10 - Pushdown Automata

We have seen that finite automata are limited in that they are only capable of accepting regular languages. Such languages (and machines) are useful for lexical scanning, as we have seen. But for parsing, we found it useful to build a machine capable of recognizing a context-free language. Finite automata aren’t adequate for this task because they have no “memory”, beyond their states, which are finite in number.

Pushdown Automata (PDA) add a stack to the existing notion of a finite state machine, giving them an infinite (albeit simple) memory mechanism:

A Nondeterministic Pushdown Acceptor (NPDA) is defined by the septuple:

- $Q$ – a finite set of states
- $\Sigma$ – an input alphabet
- $\Gamma$ – a stack alphabet
- $\delta$ – transitions $Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow Q \times \Gamma^*$
- $s$ – initial state $\in Q$
- $\sqsubseteq$ – start stack symbol $\in \Gamma$
- $F$ – set of final states $\subseteq Q$

For example, transition $\delta(q,a,c) = (q',w)$ would mean:

1. the machine is currently in state $q$
2. consume (read in) the next symbol $a$ from the input string
3. pop the symbol at the top of the stack
4. move to state $q'$
5. push a new string onto the top of the stack
Example 1 – Construct a NPDA for the language \{ a^n b^n \; ; \; n \geq 1 \}

\begin{align*}
Q &= \{q_0, q_1, q_2, q_3\} \\
\Sigma &= \{a, b\} \\
\Gamma &= \{\emptyset, 1\} \\
z &= \emptyset \\
s &= q_0 \\
F &= \{q_3\} \\
\delta(q_0, a, \emptyset) &= \{ (q_1, 1\emptyset) \} \\
\delta(q_1, a, 1) &= \{ (q_1, 11) \} \\
\delta(q_1, b, 1) &= \{ (q_2, \lambda) \} \\
\delta(q_2, b, 1) &= \{ (q_2, \lambda) \} \\
\delta(q_2, \lambda, \emptyset) &= \{ (q_3, \lambda) \}
\end{align*}

Example 2 – Construct a NPDA for the language \{ wcw^R \; ; \; w \in \{a,b\}^+ \}

\begin{align*}
Q &= \{q_0, q_1, q_2, q_3\} \\
\Sigma &= \{a, b, c\} \\
\Gamma &= \{\emptyset, a, b, c\} \\
z &= \emptyset \\
s &= q_0 \\
F &= \{q_3\} \\
\delta(q_0, n, \emptyset) &= \{ (q_1, n\emptyset) \} \; n \in \{a,b\} \\
\delta(q_1, n, x) &= \{ (q_1, nx) \} \; n, x \in \{a,b\} \\
\delta(q_1, c, n) &= \{ (q_2, n) \} \; n \in \{a,b\} \\
\delta(q_2, n, n) &= \{ (q_2, \lambda) \} \; n \in \{a,b\} \\
\delta(q_2, \lambda, \emptyset) &= \{ (q_3, \lambda) \}
\end{align*}

NPDA's accept all strings for which there is some path to an “accept” state.

The distinction between “deterministic” and “non-deterministic” PDAs is a bit different than it is for FAs. PDAs are only non-deterministic if there is more than one choice for a given scenario. Note that in a PDA, the input string isn’t the only factor in a state transition; the symbol at the top of the stack also plays a role. By this definition, ex. 1 (above) is deterministic. Finally, unlike FAs, “deterministic” and “non-deterministic” PDAs aren’t equivalent. NPDA’s have more expressive power than DPDAs.

We will focus on NPDA’s, because they are equivalent to CFGs.
**Instantaneous Description (ID)** \(\triangleright\)

It is often useful to illustrate specific transitions in a PDA. A convenient notation for doing this uses the “\(\triangleright\)” symbol, and shows the remaining unread part of the input string, and the stack content. For example:

\[(q_1, \text{aaabb}, bx) \triangleright (q_2, \text{aabb}, yx)\]

Here, the transition is from state \(q_1\) to \(q_2\). The first “a” was read from the input string. The symbol “b” at the top of the stack was replaced with “y”.

