDA that recognizes L(G).

Step-by-step solution

Step 1 of 7

Consider the following grammar:

$S \to T \dashv$
$T \to TaTb \mid TbTa \mid \varepsilon$

It is required to show that $L(G) = \{w \dashv \mid w\}$

Comment

Step 2 of 7

• Here, it can be prove by the induction that the string w contains an equal number of a 's and b 's. Induction can be done in two ways either from the left side of the induction or from the right side of the induction.

• In order to prove $L(G) = \{w \dashv \mid w\}$, first ε handles occurs in reduction as it is indicated by the underscores in the leftmost reduction of the string $(\ )(\ ) \dashv$.

Comment

Step 3 of 7

• It is being known that $T \to \varepsilon$ and thus $T \dashv \to \varepsilon$ and $S \to \varepsilon$.

Consider the string w contains $aaabbb$. Each and every string which is being generated by the grammar *G* is in *L*.

$aaabbb \rightarrowtail aaSbb \rightarrowtail aSb \rightarrowtail S \rightarrowtail T \dashv$
$w \in T \dashv \in S \in L(G)$

Similarly in the second case, $w \dashv \in T \in S$

• Hence, it is being proved that any language G contains equal number of a's and b's in string w and $w \dashv$

Comment

Step 4 of 7

Using the *DK*-test it is to be shown that the grammar belongs to *DCFG*.

DK test depend upon one simple and surprising fact. *DK* is used for accepting an input z only if these two conditions are met.

• z is the prefix of the valid string $v = zw$.

• z should be end with a handle of v.

Comment

Step 5 of 7

Consider the grammar $L(G)=\{w \dashv \mid w\}$ which contains an equal number of a's and b's.

• So, if the grammar contains an equal number of a's and b's then the grammar $a^n b^n$ becomes $a^n b^n$. In case of *DCFG* all the handles are being forced. So, if zw is a one of the valid string with the prefix z that will end for handling zw'.

• In case of *DCFG* all the handles are being forced. So, if zw is a one of the valid string with the prefix z that will end for handling zw'. For all this property *DK* must be capable of handling all the accept states. But, the condition is that outgoing path should not be there.

Comment

Step 6 of 7

• Here, the construction of *DK* test is done for concluding that *G* is deterministic if the property of accept state is satisfied. For the construction of the *DFA*, first *NFA* need to be constructed. After the construction it is found that no outgoing path is there from the start state.

• Here, $aaabbb$ is a valid string and it handles zw so after the reduction of the rule it becomes: $T \to z.w$. Grammar must contain the following rules:

$$S_1 \to z_1 S_2 w_1$$
$$S2 \to z_2 S_3 w_2$$
$$\vdots$$
$$S_l \to z_l T w_l$$
$$T \to zw$$

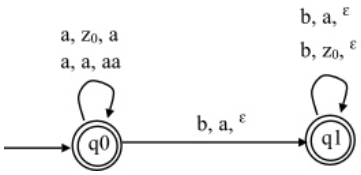
K contains the path from the start state to the end state. First the input $z = xu$ is being read after that transition is done and shifting of the string is done. After the shifting it is being found that there is no outgoing path from the start state.

Hence with the help of *DK* test it is proved that grammar belong to *DCFG*.

Comment

Step 7 of 7

The DPDA that recognizes the L(G) is as follows:



Comment

Problem

Let G_1 be the following grammar that we introduced in Example 2.45. Use the *DK*-test to show that G_1 is not a DCFG.

$$R \to S \mid T$$
$$S \to \mathtt{a}S\mathtt{b} \mid \mathtt{ab}$$
$$T \to \mathtt{a}T\mathtt{bb} \mid \mathtt{abb}$$

Step-by-step solution

Step 1 of 3

DK **-test is a procedure which is exactly used to determine whether a CFG is deterministic or not**. DK -test comes into picture because the definition of DCFG's does not give any accurate way to determine whether a CFG is deterministic. The procedure of DK-test can be explained as follow.

Starting with a CFG G , construct the associated DFA DK .Now, determine whether G is deterministic by testing $DK's$ accept states. The DK -test stipulates that every accept state contains:

1. **Exactly one completed rule, and**

2. **No dotted rule in which terminal symbol immediately follows the dot (that is, no dotted rule of the form** $B \to u.av \; for \; a \in \sum$).

