Converting many-state PDA to one-state PDA

Summary

Let’s say \( P \) is a many-state PDA with transition relation \( \delta \). We’ll create a single-state PDA \( P_1 \) with transition relation \( \delta_1 \). The stack symbols of \( P_1 \) look like \([pXq]\) where \( p \) and \( q \) are states of \( P \) and \( X \) is a stack symbol from \( P \) (of course, \( X \) might also be an input symbol). \( P_1 \) has only one state which we’ll call 1 for distinction. For each transition in \( \delta \), create one or more transitions in \( \delta_1 \). There is really only one rule, but we can break it down into three cases for clarity:

1. \( \delta \) pops a symbol (and moves from \( p \) to \( q \))
   \[
   \delta(p, a, X) = \{(q, c)\}
   \]
   \[
   \downarrow
   \]
   \[
   \delta_1(1, a, [pXq]) = \{(1, c)\}
   \]

2. \( \delta \) replaces a symbol
   \[
   \delta(p, a, X) = \{(q, Y)\}
   \]
   \[
   \downarrow
   \]
   \[
   \forall b \quad \delta_1(1, a, [pXb]) = \{(1, [qYb])\}
   \]

3. \( \delta \) pushes 2 or more symbols
   \[
   \delta(p, a, X) = \{(q, YZ)\}
   \]
   \[
   \downarrow
   \]
   \[
   \forall g, b \quad \delta_1(1, a, [pXb]) = \{(1, [qYg][gZb])\}
   \]

Note: If \( \delta \) pushes more symbols, e.g. \( WXYZ \), then we must make a guess for each symbol, and the RHS will look like: \([qWg_1][g_1Xg_2][g_2Yg_3][g_3Zb]\).

Discussion

The trick is encoding the state into the stack in \( P_1 \). To start, we always keep the current state with the topmost stack symbol. It’s on the left – so if \([pAq]\) is on the top of the stack, then we’re in state \( p \). But this is not quite enough. If we pop \([pAq]\) we must still know what state we’re in. That’s where \( q \) comes in. Before we push a new symbol onto the stack we guess what state we’ll be in when the stack returns to the same height. That guess is on the right (the \( q \) in \([pAq]\)) – we’ll verify the guess later since we can only delete \([pAq]\) if there is a transition in \( \delta \) that deletes \( A \) and moves from \( p \) to \( q \).

When building the stack we always need to maintain consistency of adjacent symbols (e.g. \([pAq][qBr]\)) so deleting \([pAq]\) actually moves us to state \( q \). Here are some example stack operations and their meanings:
1. \([pAq][qBr] \rightarrow [qBr]\)
   This means that we pop \(A\) from the stack and move from state \(p\) to \(q\).

2. \([pAq][qBr] \rightarrow [sXq][qBr]\)
   This means that we replace \(A\) with \(X\) and move to state \(s\). The symbol \([pAq]\) indicates a previous guess that the eventual deletion of the symbol in this position would take us to state \(q\), so we must maintain that guess in \([sXq]\).

3. \([pAq][qBr] \rightarrow [sXg][gYq][qBr]\)
   Replace \(A\) by \(XY\) and move to state \(s\). Here we increase the height of the stack, so we must guess the state \(g\) that \(P\) will move to when this symbol is eventually deleted. It may seem that by allowing any state \(g\) we are somehow being too permissive, but that is not the case. We can only delete \([sXg]\) if there is a transition in \(\delta\) that deletes \(X\) and moves from \(s\) to \(g\) — if we guess incorrectly we place a symbol on the stack that can never be deleted.

Example
Let \(P\) be a PDA that accepts \(\{0^n1^n|n \geq 1\}\) and has three states:

1. State \(p\) reads 0’s from the input and pushes them onto the stack. It may stay in state \(p\) or move to state \(q\).
2. State \(q\) reads a single 1 from the input and moves to state \(r\).
3. State \(r\) reads 0’s from the input and pops 0’s from the stack.

\(\delta\) looks like:

\[
\begin{align*}
\delta(p, 0, Z_0) &= \{(p, 0Z_0), (q, 0Z_0)\} \\
\delta(p, 0, 0) &= \{(p, 00), (q, 00)\} \\
\delta(q, 1, 0) &= \{(r, 0)\} \\
\delta(r, 0, 0) &= \{(r, \epsilon)\} \\
\delta(r, 0, Z_0) &= \{(r, \epsilon)\}
\end{align*}
\]
Here are the stack symbols of $P_1$:

\[
[p0p], [p0q], [p0r] \quad [p1p], [p1q], [p1r] \quad [pZ0p], [pZ0q], [pZ0r] \\
[q0p], [q0q], [q0r] \quad [q1p], [q1q], [q1r] \quad [qZ0p], [qZ0q], [qZ0r] \\
[r0p], [r0q], [r0r] \quad [r1p], [r1q], [r1r] \quad [rZ0p], [rZ0q], [rZ0r]
\]

Here are three examples of transitions in $\delta$ and the transitions in $\delta_1$ that they generate.

1. $\delta(p, 0, 0) \rightarrow (q, 00)$

\[
\delta_1(1, 0, [p0p]) = \{(1, [q0p][p0p]), (1, [q0q][q0p]), (1, [q0r][r0p])\}
\]

\[
\delta_1(1, 0, [p0q]) = \{(1, [q0p][p0q]), (1, [q0q][q0q]), (1, [q0r][r0q])\}
\]

\[
\delta_1(1, 0, [p0r]) = \{(1, [q0p][p0r]), (1, [q0q][q0r]), (1, [q0r][r0r])\}
\]

2. $\delta(q, 1, 0) \rightarrow (r, 0)$

\[
\delta_1(1, 1, [q0p]) = \{(1, [r0p])\}
\]

\[
\delta_1(1, 1, [q0q]) = \{(1, [r0q])\}
\]

\[
\delta_1(1, 1, [q0r]) = \{(1, [r0r])\}
\]

3. $\delta(r, 0, 0) \rightarrow (r, \epsilon)$

\[
\delta_1(1, 0, [r0r]) = \{(1, \epsilon)\}
\]

It may be useful to work through some example stack configurations to see how this machine accepts the string 00100. Hint. In the middle of the computation, the string 100 will remain from the input and the stack will look like:

\[
[q0r][r0r][rZ0r]
\]