In example 1 above, the series of transitions in accepting the input string “aabb”, using the ID notation, would be:

\[(q_0, \text{aabb}, \emptyset) \triangleright (q_1, \text{abb}, 1\emptyset) \triangleright (q_1, \text{bb}, 11\emptyset) \triangleright (q_2, \text{b}, 1\emptyset) \triangleright (q_2, \lambda, \emptyset) \triangleright (q_3, \lambda, \lambda)\]

**Building a NPDA for a CFG**

This is done most easily if the CFG is in *Greibach Normal Form* (GNF), which requires all of the CFG rules to be in the following form:

\[
A \rightarrow t B \\
\text{or} \quad A \rightarrow t \quad \text{where } t \text{ is a terminal, and } B \text{ is one or more non-terminals}
\]

Converting a CFG to GNF can sometimes be tricky, but in many cases it can be easily done by inspection. We won’t learn how to convert a CFG to GNF in general, just some simple cases.

The general approach for building the NPDA is then as follows:

1. there are three (3) states: \(q_0\), \(q_1\), and \(q_2\)
2. there is a transition from \(q_0\) to \(q_1\) which pushes \(S\) onto the stack
3. each grammar rule \(A \rightarrow t B\) then becomes a transition in the PDA:
   - the transition is from state \(q_1\) back to itself
   - consume the terminal \(t\), for stack symbol \(A\)
   - push non-terminals \(B\) (if any) onto the stack
   - for example: \(\delta(q_1, t, A) = (q_1, B)\)
4. a final transition is made to the accept state \(q_2\)
Example 3 – Build a NPDA for the CFG $S \rightarrow aSbb \mid a$

First, convert the CFG to GNF. Here, we can simply create new non-terminals for the b’s in the first rule. The resulting GNF form is:

$$
S \rightarrow aSXY \mid a \\
X \rightarrow b \\
Y \rightarrow b
$$

Using the construction described on the previous page, the resulting transition rules are as follows:

$$
\delta(q_0, \lambda, \text{\textentry}) = \{ (q_1, S \text{\textentry}) \} \\
\delta(q_1, a, S) = \{ (q_1, SXY), (q_1, \lambda) \} \\
\delta(q_1, b, X) = \{ (q_1, \lambda) \} \\
\delta(q_1, b, Y) = \{ (q_1, \lambda) \} \\
\delta(q_1, \lambda, \text{\textentry}) = \{ (q_2, \lambda) \}
$$

The series of transitions that accept the string “aabb”, are:

$$(q_0, aabb, \text{\textentry}) \xrightarrow{} (q_1, aabb, S \text{\textentry}) \xrightarrow{} (q_1, abb, SXY \text{\textentry}) \xrightarrow{} (q_1, bb, XY \text{\textentry}) \xrightarrow{} (q_1, b, Y \text{\textentry}) \xrightarrow{} (q_1, \lambda, \text{\textentry}) \xrightarrow{} (q_2, \lambda, \lambda)$$

Example 4 – Build a NPDA for the following CFG:

$$
S \rightarrow aA \\
A \rightarrow aABC \mid bB \mid a \\
B \rightarrow b \\
C \rightarrow c
$$

Since the grammar is already in GNF, we can proceed as previously:

$$
\delta(q_0, \lambda, \text{\textentry}) = \{ (q_1, S \text{\textentry}) \} \\
\delta(q_1, a, S) = \{ (q_1, A) \} \\
\delta(q_1, a, A) = \{ (q_1, ABC), (q_1, \lambda) \} \\
\delta(q_1, b, A) = \{ (q_1, B) \} \\
\delta(q_1, b, B) = \{ (q_1, \lambda) \} \\
\delta(q_1, c, C) = \{ (q_1, \lambda) \} \\
\delta(q_1, \lambda, \text{\textentry}) = \{ (q_2, \lambda) \}
$$

And an example series of transitions that accept the string “aaabc”, are:

$$(q_0, aaabc, \text{\textentry}) \xrightarrow{} (q_1, aaabc, S \text{\textentry}) \xrightarrow{} (q_1, abc, A \text{\textentry}) \xrightarrow{} (q_1, abc, ABC \text{\textentry}) \xrightarrow{} (q_1, bc, BC \text{\textentry}) \xrightarrow{} (q_1, c, C \text{\textentry}) \xrightarrow{} (q_1, \lambda, \text{\textentry}) \xrightarrow{} (q_2, \lambda, \lambda)$$