Practical No. 3

Title: JFLAP Tool

Theory:

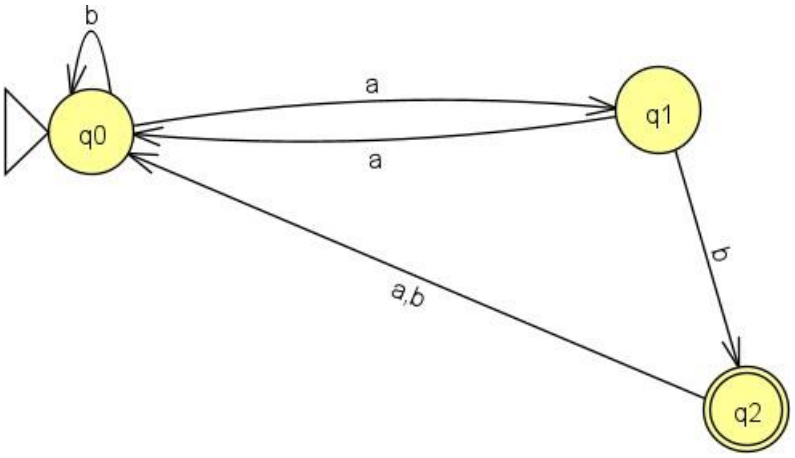
JFLAP is a package of graphical tools which can be used as an aid in learning the basic concepts of Formal Languages and Automata Theory. The original program (FLAP) was written in C/C++ for X-window based systems. Due to its success as a visual aid in introductory courses in Theoretical Computer Science, the Java version of FLAP was created, which should work on virtually any system.

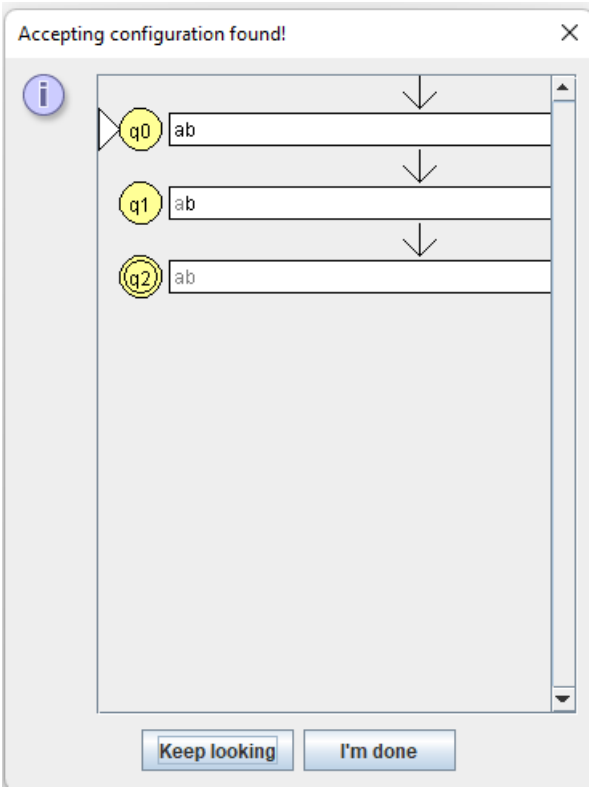
Using JFLAP, one should be able to design and simulate several variations of finite automata (FA), pushdown automata (PDA), one-tape Turing machines (TM) and two-tape Turing machines (TTM). The user draws the transition diagram of the desired automaton and, once the picture is complete, the user enters an input string and then "runs" the automaton, being able to view all the generated configurations. In addition, JFLAP can handle grammars (GRM) and Regular Expressions (REX).

Several conversions from one representation to another are supported. The conversions are nondeterministic finite automaton (NFA) to deterministic finite automaton (DFA), DFA to minimum state DFA, NFA to regular grammar, NFA to Regular Expression, regular grammar to NFA, Regular Expression to NFA, nondeterministic pushdown automaton (NPDA) to context-free grammar (CFG), and three algorithms for CFG to NPDA. Two of the CFG to NPDA conversions are useful in studying LL and LR parsing.

