Mathematical Methods for Computer Science 1 Fall 2017

Series 12

Sylvain Julmy

1

The grammar $G = (V, T, P, S)$ where

$$
\begin{aligned}
&V = \{P, A, B\} \\
&T = \{0, 1\} \\
&P = \{ \\
&\quad P \to AB \\
&\quad A \to AA \mid 0 \mid 10 \\
&\quad B \to 1 \mid \epsilon \\
&\quad \} \\
&S = P
\end{aligned}
$$

generates the language $(0 + 10)^*(1 + \epsilon)$

2

a)

We use the following PDA in order to recognize L :

$$P = (\{q_0\}, \{0, 1\}, \{Z_0, 0, 1\}, \delta, q_0, Z_0, \emptyset)$$

where δ is defined as the following :

$$
\begin{aligned}
\delta(q_0, 0, Z_0) &= \{(q_0, 0)\} \\
\delta(q_0, 1, Z_0) &= \{(q_0, 1)\} \\
\delta(q_0, 0, 0) &= \{(q_0, 00)\} \\
\delta(q_0, 0, 1) &= \{(q_0, \epsilon)\} \\
\delta(q_0, 1, 0) &= \{(q_0, \epsilon)\} \\
\delta(q_0, 1, 1) &= \{(q_0, 11)\}
\end{aligned}
$$

b)

We use the following PDA in order to recognize L :

$$P = (\{q_0, q_1, q_2\}, \{0, 1\}, \{Z_0, 0, 1\}, \delta, q_0, Z_0, \{q_2\})$$

where δ is defined as the following :

$$\begin{aligned}
\delta(q_0, 0, Z_0) &= \{(q_1, Z_0 0)\} \\
\delta(q_0, 1, Z_0) &= \{(q_1, Z_0 1)\} \\
\delta(q_1, 0, 0) &= \{(q_1, 00)\} \\
\delta(q_1, 0, 1) &= \{(q_1, \epsilon)\} \\
\delta(q_1, 1, 0) &= \{(q_1, \epsilon)\} \\
\delta(q_1, 1, 1) &= \{(q_1, 11)\} \\
\delta(q_1, \epsilon, Z_0) &= \{q_2, Z_0\}
\end{aligned}$$

3

a)

Using the algorithm described in lecture, we convert $S \to 0S0 \mid 1S1 \mid \epsilon$ to the following PDA :

$$P = (\{q\}, \{0, 1, \epsilon\}, \{0, 1, \epsilon, S\}, \delta, q, S, \emptyset)$$

where δ is defined as the following :

$$\begin{aligned}
&\delta(q, \epsilon, S) = \{(q, 0S0), (q, 1S1), (q, \epsilon)\} \\
&\delta(q, 0, 0) = \{(q, \epsilon)\} \\
&\delta(q, 1, 1) = \{(q, \epsilon)\}
\end{aligned}$$

b)

The word is accepted :

1. Top of the stack is S, consume ϵ from the string, S from the stack and push $0S0$.
2. Top of the stack is $0S0$, consume 0 from the string, 0 from the stack and push ϵ.
3. Top of the stack is $S0$, consume ϵ from the string, S from the stack and push $1S1$.
4. Top of the stack is $1S10$, consume 1 from the string, 1 from the stack and push ϵ.
5. Top of the stack is $S10$, consume ϵ from the string, S from the stack and push $1S1$.

6. Top of the stack is $1S110$, consume 1 from the string, 1 from the stack and push ϵ.
7. Top of the stack is $S110$, consume ϵ from the string, S from the stack and push $0S0$.
8. Top of the stack is $0S0110$, consume 0 from the string, 0 from the stack and push ϵ.
9. Top of the stack is $S0110$, consume S from the string, S from the stack and push ϵ.
10. Top of the stack is 0110, consume 0 from the string, 0 from the stack and push ϵ.
11. Top of the stack is 110, consume 1 from the string, 1 from the stack and push ϵ.
12. Top of the stack is 10, consume 1 from the string, 1 from the stack and push ϵ.
13. Top of the stack is 0, consume 0 from the string, 0 from the stack and push ϵ.
14. The stack is empty, the word is accepted.

Stops before the word is read completely :

1. Top of the stack is S, consume ϵ from the string, S from the stack and push $0S0$.
2. Top of the stack is $0S0$, consume 0 from the string, 0 from the stack and push ϵ.
3. Top of the stack is $S0$, consume ϵ from the string, S from the stack and push $1S1$.
4. Top of the stack is $1S10$, consume 1 from the string, 1 from the stack and push ϵ.
5. Top of the stack is $S10$, consume ϵ from the string, S from the stack and push ϵ.
6. Top of the stack is 10, consume 1 from the string, 1 from the stack and push ϵ.
7. Top of the stack is 0, consume 0 from the string, 0 from the stack and push ϵ.
8. The stack is empty $\rightarrow$ end of the processing without reading the whole word.

4

Let M be the PDA that accept the language L, then we have the transition function δ of M is a map $Q \times (\Sigma \cup \epsilon) \times \Gamma \mapsto Q \times \Gamma^*$.
The idea is to create additional state when a single transition is pushing more than 1 symbol on the stack. Each of those additional transition would push a symbol on the stack.
For each transition $\delta(q_i, a, A_i) \rightarrow (q_j, A_i A_{i+1} \dots A_{i+n})$ where $n > 1$, $a \in \Sigma \cup \epsilon$, $A_i \in S$ we create additional state with corresponding transition function.

$$\delta(q_i, a, A_i) \rightarrow (q_j, A_j A_{j+1} \dots A_{j+n})$$

is transformed into

$$\begin{gathered}
\delta(q_i, a, A_i) \rightarrow (q_j^{(1)}, A_j A_{j+1}) \\
\delta(q_j^{(1)}, \epsilon, A_{j+1}) \rightarrow (q_j^{(2)}, A_{j+1} A_{j+2}) \\
\dots \\
\delta(q_j^{(n-1)}, \epsilon, A_{j+n-1}) \rightarrow (q_j, A_{j+n-1} A_{j+n})
\end{gathered}$$

The language accepted by such a restricted PDA is the same as the original PDA because we add state that did not exist and only transition between those state in a very specific way. Clearly, state from the original PDA can't reach the new state $q_j^{(k)}$ because they simply don't exist in the original PDA. The created state can't reach the states from the original PDA because we only establish transition with the q_i and q_j state with those.
We just simulate a push of multiple symbol by multiple push of one symbol.

5

A PDA with a bounded stack height has a finite number of possibility of storage. Therefore, we can construct a finite state automata from such a PDA. A PDA is an ϵ-automata which has access to a stack in order to store information. The idea is to construct an ϵ-automata from a PDA with a bounded stack height.
The idea is to create state that represents the content of the stack.
I have no time left to demonstrate the idea...