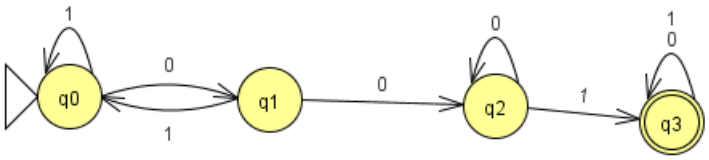
Converting a DFA to a CFG

General idea of the algorithm

The DFAs and NFAs that you have studied prior to being introduced to grammars accept languages that are called Regular Languages. Grammars at least initially seem to be a different system and an interesting question to answer is whether any Regular language can be produced by some grammar. The answer is yes!

If a language is regular, there is a DFA for the language and what is demonstrated here is how you can take a DFA and from it create a grammar that produces the same language.

Here is a DFA that we will convert to a CFG. It just accepts any string that contains the substring 001 in it.

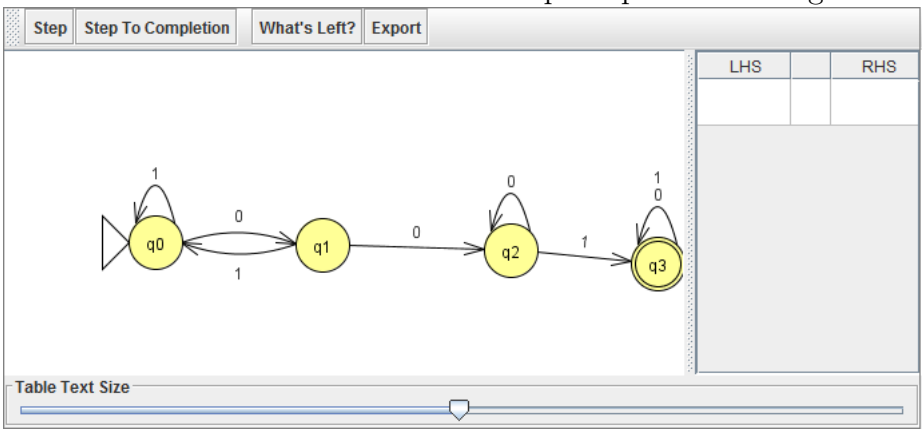


For the conversion to CFG the basic theory is as follows

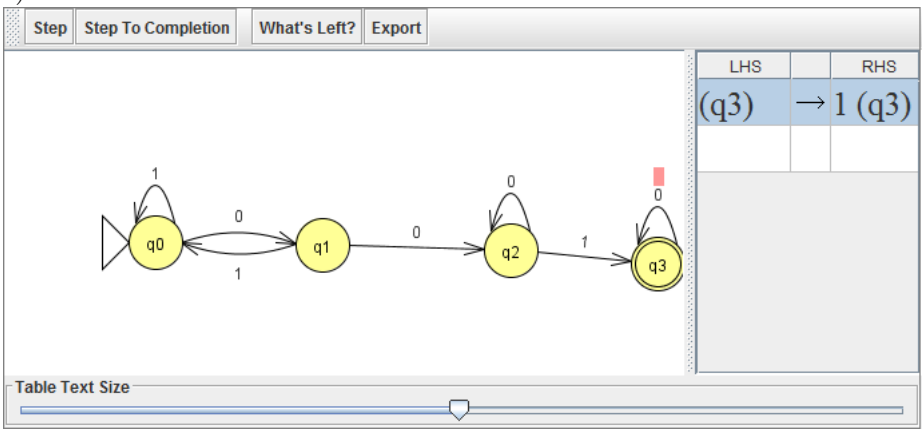
1. Make a variable for every state of the DFA. In JFLAP for the state qi we make the variable (qi).
2. Add the rule $(qi) \to a(qj)$ if $\delta(qi, a) = qj$.
3. Add the rule $(qk) \to \varepsilon$ if qk happens to be an accept state.

