1

Context-Free Languages (10)

1.1 Pushdown Automata (10.8)

Automata That Accept Context-free Languages?
What kind of automaton is needed for accepting context-free languages? For a Chomsky normal form context-free grammar $\Gamma$ with terminals $T$, and additional bracket symbols $P$,

- Theorem 7.2 says $E_{P}(L(\Gamma_s)) = L(\Gamma)$. 


Theorem 7.3 says $L(\Gamma_s) = R \cap \text{PAR}_n(T)$.

We shall first try to construct an appropriate automaton for recognizing $L(\Gamma_s)$.

$R$ is accepted by a finite automaton; we need additional facilities to check if some given words belong to $\text{PAR}_n(T)$.

A first-in-last-out “pushdown stack” is needed to recognize $\text{PAR}_n(T)$.

**Pushdown Stack**

At each step, one or both of the following operations can be performed:

1. The symbol at the “top” of the stack may be read and discarded. This operation is called “popping the stack”.

2. A new symbol may be “pushed” onto the stack.

A stack can be used to identify a string as belonging to $\text{PAR}_n(T)$ as follows:

- A special symbol $J_i$ is introduced for each pair $(i,)_i, i = 1, 2, \ldots, n$.

- As the automaton moves from left to right over a string, it pushes $J_i$ onto the stack whenever it sees $(i$, and it pops the stack, eliminating a $J_i$, whenever it sees $)_i$.

- In case the string belongs to $\text{PAR}_n(T)$, the automaton will terminate with an empty stack after moving to the right end of the string.

**Notations**

Let $T$ be a given alphabet and let $P = \{(i,)_i | i = 1, 2, \ldots, n\}$. Let $\Omega = \{J_1, J_2, \ldots, J_n\}$, where we have introduced a single symbol $J_i$ for each pair $(i,)_i, 1 \leq i \leq n$. Let $u \in (T \cup P)^*$, say, $u = c_1c_2 \ldots c_k$, where $c_1, c_2, \ldots, c_k \in T \cup P$.

We define a sequence $\gamma_j(u)$ of elements of $\Omega^*$ to characterize the content of the pushdown stack as follows:

\[
\begin{align*}
gamma_1(u) &= 0 \\
gamma_{j+1}(u) &= \begin{cases} 
gamma_j(u) & \text{if } c_j \in T \\
J_i \gamma_j(u) & \text{if } c_j = (i \\
\alpha & \text{if } c_j = )_i \text{ and } \gamma_j(u) = J_i \alpha \end{cases}
\end{align*}
\]

for $j = 1, 2, \ldots, k$. Note that if $c_j = )_i$, $\gamma_{j+1}(u)$ will be undefined unless $\gamma_j$ begins with the symbol $J_i$ for the very same value of $i$. Furthermore, if a particular $\gamma_r(u)$ is undefined, all $\gamma_j(u)$ with $j > r$ will also be undefined.
Words in \( \text{PAR}_n(T) \) Are Balanced

**Definition.** We say that the words \( u \in (T \cup P)^* \) is balanced if \( \gamma_j(u) \) is defined for \( 1 \leq j \leq |u| + 1 \) and \( \gamma_{|u|+1}(u) = 0 \).

**Theorem 8.1.** Let \( T \) be an alphabet and let 
\[
P = \{(i, i) \mid i = 1, 2, \ldots, n\}, \quad T \cap P = \emptyset.
\]
Let \( u \in (T \cup P)^* \). Then \( u \in \text{PAR}_n(T) \) if and only if \( u \) is balanced. The proof of Theorem 8.1 is via a series of easy lemmas.

**Lemmas**

**Lemma 1.** If \( u \in T^* \), then \( U \) is balanced. **Lemma 2.** If \( u \) and \( v \) are balanced, so \( uv \).

**Lemma 3.** Let \( v = (i \ u \ )_i \). Then \( u \) is balanced if and only if \( v \) balanced. **Lemma 4.** If \( u \) is balanced and \( uv \) balanced, then \( v \) is balanced.

**Lemma 5.** If \( u \in \text{PAR}_n(T) \), then \( u \) is balanced. **Lemma 6.** If \( u \) is balanced, the \( u \in \text{PAR}_n(T) \).

