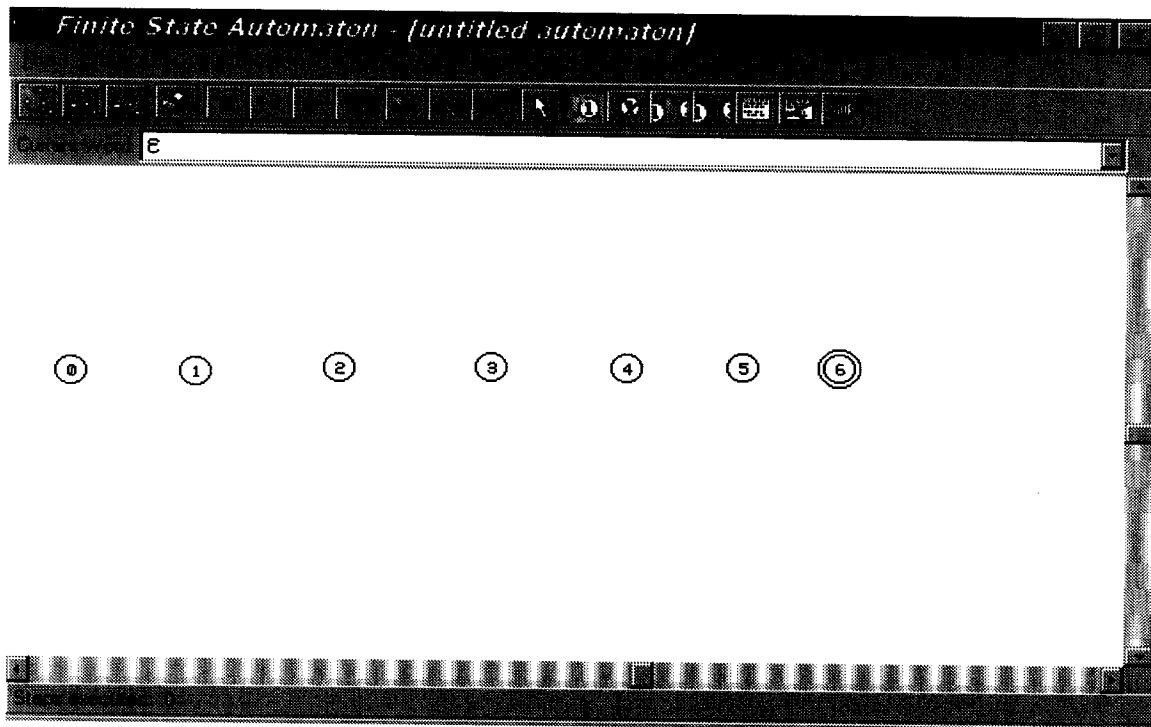


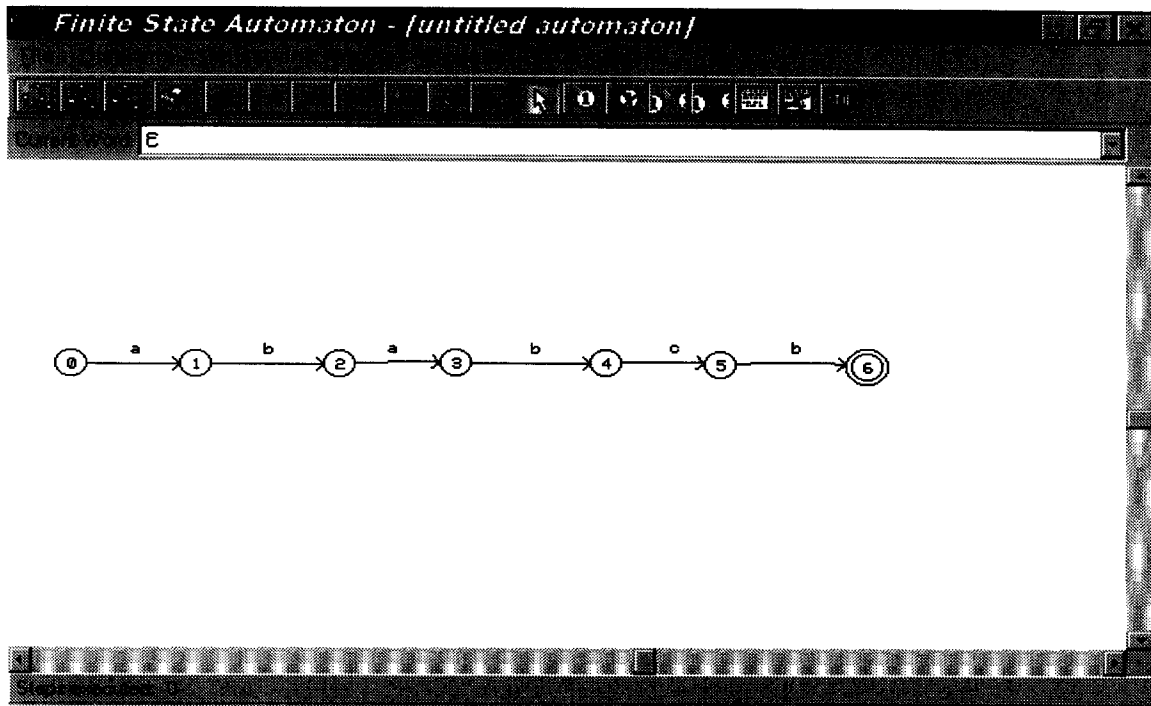
**Hands-On Tutorial for *Deus ex machina* (Designed by Nicolae O. Savoiu)**  
**Simulation Software Accompanying**  
**R. Gregory Taylor, *Models of Computation and Formal Languages***  
**(Oxford University Press, 1998)**

We restrict our attention to a single simulation, namely, that for finite-state automata and guide the user through the creation of a machine that accepts all and only the input strings over alphabet  $\Sigma = \{a, b, c\}$  that happen to contain an occurrence of the substring  $w = ababc b$ . (Thus this example is related to the general topic of *string matching*.)

