COSC 385.001 – Automata

Spring 2018

Project 2

Completed: March 09, 2018

Deadline: Wednesday, March 07, 2018, 03:00 PM

Problem-8

Jose Dixon

Vojislav Stojkovic

Points: 15

II. Description and Text of Problem

9. L = { w | w ∈ {a, b}\* and a(w) > b(w) }.

a(w) = number of a’s in a string w.

b(w) = number of b’s in a string w.

III. Grammar

Geq = (V, T, P, S)

V = {STUUTABU }

T = {a, b, λ}

P = { S-> aUB, T ->b, U -> aUA, U -> aTU, T -> aTT, A-> λ, B -> λ, U -> λ}

S = a

Part 4. Automaton

M ={Q, ∑, Γ, δ, S, F}

Q = {q0, q1, q2, q3}

∑ = {a, b}

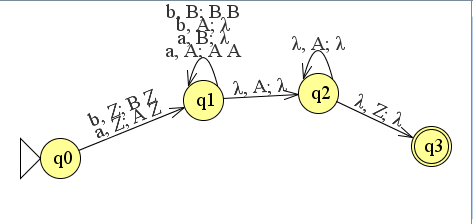
Γ = {A, B, Z}

q0 = q0

Z0 = Z

F = q3

Graph



Tables

|  |  |  |
| --- | --- | --- |
| Q0 | | |
| Δ | a | b |
| Z | Q0 -> q1  Pop(Z)  Adv. | Q0 -> q1  Pop(Z)  Adv. |
| A | Q0 -> q1  Push(AZ)  Adv. |  |
| B |  | Q0 -> q1  Push (BZ)  Adv. |

|  |  |  |  |
| --- | --- | --- | --- |
| Q1 | | | |
| Δ | a | b | λ |
| A | Q1 -> q1  Pop(A)  Push(AA)  Adv. | Q1 -> q1  Pop(A)  Adv. | Q1 -> q2  Pop(A)  Adv. |
| B | Q1 -> q1  Pop(A)  Adv. | Q1 -> q1  Pop(B)  Push (BB)  Adv. |  |
| Z |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Q2 | | | |
| Δ | a | b | λ |
| A |  |  | Q2 -> q2  Pop(A)  Adv. |
| B |  |  |  |
| Z |  |  | Q2 -> q3  Pop(Z)  Adv. |

Part 5. Explanations and Algorithms

## **Basic Structure of PDA**

A pushdown automaton is a way to implement a context-free grammar in a similar way we design DFA for a regular grammar. A DFA can remember a finite amount of information, but a PDA can remember an infinite amount of information.

Basically a pushdown automaton is −

**"Finite state machine" + "a stack"**

A pushdown automaton has three components −

* an input tape,
* a control unit, and
* a stack with infinite size.

The stack head scans the top symbol of the stack.

A stack does two operations −

* **Push** − a new symbol is added at the top.
* **Pop** − the top symbol is read and removed.

A PDA may or may not read an input symbol, but it has to read the top of the stack in every transition.

A PDA can be formally described as a 7-tuple (Q, ∑, S, δ, q0, I, F) −

* **Q** is the finite number of states
* **∑** is input alphabet
* **S** is stack symbols
* **δ** is the transition function: Q × (∑ ∪ {ε}) × S × Q × S\*
* **q0** is the initial state (q0 ∈ Q)
* **I** is the initial stack top symbol (I ∈ S)
* **F** is a set of accepting states (F ∈ Q)

The following diagram shows a transition in a PDA from a state q1 to state q2, labeled as a,b → c −

This means at state **q1**, if we encounter an input string **‘a’** and top symbol of the stack is **‘b’**, then we pop **‘b’**, push **‘c’** on top of the stack and move to state **q2**.

## **Terminologies Related to PDA**

### **Instantaneous Description**

The instantaneous description (ID) of a PDA is represented by a triplet (q, w, s) where

* **q** is the state
* **w** is unconsumed input
* **s** is the stack contents

### **Turnstile Notation**

The "turnstile" notation is used for connecting pairs of ID's that represent one or many moves of a PDA. The process of transition is denoted by the turnstile symbol "⊢".

Consider a PDA (Q, ∑, S, δ, q0, I, F). A transition can be mathematically represented by the following turnstile notation −

(p, aw, Tβ) ⊢ (q, w, αb)

This implies that while taking a transition from state **p** to state **q**, the input symbol **‘a’** is consumed, and the top of the stack **‘T’** is replaced by a new string **‘α’**.

**Note** − If we want zero or more moves of a PDA, we have to use the symbol (⊢\*) for it.

Part 6. Program – Listing

// CS385-Automata-P2.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include <iostream>

#include <string>

using namespace std;

char alphabet{ 'a', 'b' }

char stack\_elements = {'A', 'B', 'Z'}

char stack;

int tape;

char input, pop, push;

void q0(char input, char pop, char push)

{

if (input == b && pop == Z && push == B && push == Z)

push(BZ);

pop(Z);

stack[3] = stack[i] + 1;

else if (input b && pop == Z && push == A && push == Z)

push(AZ);

pop(Z);

stack[2] = stack[i] + 1;

}

void q1(char input, char pop, char push)

{

if (input == a && pop == A && push == A && push == A)

push(A);

push(A);

pop(A);

stack[1] = stack[i] + 1;

else if (input == a && pop == B && push == Ep)

pop(B);

push(Ep);

else if (input == b && pop == A && push == Ep)

pop(A);

push(Ep);

else if (input == b && pop == B && push == B && push == B)

pop(B);

push(B); push(B);

else if (input == Ep && pop == B && push == Ep)

pop(B);

push(Ep);

stack[1] = stack[i] + 1;

}

void q2(char input, char pop, char push)

{

if (input == Ep && pop == B && push == Ep)

pop(B);

else if (input == Ep && pop == Z && push == Z)

pop(Z); push(Z);

stack[0] = stack[i] + 1;

void Acc(q2)

{

if (tape > state[3] = stack[0] + stack[1] + stack[2])

stack[3] =

stack[0] + stack[1] + stack[2];

cout << tape << Stack[3] << "is accepted" << endl;

}

void Rej(q2)

{

if (state[3] != stack[0] + stack[1] + stack[2])

stack[3] =

stack[0] + stack[1] + stack[2];

cout << tape << Stack[3] << "is rejected" << endl;

}

int main()

{

q0();

q1();

q2();

Acc(q2);

Rej(q2);

}

Part 7 – Test Examples

aab is accepted

aa is accepted

bb is rejected

aaabb is accepted