Transformation of an NPDA to CFG

Jay Bagga

1 NPDA to CFG

By now you are familiar with context-free grammars and nondeterministic pushdown automata. They are equivalent in the sense that both generate the class of context-free languages. In this exercise, we see an algorithm to convert a given NPDA to an equivalent CFG. For each transition of the given NPDA, we want to create a set of productions that produce the same behavior as the transition.

We shall start with the following NPDA. See Figure 1. What context-free language does this NPDA accept? Enter this NPDA into JFLAP.

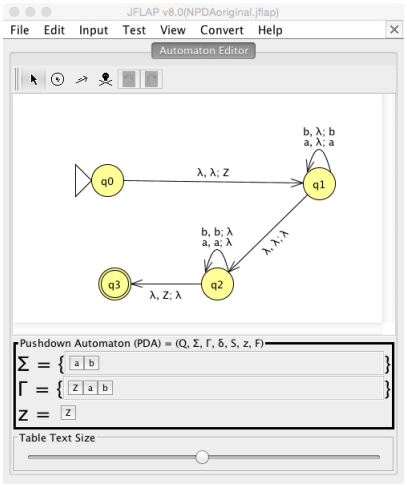

Figure 1: An Example NPDA

The conversion algorithm that JFLAP uses requires that the NPDA should be in a certain format. In particular, each transition must pop exactly one symbol and push exactly zero or two symbols. Select Convert: Grammar. You see a message that the NPDA in Figure 1 is not in the required format. The message also lists the transitions that are not in the required format. We must modify the NPDA. We do so by deleting those transitions and introducing new ones. The modified NPDA is shown in Figure 2. Study this carefully. Do the new transitions now satisfy the required format? Also, are they equivalent to the transitions in Figure 1?

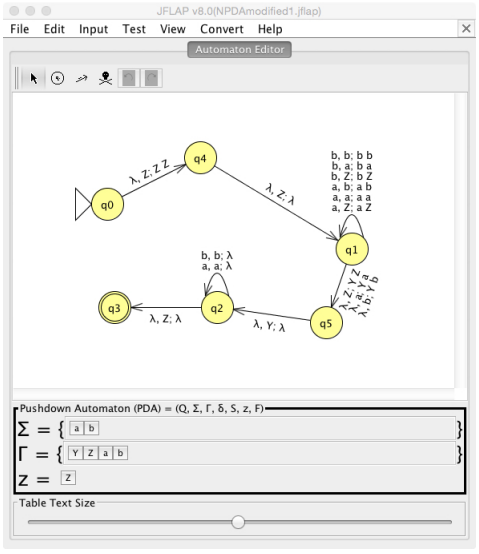

Figure 2: Modified NPDA

The basic idea of converting a transition into a set of productions is as follows. For each pair of states q_i and q_j we consider all strings that can take the NPDA from q_i and q_j. We introduce variables of the form q_iAq_j, where A is a stack symbol. This variable is to be interpreted as follows: "if when moving along a path from q_i and q_j, the stack is exactly the same except that A is removed from the stack." Then a production of the form $q_iAq_j \rightarrow a(q_iAq_k)(q_kAq_j)$ says that in going from q_i and q_j, input a was read and the stack was the same except A was popped, and this path went to a q_k and other symbols B and C were pushed along this path and were popped off.

Choose Convert: Grammar again. Since the NPDA is now in the correct form, the algorithm produces the grammar as shown in Figure 3. A large number of productions are generated. Study the productions and assign meanings to them as above. Practice by repeating this procedure for another NPDA of your choice.

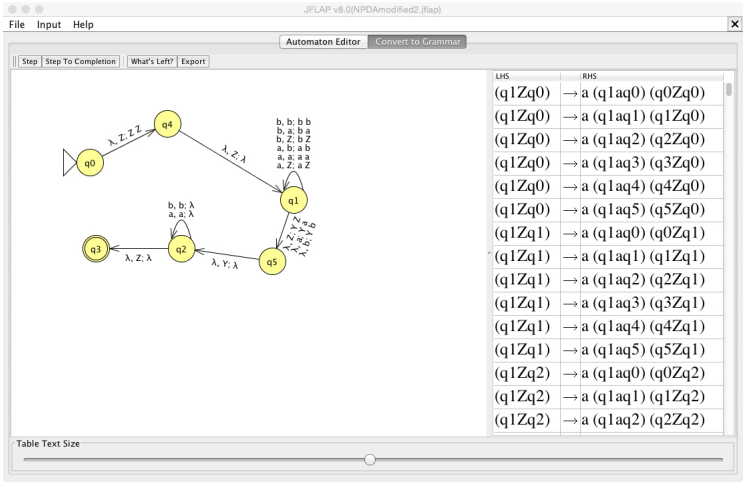

Figure 3: NPDA to CFG

2 References

1. Introduction to the Theory of Computation (Third Edition), Michael Sipser. Cengage Learning. 2013.

2. JFLAP - An Interactive Formal Languages and Automata Package, Susan H. Rodger and Thomas W Finley. Jones and Bartlett Publishers. 2006