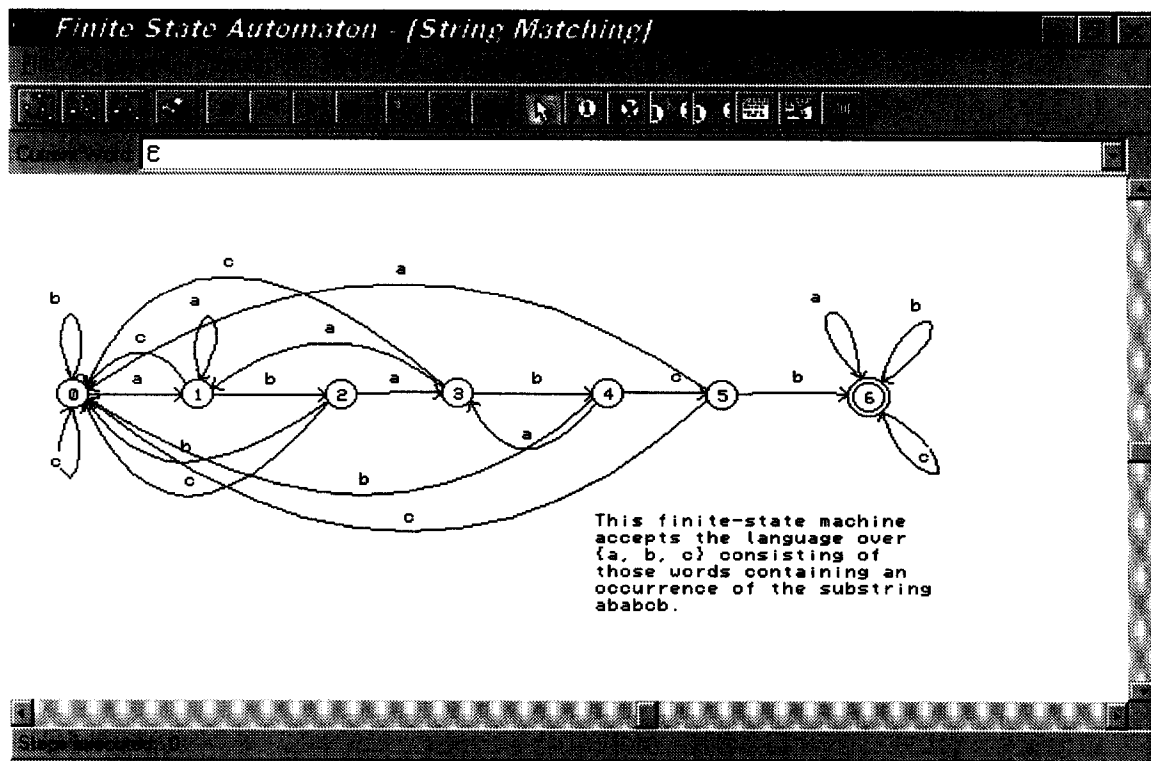
To begin, open the icon **FSA** within the FSA group. This will cause the FSA window to open. From the File menu, choose **Machine/New**. First choose **Alphabet/Set Input Alphabet** and highlight characters *a*, *b*, and *c* within the Input Alphabet window and click **OK**. We shall need a total of seven states—one more than  $length(w)$ . Click on **Diagram/Add Node** and position states 0 through 6 in a horizontal arrangement starting on the left using the mouse pointer and clicking once for each state. To make state 6 a *terminal state*, right click on that state and check **Terminal node**. The screen should now appear as in the diagram below.



Next, let's add a "success" instruction for each character within  $w$ . For example, when in initial state 0 reading character *a*, our machine will proceed into state 1. Select **Diagram/Add Transition**. Then click first on source state 0, then on target state 1, and finally click once in the space between the two states. This will cause a directed arc from state 0 to state 1 to appear. Set the Scan Symbol to *a* within the Transition Properties window and click **OK**. (Use the mouse to straighten the arc and center its label if necessary.) Doing this for each of the other characters of  $w$  will result in the state diagram shown below. (Deleting an arc is accomplished from the **Diagram** menu. Also, to change an arc label, right click on it.)



These are the so-called “success” arcs. In addition, we require two “fail” arcs from each of states 0 through 5 as well as three self-loops at terminal state 6. Add these now in the same manner, using the diagram below as guide. To produce the *b*-self-loop at state 0, select Diagram/Add Transition and click *twice* on state 0.



Note that we have set the title in the title bar to “String Matching”. Use File/Set Title for this. Also, the textbox in the lower right was added using Diagram/Add Text. It is a good time to save our diagram. Choose File/Save and call this machine `strmatch.fsa`.

Finally, let’s run the machine on the text `abcbabcbababcbabbabaabc`, which does happen two contain an occurrence of the search string. To do this, choose Words/Add Word, type in this text in the window that appears, and click OK. To run, select Simulation/Run. The machine will halt in terminal state 6 and, by definition, accept this word. As a second example, choose Words/Add Word and enter a word in which `ababcb` does not occur. Choose Simulation/Reset and then Simulation/Run. This time the machine will halt in a nonterminal state.

**Software.** The software was designed by Nicolae O. Savoiu and is available on the Internet. To download this software, point your WWW browser to <http://www.ics.uci.edu/~savoiu/dem> and follow the on-line instructions. Help files are available at the same site.

**On-Line Resources.** The website for the textbook is found at

<http://starbase.cs.trincoll.edu/~rtaylor/thcomp/>

wherein instructor’s guide, solutions manual, errata list, and so forth may be found.