**Pushdown Automata**

A pushdown automaton \( \mathcal{M} \) consists of

- a finite set of states \( Q = \{q_1, \ldots, q_m\} \), where \( q_1 \) is the initial state, and \( F \subseteq Q \) is the set of final, or accepting, states,

- a tape alphabet \( A \),

- a pushdown alphabet \( \Omega \),

- a symbol \( 0 \) not in \( A \) nor in \( \Omega \), and

- a finite set of transitions which each is a quintuple of the form

\[
q_i a U : V q_j
\]

where \( a \in \tilde{A} = A \cup \{0\}, U, V \in \tilde{\Omega} = \Omega \cup \{0\} \).

Intuitively, if \( a \in A \) and \( U, V \in \Omega \), the quintuple reads: “In state \( q_i \) scanning \( a \), with \( U \) on top of the stack, move one square to the right, ‘pop’ the stack removing \( U \), ‘push’ \( V \) onto the stack, and enter state \( q_j \).”

**Pushdown Automata, Continued**

For the quintuple
\[
q_i a U : V q_j
\]
where either \( a, U, V \) is \( 0 \), the transition is defined as the following.

- If \( a = 0 \), motion to the right does not take place and the stack action can occur regardless of what the symbol is actually being scanned.

- If \( U = 0 \), then nothing is to be popped.
• If \( V = 0 \), then nothing is to be pushed.

Furthermore, the distinct transitions \( q_i aU : V q_j \), \( q_ibW : Xq_k \) are called incompatible if one of the following is the case:

1. \( a = b \), and \( U = W \);
2. \( a = b \), and \( U \) or \( W \) is 0;
3. \( U = W \), and \( a \) or \( b \) is 0;
4. \( a \) or \( b \) is 0, and \( U \) or \( W \) is 0.

A pushdown automaton is deterministic if it has no pair of incompatible transitions.

**Configurations of Pushdown Automata**

Let \( u \in A^* \) and let \( M \) be a pushdown automaton. Then a **u-configuration for** \( M \) is a triple \( \Delta = (k, q_i, \alpha) \), where \( 1 \leq k \leq |u| + 1 \), \( q_i \) is a state of \( M \), and \( \alpha \in \Omega^* \). Intuitively, the u-configuration \( (k, q_i, \alpha) \) stands for the situation in which \( u \) is written on \( M \)’s tape, \( M \) is scanning the \( k \)th symbol of \( U \) — or, if \( k = |u| + 1 \), has completed scanning \( u \) — and \( \alpha \) is the string of symbols on the pushdown stack. We speak of \( q_i \) as the state of configuration \( \Delta \), and of \( \alpha \) as the stack contents at configuration \( \Delta \). If \( \alpha = 0 \), we say the stack is empty at \( \Delta \).

**Configurations of Pushdown Automata, Continued**

For a pair of \( u \)-configurations, we write \( u : (k, q_i, \alpha) \vdash_M (l, q_k, \beta) \) if \( M \) contains a transition \( q_i aU : V q_j \), where \( \alpha = U\gamma, \beta = V\gamma \) for some \( \gamma \in \Omega^* \), and either

1. \( l = k \) and \( a = 0 \), or
2. \( l = k + 1 \) and the \( k \)th symbol of \( u \) is \( a \).

Note that the equation \( \alpha = U\gamma \) is to be read simply \( \alpha = \gamma \) in case \( U = 0 \); likewise for \( \beta = V\gamma \).

**Computation by Pushdown Automata**

A sequence \( \Delta_1, \Delta_2, \ldots, \Delta_m \) of u-configurations is called a **u-computation by** \( M \) if

1. \( \Delta_1 = (1, q, 0) \) for some \( q \in Q \);
2. \( \Delta_m = (|u| + 1, p, \gamma) \) for some \( p \in Q \) and \( \gamma \in \Omega^* \), and
3. \( u : \Delta_i \vdash_M \Delta_{i+1} \), for \( 1 \leq i < m \).

This u-computation is called accepting if the state at \( \Delta_1 \) is the initial state \( q_1 \), the state \( p \) at \( \Delta_m \) is in \( F \), and the stack at \( \Delta_m \) is empty. We say that \( M \) accepts the string \( u \in A^* \) if there is an accepting u-computation by \( M \). We write \( L(M) \) for the set of strings accepted by \( M \), and we call \( L(M) \) the **language accepted by** \( M \).
Pushdown Automata, Examples
See Examples $M_1, M_2$, and $M_3$ at page 312 in the textbook.