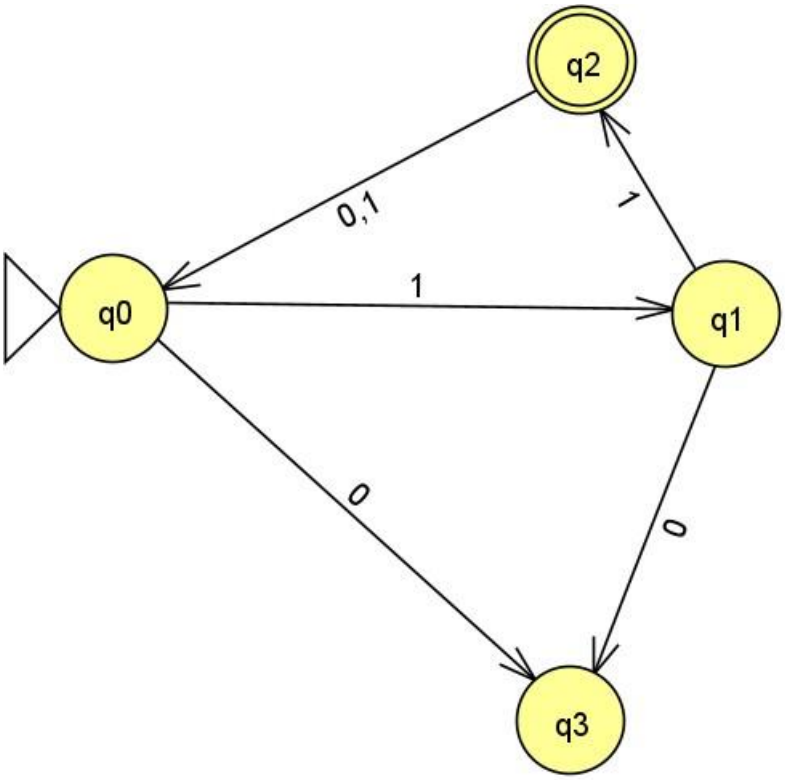
A finite automaton is the first type of representation for a regular language that we will examine. In this chapter we will construct a deterministic finite automaton (DFA) in JFLAP, illustrate several methods of simulating input on that automaton, discuss nondeterministic finite automata (NFAs) in JFLAP, and present simple analyses that JFLAP may apply to automata

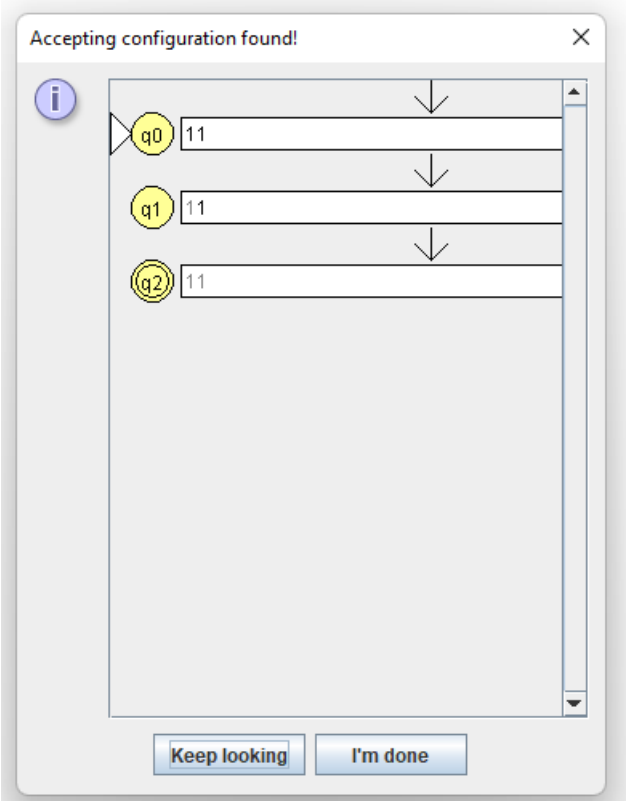
1. Construct DFA accepting all string ∑={a,b} ending with substring ab