Stepping through the conversion process in JFLAP

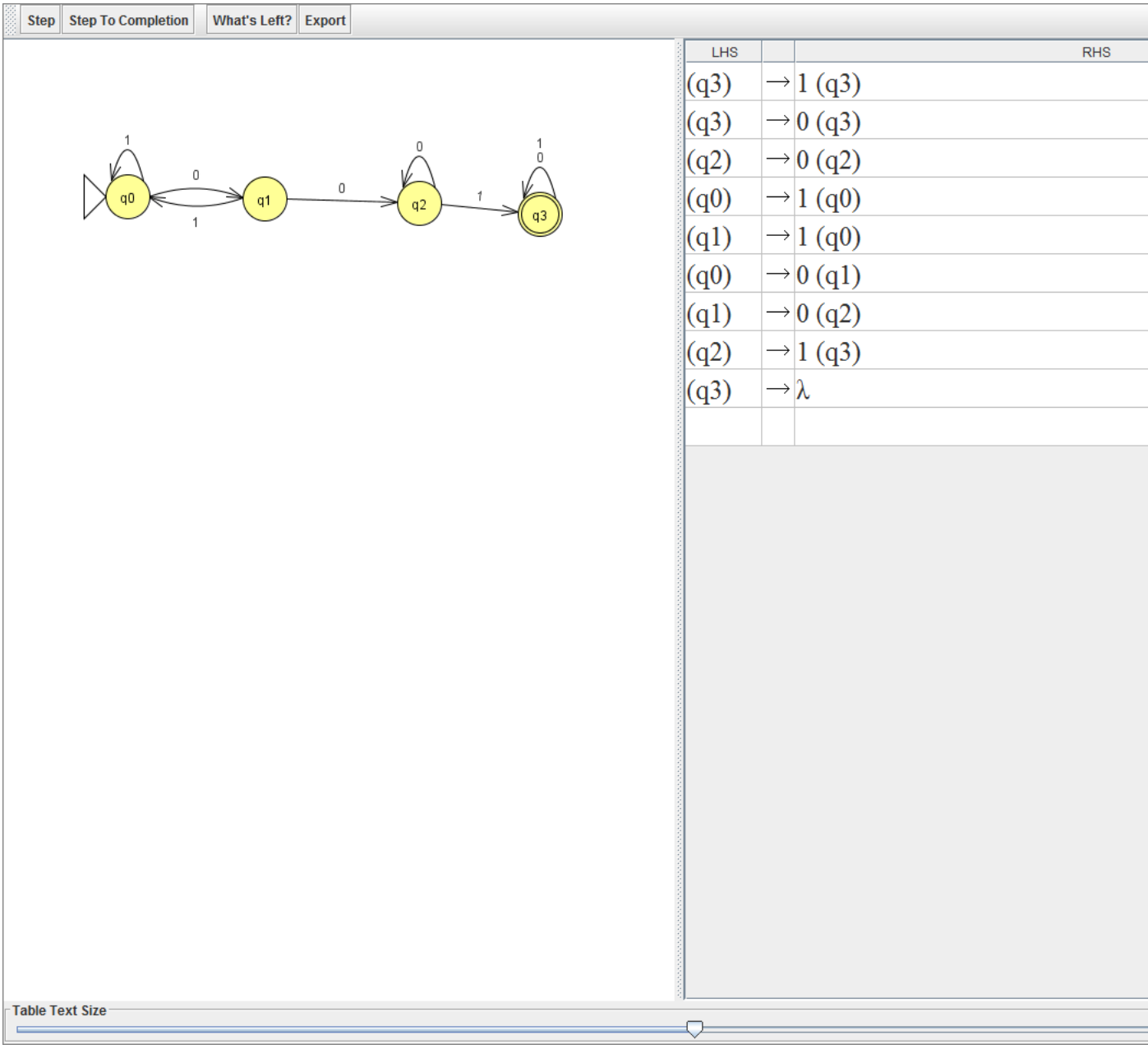
Click on Convert to Grammar. That will open up the following window



Now JFLAP can create the rules one by one by stepping through each transition. For instance when we want to convert the transition $\delta(q3, 1) = q3$ that creates the rule $(q3) \rightarrow 1(q3)$.



The other rules follow the same idea. The only one that is different is the rule that has $q3 \rightarrow \varepsilon$ which is there because q3 is a final state.


Step
Step To Completion
What's Left?
Export
q0
q1
q2
q3
0
1
0
1
0
1
0
1
LHS
RHS
(q3) → 1 (q3)
(q3) → 0 (q3)
(q2) → 0 (q2)
(q0) → 1 (q0)
(q1) → 1 (q0)
(q0) → 0 (q1)
(q1) → 0 (q2)
(q2) → 1 (q3)
(q3) → λ
Table Text Size