Separators and Deterministic Pushdown Automata

**Theorem 8.2.** Let $\Gamma$ be a Chomsky normal form grammar with separator $\Gamma_s$. Then there is a deterministic pushdown automaton $M$ such that $L(M) = L(\Gamma_s)$. **Proof Outline.**

By Theorem 7.3, for suitable $n$,

$$L(\Gamma_s) = R \cap \text{PAR}_n(T),$$

where $R$ is a regular language, and $T$ is the set of terminals of $\Gamma$. Let $P = \{(i_i) | i = 1, 2, \ldots, n\}$, and $M_0$ be a dfa with alphabet $T \cup P$ that accepts $R$. Let $Q = \{q_1, q_2, \ldots, q_m\}$ be the states of $M_0$, $q_1$ the initial states, $F \subseteq Q$ the accepting states, and $\delta$ the transition function. We construct a pushdown automaton $M$ with tape alphabet $T \cup P$ and the same states, initial state, and accepting states as $M_0$. $M$ is to have the pushdown alphabet $\Omega = \{J_1, \ldots, J_n\}$.

Separators and Deterministic Pushdown Automata, Continued

**Proof Outline (Continued).** The transitions of $M$ are as follows for all $a \in Q$:

1. for each $a \in T$, $qa0 : 0p$, where $p = \delta(q, a)$;
2. for $i = 1, 2, \ldots, n$, $q_i0 : J_i p_i$, where $p_i = \delta(q, (i))$;
3. for $i = 1, 2, \ldots, n$, $q_i J_i : 0 \bar{p}_i$, where $\bar{p}_i = \delta(q, (i))$

Note that, by definition, $M$ is deterministic. It remains to be proved that, for any $u \in L(\Gamma_s)$, there is an accepting $u$-computation by $M$ ($\Rightarrow$). Conversely, we need to prove that, if $M$ accepts $u \in (T \cup P)^*$, then there is a derivation of $u$ in $\Gamma_s$ ($\Leftarrow$). $\square$

Separators and Deterministic Pushdown Automata, Continued

**Proof Outline (Continued).** ($\Rightarrow$) Let $u = c_1c_2 \ldots c_K \in L(\Gamma_s)$, where $c_1, c_2, \ldots, c_K \in (T \cup P)$. Then there is a sequence of states $p_1, p_2, \ldots, p_{K+1} \in Q$ such that $p_1 = q_1, p_{K+1} \in F$, and $\delta(p_i, c_i) = p_{i+1}, i = 1, 2, \ldots, K$. Since $u \in \text{PAR}_n(T)$, by Theorem 8.1, $u$ is balanced, so that $\gamma_j(u)$ is defined for $j = 1, 2, \ldots, K + 1$ and $\gamma_{K+1}(u) = 0$. We let

$$\Delta_j = (j, p_j, \gamma_j(u)), \quad j = 1, 2, \ldots, K + 1.$$ 

It follows that

$$u : \Delta_j \vdash_M \Delta_{j+1}, \quad j = 1, 2, \ldots, K.$$ 

Thus $\Delta_1, \Delta_2, \ldots, \Delta_{K+1}$ is an accepting $u$-computation by $M$. 5
Separators and Deterministic Pushdown Automata, Continued

Proof Outline (Continued). \(\Leftarrow\) Conversely, let \(\mathcal{M}\) accept \(u = c_1 c_2 \ldots c_K\). Thus \(\Delta_1, \Delta_2, \ldots, \Delta_{K+1}\) is an accepting \(u\)-computation by \(\mathcal{M}\). Let \(\Delta_j = (j, p_j, \gamma_j), j = 1, 2, \ldots, K + 1\). Since

\[ u : \Delta_j \vdash \mathcal{M} \Delta_{j+1}, \quad j = 1, 2, \ldots, K \]

and \(\gamma_1 = 0\), we see that \(\gamma_j\) satisfies the defining recursion for \(\gamma_j(u)\) and hence, \(\gamma_j = \gamma_j(u)\) for \(j = 1, 2, \ldots, K + 1\). Since \(\gamma_{K+1} = 0\), \(u\) is balanced and hence \(u \in \text{PAR}_n(T)\). Finally, we have \(p_1 = q_1, p_{K+1} \in F\), and \(\delta(p_j, c_j) = p_{j+1}\). Therefore the dfa \(\mathcal{M}_0\) accepts \(u\), and \(u \in R\). We conclude that \(u \in L(\Gamma_s)\).