Comment

Step 2 of 3

Now**, consider the grammar** G_1 which is given below:

$R \to S \mid T$
$S \to aSb \mid ab$
$T \to aTbb \mid abb$

• **The language of grammar** is $B \cup C$ where $B = \{a^m b^m \mid m \geq 1\} and \; C = \{a^m b^{2m} \mid m \geq 1\}$. In the leftmost reduction of string $aaabbb \in L(G_1)$. It is underlined the handle at each step:

$aa\underline{abb}b \to a\underline{aSb}b \to \underline{aSb} \to \underline{S \to R}$

• **Equivalently**, this is leftmost reduction of the string $aaabbbbbb$:

$aa\underline{abb}bbbb \to a\underline{aTbb}bb \to \underline{aTbb} \to \underline{T} \to R$.

• In the both of the cases which is given above, the leftmost reduction as shown happens to be **the only reduction possible**. In other grammars, where many reduction may occur.

• Here, it is notice that the handles of $aaabbb$ and $aaabbbbbb$ **are unequal**, even though the initial part of these strings agree.

Comment

Step 3 of 3

Therefore, from the above discussion **it violates the condition of** DK **-test (as** DK **-test say that every accept state must consist of exactly one completed rule**), because **so many completed rules present here**. **Thus,** it can be said that "**The grammar** G_1 **is not a DCFG".**

Comment

Problem

Let A = L(G_1) where G_1 is defined in Problem 2.55. Show that A is not a DCFL.

(Hint: Assume that A is a DCFL and consider its DPDA P. Modify P so that its input alphabet is {a, b, c}. When it first enters an accept state, it pretends that c's are b's in the input from that point on. What language would the modified P accept?)

Problem 2.55

Let G_1 be the following grammar that we introduced in Example 2.45. Use the *DK*-test to show that G_1 is not a DCFG.

$$R \rightarrow S \mid T$$
$$S \rightarrow \mathtt{a}S\mathtt{b} \mid \mathtt{ab}$$
$$T \rightarrow \mathtt{a}T\mathtt{bb} \mid \mathtt{abb}$$

Step-by-step solution

Step 1 of 3

On the contrary suppose *A* is a deterministic context free language. Consider *p* as the pumping length of *A*, such that *p* length string of *A* will satisfy the pumping lemma. Consider '*m*' as string of *A* with $m = 0^{2p}0^p1^p0^{2p}$. In order to satisfy the assumption, there are following ways so that '*w*' can be written as $uvxyz$, where $|vy| \geq 1$ and $|vxy| \leq p$, uv^ixy^iz string lies in *A* for any value of *i*.

Comment

Step 2 of 3

On the basis of above condition there are only 3 cases which are as follow:

Case 1:

In string uv^ixy^iz, vy only have 0s which are getting from the last 0^{2p} of *m*. Assume that '*i* 'is any number which satisfy the condition $7p > |vy| \times (i+1) \geq 6p$. Then, may be the length of uv^ixy^iz is not become the multiple of 3, or may be the string is in the form of wtw' specified that $|w| = |t| = |w'|$ having all zero's in w' and not all zero's in w $\left(\text{Means}, w^R \neq w'\right)$.

Case 2:

In string uv^ixy^iz, vy does not have 0s in last 0^{2p} of *m*. Then, may be the length of uv^2xy^2z is not become the multiple of 3, or may be the string is in the form of wtw' specified that $|w| = |t| = |w'|$ having all zero's in w and not all zero's in w' $\left(\text{Means}, w^R \neq w'\right)$.

Comment

Step 3 of 3

Case 3:

In string uv^ixy^iz, vy have some 0s which are getting from the last 0^{2p} of *m*. In this case $|vxy| \leq p$, therefore vxy must be a substring 1^p0^p. Then, may be the length of uv^2xy^2z is not become the multiple of 3, or may be the string is in the form of wtw' specified that $|w| = |t| = |w'|$ having all zero's in w and not all zero's in w' $\left(\text{Means}, w^R \neq w'\right)$.

By all these case it is observes that *m* cannot satisfy the conditions of the assumption. Therefore, a contradiction occurs. Hence, it can be state that *A* is not a deterministic context free language.

Comment

Problem

Let $B = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$. Prove that B is not a DCFL.

Step-by-step solution

Step 1 of 1

The following facts will be used to proof " $B = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$ is not a DCFL (Deterministic Context Free Language).

- B is context free.
- If B is deterministic context free then $\overline{B} = \{a, b, c\}^* - B$ is deterministic context free.
- $B_1 = \{a^i b^j c^k \mid i \neq j \text{ and } j \neq k\}$ is not context free .