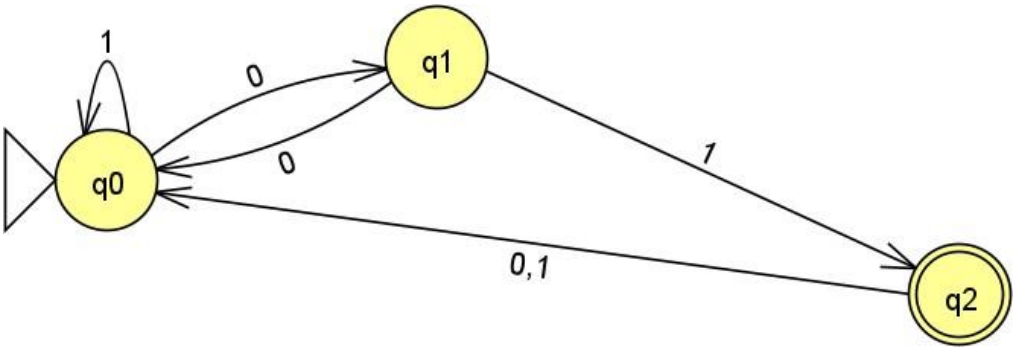


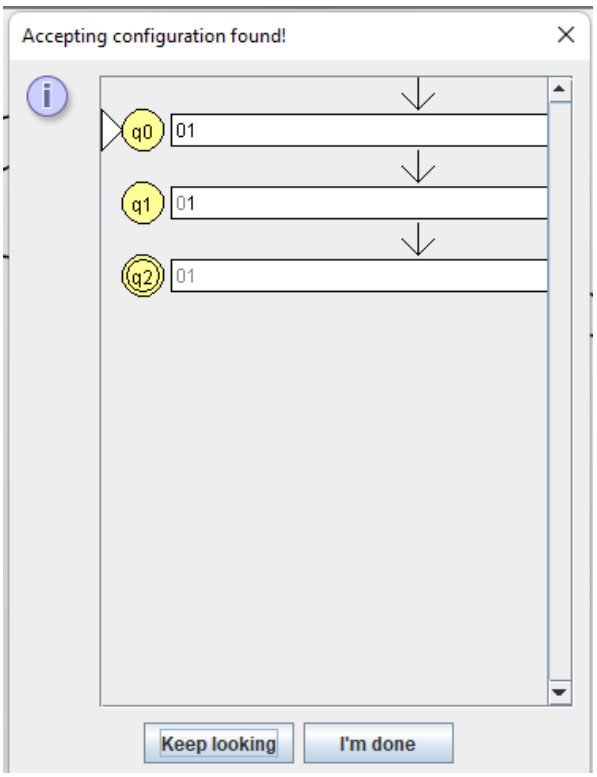
2. Construct DFA accepting all string ∑={0,1} ending with substring 11