\(\square\)

Atomic Pushdown Automata

A pushdown automaton is called atomic (whether or not it is deterministic) if all of its transition are one of the following forms:

1. \(pa_0 : 0q\),
2. \(p0U : 0q\),
3. \(p00 : Vq\).

Thus, at each step in a computation an atomic pushdown automaton can read the tape and move right, or pop a symbol off the stack or push a symbol on the stack. But, unlike pushdown automata in general, it cannot perform more than one of these actions in a single step. We will later show that for any pushdown automata \(\mathcal{M}\), there is an atomic pushdown automata \(\bar{\mathcal{M}}\) such that \(L(\mathcal{M}) = L(\bar{\mathcal{M}})\).

Computation Records of Atomic Pushdown Automata

Let \(\mathcal{M}\) be a given atomic pushdown automata with tape alphabet \(T\) and pushdown alphabet \(\Omega = \{J_1, J_2, \ldots, J_n\}\). We set

\[ P = \{(i,)i \mid i = 1, 2, \ldots, n\} \]

and show how to use the “brackets” to define a kind of “records” of a computation by \(\mathcal{M}\). Let \(\Delta_1, \Delta_2, \ldots, \Delta_m\) be a \(v\)-computation by \(\mathcal{M}\), where \(v = c_1 c_2 \ldots, c_K\) and \(c_k \in T, k = 1, 2, \ldots, K\), and where \(\Delta_i = (l_i, p_i, \gamma_i), i = 1, 2, \ldots, m\). We set

\[
\begin{align*}
    w_1 &= 0 \\
    w_{i+1} &= \begin{cases} 
        w_i c_i & \text{if } \gamma_{i+1} = \gamma_i \\
        w_{i+j} & \text{if } \gamma_{i+1} = J_j \gamma_i \\
        w_i j & \text{if } \gamma_i = J_j \gamma_{i+1}
    \end{cases} \quad 1 \leq i < m
\end{align*}
\]
Computation Records of Atomic Pushdown Automata, Continued

Now let \( w = w_m \), so that \( \text{Exp}(w) = v \) and \( m = |w| + 1 \). This word \( w \) is called the record of the given \( v \)-computation \( \Delta_1, \ldots, \Delta_m \) by \( \mathcal{M} \). From \( w \) we can read off not only the word \( v \) but also the sequence of “pushes” and “pops” as they occur. In particular, \( w_i, 1 < i \leq m \), indicates how \( \mathcal{M} \) goes from \( \Delta_{i-1} \) to \( \Delta_i \).

An Atomic Automaton for \( L(\Gamma) \)

We now modify the pushdown automaton \( \mathcal{M} \) of Theorem 8.2 so that it will accept \( L(\Gamma) \) instead of \( L(\Gamma_s) \). The idea is to use nondeterminism to “guess” the location of the “brackets” \((i, j)\). Continuing to use the notation of the proof of Theorem 8.2, We define a pushdown automaton \( \mathcal{M} \) with the same states, initial state, accepting states, the pushdown alphabet as \( \mathcal{M} \). However, the tape alphabet of \( \mathcal{M} \) will be \( T \) (rather than \( T \cup P \)). The transitions of \( \mathcal{M} \) are, for all \( q \in Q \):

1. for each \( a \in T \), \( qa0 : 0p \), where \( p = \delta(q, a) \);
2. for \( i = 1, 2, \ldots, n \), \( q00 : J_ip_i \), where \( p_i = \delta(q, (i)) \);
3. for \( i = 1, 2, \ldots, n \), \( q0J_i : 0p_i \), where \( p_i = \delta(q, i) \).

Depending on the transition function \( \delta \), \( \mathcal{M} \) can certainly be non-deterministic. Note that \( \mathcal{M} \) is atomic (though \( \mathcal{M} \) is not). It remains to be proved that \( L(\mathcal{M}) = L(\Gamma) \).

