4CS4-6: Theory of Computation (Pushdown Automata)

Lecture 17: March, 26 & 30, 2020

Prof. K.R. Chowdhary *: Professor of CS*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

17.1 Introduction

We have discussed in previous chapters that finite automaton machines accept languages generated by regular expressions. These regular expressions specify the regular languages that are subsets of context-free languages. All the languages generated by CFGs (context-free grammars) that are not regular, hence cannot be accepted by the finite automata. Therefore, the regular languages and regular grammars are proper subsets of Context-free Languages (CFLs) and CFGs, respectively.

Let us consider the CFL $L = \{u = ww^R \mid w \in (a, b)^*\}$. This language can be generated by CFG $G = (V, \Sigma, S, P)$, where P is given as:

$$S \to aSa \mid bSb \mid \varepsilon. \tag{17.1}$$

We note that CFL $L = L(G)$ in above is a palindrome language, as in each string $w \in L(G)$, there is a mirror image of the string from the center of w on each side with length $|w|/2$. Therefore, any device which can recognizes the string w, must read it from left to right, and store the first half of it in some form, then compare the second half of the input when read, with the previously stored half input in reverse order, i.e., the last symbol stored is compared first. If there is perfect match, then $w \in L(G)$, else $w \notin L(G)$. Note that, for this purpose, the recognizing device needs to store one half of the input string, which may be of arbitrary length. Thus, the finite automata cannot recognize L, as the later does not have memory. This is one proof that the strings generated by grammar (17.1) are not in regular language.

The purpose of stack is to allow the PDA to remove the letters in the word that it has read so that it can duplicate them with the other letters/symbols.

A machine which is modified and upgraded version of a finite automata, and capable of accumulating the input symbols as they are read by appending them one at a time to a stored string, can recognize the palindrome language. It somehow (non-deterministically) finds that center of the input string has reached. Thereafter, as it reads the remaining part of the string and every time a symbol is read, another symbol is taken off from the already stored string, in Last-In-First-Out (LIFO) order, and compared with the current symbol read from input. In this way, all the rest half of the input symbols are compared with the symbols already read and stored. The string is considered accepted by this new version of finite automata called Pushdown Automata (PDA) if there is match in all the second half of the symbols with first half, in reverse order.

As another example, a CFG G with productions $S \to SS \mid (S) \mid \varepsilon$, which generates strings of balanced parentheses, is a non-regular language. To recognize the language strings of this grammar, a PDA can be used, which inserts (pushes) every left parenthesis read from input string into the stack and retrieves a pushed symbol from top of the stack when a right parenthesis is read from the input. When stack is empty and at the same time the input sequence of symbols terminates, shows that input string has balanced parentheses hence the string is accepted. Otherwise, the string is rejected, indicating that it is not a balanced parenthesis sequence. A typical derivation sequence of string for this grammar can be:

$$\begin{aligned} S &\Rightarrow SS \Rightarrow (S)(S) \Rightarrow (SS)(S) \Rightarrow ((S)(S))(S) \\ &\Rightarrow^* (()())() \end{aligned}$$

which is a balanced parentheses string, however, the center of this string is not explicit.

Example 17.1 *Consider a grammar with productions $S \to (S) \mid \varepsilon$. Some strings that can be generated using this grammar are:*

$$S \Rightarrow (S) \Rightarrow ((S)) \Rightarrow (())$$
$$S \Rightarrow (S) \Rightarrow ((S)) \Rightarrow (((S))) \Rightarrow ((()))$$

Since all the left parentheses get exhausted before start of the first closing parenthesis, centre of the string can be determined with certainty. Therefore a PDA that can recognize these strings is deterministic.

17.2 Pushdown Automata

Theorem 17.2 *A set of strings is a context-free language if and only if it is accepted by a nondeterministic pushdown storage automaton (Chomsky, 1962; Schützenberger, 1963)*

The diagram of the pushdown automata reading the input string *abaabbaaba* is shown in the figure 17.1. The storage device used in the pushdown automata is called *stack* or *pushdown store*, which inserts (stores) the symbols at top as they are read from input. When it reads these back from store, they are retrieved from the top in LIFO order only.

A pushdown automaton consists of three parts: a read-only input tape, a finite state control and a read-write pushdown store, called stack Depending on the state this automaton is in, the input symbol it is reading, and the symbol at the top of the stack, the PDA may make a transition to any one of the finite set of states, and writes a finite number of symbols on the stack. Writing a null-symbol corresponds to erasing the top-most symbol.