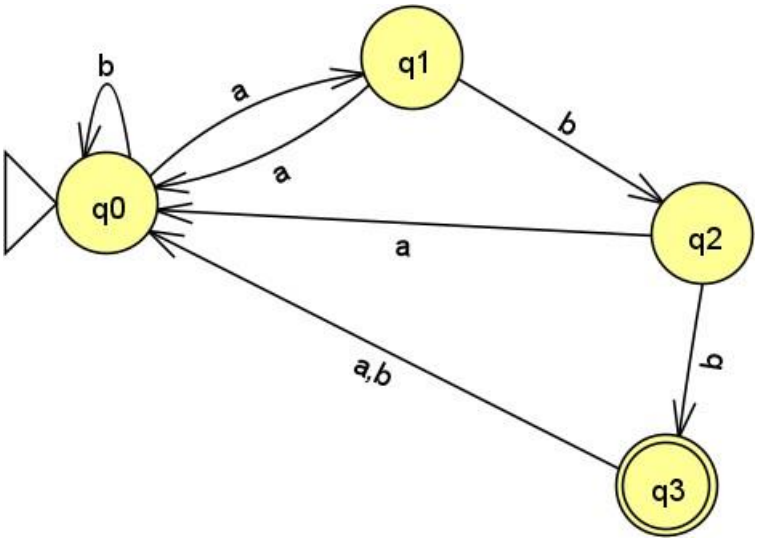


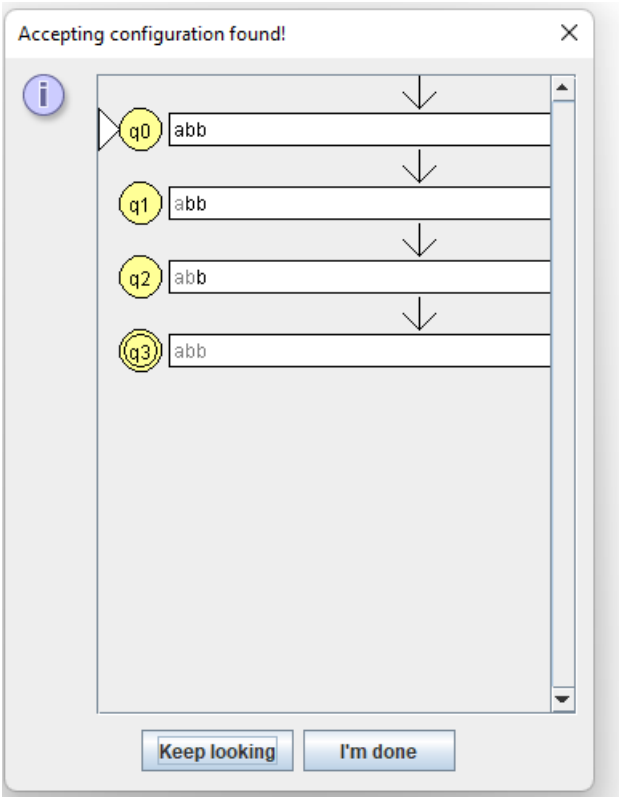
3. Construct DFA accepting all string ∑={0,1} ending with substring 01