\( v \in L(\Gamma) \Rightarrow v \in L(\mathcal{M}) \)

Let \( v \in L(\Gamma) \). Then, since \( \text{Exp}(L(\Gamma_s)) = L(\Gamma) \), there is a word \( w \in L(\Gamma_s) \) such that \( \text{Exp}(w) = v \). By Theorem 8.2, \( w \in L(\mathcal{M}) \). Let

\[
\Delta_i = (i, p_i, \gamma_i), \quad i = 1, 2, \ldots, m
\]

be an accepting \( w \)-computation by \( \mathcal{M} \) (with \( m = |w| + 1 \)). Let \( n_i = 1 \) if \( w : \Delta_i \vdash_{\mathcal{M}} \Delta_{i+1} \) is via transition \( qa0 : 0p \) (with \( p = \delta(q, a) \)); otherwise \( n_i = 0, 1 \leq i < m \). Let

\[
l_1 = 1, \\
l_{i+1} = l_i + n_i, \quad 1 \leq i < m.
\]

Finally let

\[
\bar{\Delta}_i = (l_i, p_i, \gamma_i), \quad 1 \leq i < m.
\]

Now, it can be checked that

\[
v : \bar{\Delta}_i \vdash_{\mathcal{M}} \bar{\Delta}_{i+1}, \quad 1 \leq i < m.
\]

Since \( \bar{\Delta}_m = (|v| + 1, q, 0) \) with \( q \in F \), we conclude \( v \in L(\mathcal{M}) \).
Let \( v \in L(A) \Rightarrow v \in L(\Gamma) \).
Let \( v \in L(A) \). Let
\[
\bar{\Delta}_i = (l_i, p_i, \gamma_i), \quad i = 1, 2, \ldots, m
\]
be an accepting \( v \)-computation by \( A \). Using the fact that \( A \) is atomic, we can let \( w \) be the record of this computation as defined earlier so that \( \text{Er}_P(w) = v \) and \( m = |w| + 1 \). Let \( \Delta_i = (i, p_i, \gamma_i), i = 1, 2, \ldots, m \), and we observe that
\[
w : \Delta_i \vdash_A \Delta_{i+1}, \quad i = 1, 2, \ldots, m.
\]
Since \( p_m \in F \) and \( \gamma_m = 0 \), \( \Delta_1, \Delta_2, \ldots, \Delta_m \) is an accepting \( w \)-computation by \( A \). Thus by Theorem 8.2, \( w \in L(\Gamma_s) \). Hence, \( v \in L(\Gamma) \).

Context-free Languages and Pushdown Automata

**Theorem 8.3.** Let \( \Gamma \) be a Chomsky normal form context-free grammar. Then there is a pushdown automaton \( A \) such that \( L(A) = L(\Gamma) \). \( \square \)

**Theorem 8.4.** For every context-free grammar \( L \), there is a pushdown automaton \( A \) such that \( L = L(A) \).

Note that to prove Theorem 8.4, we need to take care of the case where \( 0 \in L \), hence \( L = L(\Gamma) \cup \{0\} \) for a Chomsky normal form context-free grammar \( \Gamma \). For such a case, we need to modify the pushdown automaton \( A \) that accepts \( L(\Gamma) \). Actually we modify the dfa component \( A_0 \) of \( A \) to build an equivalent nonrestarting dfa. After that, we add the initial state of this new dfa to the set of accepting states so that \( 0 \) will be recognized.

Atomic Pushdown Automata, Revisited

**Theorem 8.5.** Let \( A \) be a pushdown automaton. Then there is an atomic pushdown automaton \( \bar{A} \) such that \( L(A) = L(\bar{A}) \).

**Proof.** For each transition \( paU : Vq \) of \( A \) for which \( a, U, v \neq 0 \), we introduce two new states \( r, s \) and let \( \bar{A} \) have the transitions
\[
pr : 0r, \quad r0U : 0s, \quad s00 : Vq.
\]
If exactly one of \( a, U, V \) is 0, the only two transitions are needed for \( \bar{A} \). For each transition \( p00 : 0q \), we introduce a new state \( t \) and replace \( p00 : 0q \) with the transitions
\[
p00 : Jt, \quad t0J : 0q
\]
where \( J \) is an arbitrary symbol of the pushdown alphabet. Otherwise, \( \bar{A} \) is exactly like \( A \). Clearly, \( L(A) = L(\bar{A}) \). \( \square \)
Context-free Languages and Pushdown Automata

Theorem 8.6. For every pushdown automaton $\mathcal{M}$, $L(\mathcal{M})$ is a context-free language.