In addition, the automata may make transitions and write symbols on stack without reading symbols from input. Such transitions are called ε-moves. Thus a PDA can manipulate the

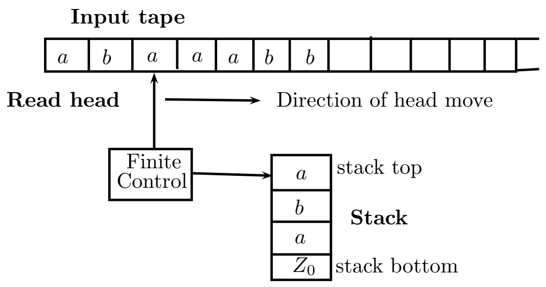


Figure 17.1: Pushdown Automata.

stack without reading anything. In fact, it can be proved that for a PDA with ε-moves there exists an equivalence PDA without ε-moves, which accepts the same language. It is assumed that the pushdown automata has potentially unbounded memory, and it is the only fact that its access to this memory is highly restricted, i.e. LIFO (Last in First Out) manner. Due to this reason the PDA is not equivalent to a Turning machine, discussed in the next chapters.

A pushdown automata is a 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F) \tag{17.2}$$

where,

Q is a finite non-empty set of states

Σ is a finite non-empty set of input symbols

Γ is finite non-empty set of stack symbols, and $\varepsilon \in \Gamma$,

$q_0 \in Q$ is initial state

$Z_0 \in \Gamma$ is initial symbol on the pushdown stack

$F \subseteq Q$ is set of final states

and δ is transition function, specified as,

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \text{ finite subsets of } Q \times \Gamma^*. \tag{17.3}$$

The transition function δ shows that a PDA, while in state $q_i \in Q$ and reading an input symbol or null (nothing has been read) and with some symbol on top of stack, will cause a transition to new state $q_j \in Q$. And, while going in transitions through these states will write finite number of symbols (or null, i.e. no symbol) on stack.

17.3 Definitions

Definition 17.3 Instantaneous Description. *Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. An Instantaneous Description (ID) of M is an element of $Q \times \Sigma^* \times \Gamma^*$. An initial ID is* *a member of $\{q_0\} \times \Sigma^* \times \{Z_0\}$. Therefore, $(q, ax, Z\alpha)$ is an ID for PDA in state $q \in Q$, currently reading input $a \in \Sigma$, and $x \in \Sigma^*$ is string yet to be read, having $Z \in \Gamma$ on the stack, and $\alpha \in \Gamma^*$ is symbol sequence in stack below Z. In a special case a may be ε, in that case the PDA will make a ε-move, that is, the input (read head) is not advanced, but may write on stack, and make a transition to a new stack.*

Definition 17.4 Transition function. *The transition function δ for PDA is defined as a mapping $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ to finite subset of $Q \times \Gamma^*$, which can be specified as,*

$$\delta(q, a, Z) = \text{ subset of } \{(p_1, \beta_1), (p_2, \beta_2), ..., (p_m, \beta_m)\}. \tag{17.4}$$

Hence $(p_1, \beta_1) \in \delta(q, a, Z)$, or in general $(p_i, \beta_i) \in \delta(q, a, Z)$. Here $q, p_i \in Q$, $a \in \Sigma$, $Z \in \Gamma$, $\beta_i \in \Gamma^*$ for $0 \leq i \leq m$. The above transition indicates that a PDA, which is in state q with input symbol a, the Z as the top symbols on stack can for any i enters state p_i and replaces the symbol on top of stack by a string β_i and advance the read head by one step. In particular, if $m = 0$, the right hand side of the above equation denotes empty set, which means that no next move is defined. It is assumed that for β_i, its left most symbol is on the top of the stack, simply for an input x, the read head points to the left most symbol in x.

Furthermore, it is emphasized that in equation (17.4),

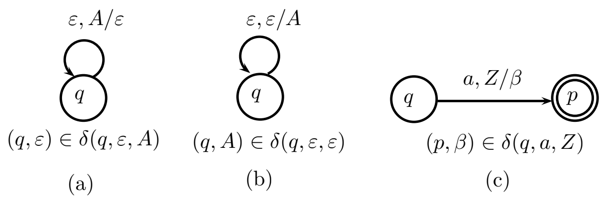
- if $a = \varepsilon$, the input head does not move,
- if $Z = \varepsilon$, the symbol at top of stack is empty,
- if $\beta_i = \varepsilon$, nothing is added (pushed) to stack.