Here, it is recorded that $\overline{B} \cap a^* b^* c^* = B_1$. **Suppose that $\overline{B}$ was a DCFL (Deterministic Context Free Language) implying that $\overline{B}$ must be a deterministic context free language.** Which in turn implies that $\overline{B}$ is a context free language (CFL). As the **intersection of a CFL and Regular Language is a CFL.** However, B_1 is not a context free language (CFL) that shows a contradiction. **Therefore, B is not Deterministic Context Free Language (DCFL)**.

Comment

Problem

Let $C = \{ww^{\mathcal{R}} \mid w \in \{0,1\}^*\}.$ Prove that C is not a DCFL. (Hint: Suppose that when some DPDA P is started in state q with symbol x on the top of its stack, P never pops its stack below x, no matter what input string P reads from that point on. In that case, the contents of P's stack at that point cannot affect its subsequent behavior, so P's subsequent behavior can depend only on q and x.)

Step-by-step solution

Step 1 of 2

A deterministic context-free language (DCFL) is defined as a language for which there is a DPDA that accepts the same language. Consider the language which is given below:

Language $L_0 = \left\{ww^R \mid w \in \{0,1\}^*\right\}$

The language L_0 which is given above is not deterministic context free language because any pushed down automata (PDA), the use of non-determinism is essential to "guess" the midpoint.

Comment

Step 2 of 2

Consider the table which is given below:

Current State	Input	Stack top	String pushed	Current State	Description
q0	0	z0	0z0	q1	
q0	1	z0	1z0	q1	0. Have to push on this one
q1	0	0	00	q1	…or this one
q1	0	1	01	q1	1a: Suppose not at midpoint have to push on this one.
q1	0	0	$\in$	q1	1b: Suppose at the midpoint
q1	1	1	11	q1	2a: Suppose not at midpoint have to push on this one.
q1	1	0	10	q1	2b:Assume at the midpoint
q1	1	1	$\in$	q1	3. Matched around midpoint
q1	$\in$	z0	z0	q2	

The above table explained the ∂ function of **pushed down automata (PDA), which is designed to recognize L_0.**

• The given PDA is described by the structure:

$$P_{L_0} = \left(\{q_0\}, \{0,1\}, \{0,1,z_0\}, \partial, q_0, z_0, \{q_0, q_2\}\right)$$

The given PDA starts by stacking 0 or 1, depending on which one comes first. The comment 1a and 1b explain the non-deterministic selection of assuming not being at a midpoint, and being a midpoint, respectively. For 2a and 2b similar logic is followed as well.

• Therefore, from the above explanation it can be said that **the language L_0, which is given by $\mathbf{L_0 = \left\{ww^R \mid w \in \{0,1\}^*\right\}}$, is not a Deterministic Context Free Language (DCFL).**

Comment

Problem

If we disallow "-rules in CFGs, we can simplify the *DK*-test. In the simplified test, we only need to check that each of *DK*'s accept states has a single rule. Prove that a CFG without "-rules passes the simplified *DK*-test iff it is a DCFG.

Step-by-step solution

Step 1 of 1

Given:

Consider a context free grammar *C* which does not have ε rule. Apply DK-test on *C* to check whether it is a deterministic CFG or not.

DK Test:

This test is used to check whether CFG is deterministic CFG or not. Consider a CFG C, create DK an associated DFA. Check whether CFG is deterministic by checking the accept state of DK. Every accept states must have:

• Only 1 completed rule.

• There should be no dotted rule that is mean dot should not be immediately after the terminal.

Proof:

Assume a contrary that C is not a deterministic CFG to show the failure of DK-test. Consider a valid string ahb which has *h* as unforced handle. It may possible that some other valid string ahb' has another handle $\hat{h} \neq h$. Therefore ahb' can be rewrite as $\hat{a}\hat{h}\hat{b}$ where b' is a terminal.

• When $ah = \hat{a}\hat{h}$, this lead to change the reduce rule because h and $\hat{h}$ both are two different handle. Hence, ah take the DK to state which has two completed rule which does not satisfy the DK-test.

• When $ah \neq \hat{a}\hat{h}$, the one extends the other. Assume that proper prefix of $\hat{a}\hat{h}$ is ah. There are same arguments with the interchanged string and in place of b' use b.

Assume that *w* is an accepting state in which DK enters on ah input. The state *w* must be an accepting states because h is the handle of ahb. As *w* is the accepting state, so the transition arrow must terminate at w also $\hat{a}\hat{h}$ take DK to accepting state by *w*.

The transition must label with b' because $b' \in \Sigma^+$ also *C* does not have null rule. Hence *w* have dotted rule that is, dot exist immediately after terminal symbol which does not satisfy the DK-test.

Conclusion:

The DK test fails; therefore contradiction occurs that C is not a deterministic CFG. Hence CFG is a DCFG.

Comment