Proof Outline. Without loss of generality, we assume $\mathcal{M}$ is atomic. The plan is to prove that for the language $L$ consisting exactly of the records of all accepting $u$-computation by $\mathcal{M}$, where $u \in L(\mathcal{M})$, we will have $L = R \cap \text{PAR}_n(T)$. $R$ will be a regular language accepted by a ndfa $\mathcal{M}_0$ devised from $\mathcal{M}$, and $T$ is tape alphabet of $\mathcal{M}$. As $L(\mathcal{M}) = \text{Er}_P(L)$, it follows that $L(\mathcal{M})$ is a context-free language. To prove $L = R \cap \text{PAR}_n(T)$, we need to show both $L \subseteq R \cap \text{PAR}_n(T)$ and $R \cap \text{PAR}_n(T) \subseteq L$.

Context-free Languages and Pushdown Automata

Proof Outline of Theorem 8.6, Continued. Let $\mathcal{M}$ have states $Q = \{q_1, q_2, \ldots, q_m\}$, initial state $q_1$, final states $F$, tape alphabet $T$, and pushdown alphabet $\Omega = \{J_1, \ldots, J_m\}$. To devise ndfa $\mathcal{M}_0$, we need $P = \{(i, i) \mid i = 1, \ldots, m\}$. $\mathcal{M}_0$ has the same states, initial state, and accepting states as $\mathcal{M}$, and transition function $\delta$ defined as follows. For each $q \in Q$,

$$\delta(q, a) = \{p \in Q \mid \mathcal{M} \text{ has the transition } qa0: 0p\} \text{ for } a \in T$$
$$\delta(q, (i) = \{p \in Q \mid \mathcal{M} \text{ has the transition } q00: J_ip\}, \quad i = 1, \ldots, n,$$
$$\delta(q, (i) = \{p \in Q \mid \mathcal{M} \text{ has the transition } q0J_i: 0p\}, \quad i = 1, \ldots, n.$$

Let $w \in L$ be the record of an accepting $u$-computation $\Delta_1, \ldots, \Delta_m$, where $\Delta_i = (l_i, p_i, \gamma_i), i = 1, \ldots, m$. By an induction, we can show that $p_m \in \delta^*(q_1, w)$. As $p_m \in F$, we have $w \in R$. By another induction, we can show that $\gamma_i(w) = \gamma_i, i = 1, \ldots, m$. As $\gamma_{|w|+1}(w) = \gamma_{|w|+1} = 0$, we know $w$ is balanced. We conclude that $w \in R \cap \text{PAR}_n(T)$.

Context-free Languages and Pushdown Automata

Proof Outline of Theorem 8.6, Continued. Conversely, let $w = c_1 \ldots c_r \in R \cap \text{PAR}_n(T)$, and let $u = \text{Er}_P(w) = d_1, \ldots, d_k$. Let $p_1, \ldots, p_{r+1}$ be some sequence of states such that $p_1 = q_1, p_{r+1} \in \delta(p_i, c_i)$ for $i = 1, \ldots, r$. We claim that

$$(l_1, p_1, \gamma_1(w)), \quad (l_2, p_2, \gamma_2(w)), \quad \ldots, \quad (l_{r+1}, p_{r+1}, \gamma_{r+1}(w))$$

where

$$l_1 = 1$$
$$l_{i+1} = \begin{cases} l_i + 1 & \text{if } c_i \in T \\ l_i & \text{otherwise} \end{cases}$$

is an accepting $u$-computation by $\mathcal{M}$ and $w$ is its record. That is, we need to show that

$$u : (l_r, p_r, \gamma_r(w)) \vdash_{\mathcal{M}} (l_{r+1}, p_{r+1}, \gamma_{r+1}(w))$$

for $i = 1, \ldots, r$. This is done by an induction $i$ and based on the transitions that are used. We then conclude $w \in L$, the language of the records of all accepting $u$-computation by $\mathcal{M}$. \qed