CS 311: Computational Structures

James Hook and Tim Sheard

October 31, 2013

6 PDA to CFG Example

Consider the PDA:

\[
\begin{align*}
\delta(0,\epsilon,\epsilon) &= \{(1,\$)\} \quad +\$ \quad 1 \\
\delta(1,a,\epsilon) &= \{(1,\#)\} \quad +\# \quad 2 \\
\delta(1,b,\#) &= \{(2,\epsilon)\} \quad -\# \quad 3 \\
\delta(2,b,\#) &= \{(2,\epsilon)\} \quad -\# \quad 4 \\
\delta(2,\epsilon,\$) &= \{(3,\epsilon)\} \quad -\$ \quad 5
\end{align*}
\]

In this summary I have indicated if a rule is a “push” of \(t\) (+t) or a “pop” of \(t\) (−t). I have also numbered each line in the definition of \(\delta\) for reference.

Recall that the construction introduces rules of the form:

\[A_{pq} \rightarrow aA_{rs}b\]

when there is a stack symbol \(t\) such that:

\[(r,t) \in \delta(p,a,\epsilon)\]
\[(q,\epsilon) \in \delta(s,b,t)\]

Note that this is exactly when the transition from \(p\) to \(r\) is labeled +\(t\) and the transition from \(s\) to \(q\) is labeled −\(t\).

Applying this rule to all of \(\delta\) yields 3 instances. They are:

\[
\begin{align*}
A_{03} &\rightarrow A_{12} \quad + - \$ \quad 1,5 \\
A_{12} &\rightarrow aA_{11}b \quad + - \# \quad 2,3 \\
& \mid aA_{12}b \quad + - \# \quad 2,4
\end{align*}
\]

Here I have annotated each rule with what symbol is being pushed and popped (+−\(t\)) and which lines in the definition of \(\delta\) are used in the construction.

The grammar is completed by using one instance of the construction that introduces null productions:

\[A_{11} \rightarrow \epsilon\]

It is, of course, safe to add all other null productions, but no other null productions contribute to the generation of any strings in the language.