Since a PDA is non-deterministic, therefore it can manipulate the symbols on stack without any input. This is expressed by state transition

$$\delta(q, \varepsilon, Z) = \{(p_1, \beta_1), (p_2, \beta_2), ..., (p_m, \beta_m)\}, \tag{17.5}$$

where current input symbol read is ε (i.e., no input has been read, and therefore the read head of PDA does not advance), q is current state, Z is symbol on top of stack. With this input, the PDA moves to state p and the symbol Z on the top of the stack is replaced by sequence β_i, where $1 \leq i \leq m$.

In general, if current state q of PDA with input a, and symbol on top of stack Z, causes the automata to move to state p and symbol on top of stack is replaced by β, then we express it as $(p, \beta) \in (q, a, Z)$.



$\varepsilon, A/\varepsilon$
q
$(q, \varepsilon) \in \delta(q, \varepsilon, A)$
(a)
$\varepsilon, \varepsilon/A$
q
$(q, A) \in \delta(q, \varepsilon, \varepsilon)$
(b)
q
$a, Z/\beta$
p
$(p, \beta) \in \delta(q, a, Z)$
(c)

Figure 17.2: Some transitions in PDA.

Some of the transitions in PDA shown in figure 17.2(a), (b), (c). The symbols, "a, Z/β" along with transition in 17.2(c) shows that the PDA currently in state q reads symbol a on tape, with symbol Z on top of the stack, will move in state p after transition, and will write β on top of the stack.

Definition 17.5 Moves of PDA. *Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. We define a relation called "move relation" by,*

$$(q, ax, Z\alpha) \vdash_M (q', x, \beta\alpha) \tag{17.6}$$

if $(q', \beta) \in \delta(q, a, Z)$ for any $q, q' \in Q$, any $a \in \Sigma \cup \{\varepsilon\}$, any $x \in \Sigma^*$, and $\alpha, \beta \in \Gamma^*$ and $Z \in \Gamma$.

Example 17.6 *Design a PDA that recognizes the language $L = \{a^n b^n \mid n \geq 0\}$.*

Assume that the PDA recognizing this language is $M = (Q, \Sigma, \delta, s, F, \Gamma, Z_0)$. Here $\Sigma = \{a, b\}$, $s = q_0$, and Z_0 is top most character on stack at start. The other parameters are yet to be determined.

The language L states that there are certain number of a's in the input string, and it is followed by exactly same number of b's. This can be recognized by, first reading all a's, and for each a read, a fixed symbol (say $A \in \Gamma$) is written into stack. When second part of the input string (containing characters b) is read, the already stored symbol A is retrieved back from the stack for each b read from input. If completion of input string and emptying out stack (retains original Z_0) coincides, we say the input is accepted. Accordingly, Q, δ, F, are defined as follows.

$$Q = \{q_0, q_1, q_2, q_3\},\ F = \{q_0, q_3\},\ \Gamma = \{\varepsilon, A, Z_0\},\ \delta(q_0, a\varepsilon) = (q_1, Z_0)$$

$$\delta(q_1, a, \varepsilon) = (q_1, A)$$

$$\delta(q_1, b, A) = (q_2, \varepsilon),$$

$$\delta(q_2, b, A) = (q_2, \varepsilon),\ \delta(q_2, \varepsilon, Z_0) = (q_3, \varepsilon)$$

Since $L = \{a^n b^n \mid n \geq 0\}$, the final state may occur at q_3 for $n = 0$, and at q_1 for $n > 0$. An input string $aabb \in L$ is thus accepted as follows.

$$\begin{aligned}
(q_0, aabb, \varepsilon) &\vdash_M (q_1, aabb, Z_0)\\
&\vdash_M (q_1, abb, AZ_0)\\
&\vdash_M (q_1, bb, AAZ_0)\\
&\vdash_M (q_2, b, AZ_0)\\
&\vdash_M (q_2, \varepsilon, Z_0)\\
&\vdash_M (q_3, \varepsilon, \varepsilon),\ \text{(The PDA halts)}.
\end{aligned}$$

This shows that for the input $aabb$, when tape string has been fully read, the pushdown stack contents are also null (Z_0 is not counted, being the end character of stack). This satisfies the condition of acceptability; hence $aabb \in L$. Figure 17.3 shows the transition diagram of this PDA.

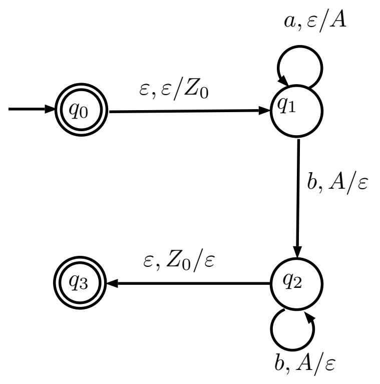


Figure 17.3: PDA recognizing language $\{a^n b^n \mid n \geq 0\}$.