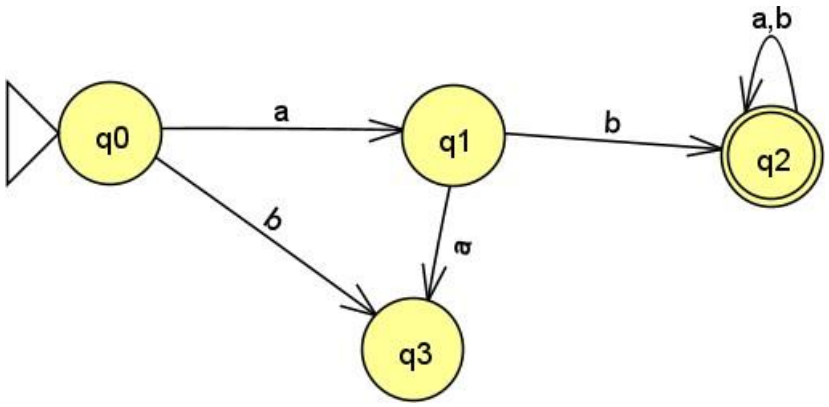


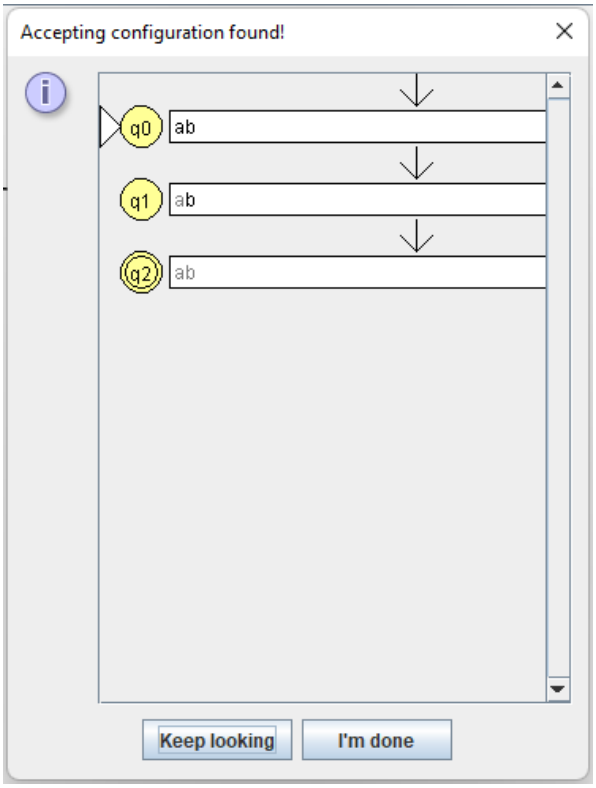
4. Construct DFA accepting all string ∑={a,b} ending with substring abb



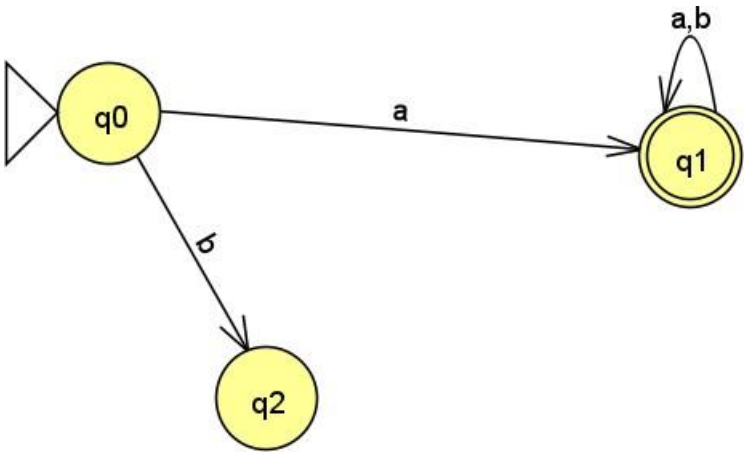


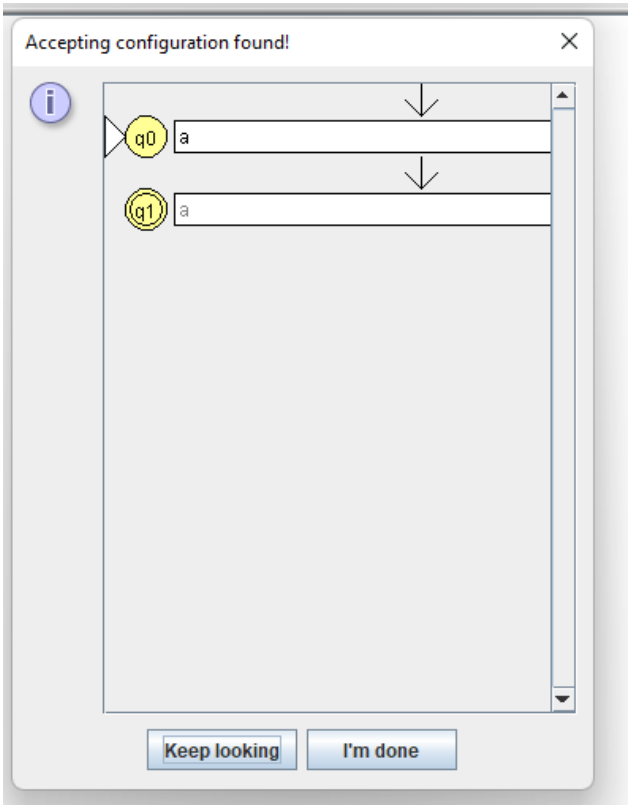
5. Construct DFA accepting all string ∑={a,b} starting with substring ab





6. Construct DFA accepting all string ∑={a,b} starting with substring a





7. Construct DFA accepting all string ∑={a,b} starting with substring aa or bb