

Writing assignment 3

Assigned Tuesday, February 17; due Thursday, March 5

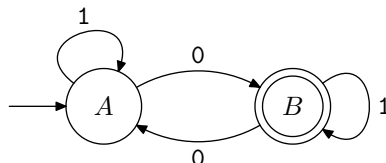
Choose one of the topics below and discuss it. Remember to follow the quality expectations for submitted work given in the course information sheet. If you draw a picture as part of your paper, be sure to include a sentence or two explaining what the picture is and what information it shows.

Please feel free to ask me questions if there is something you don't understand. I would much rather have you ask me questions and turn in a paper that shows your understanding than have you turn in a paper that doesn't make sense. Try to explain things in such a way that someone else in this class could read your paper and understand it without needing to read outside sources.

1. **Wordplay.** Read the excerpt about word graphs from *Making the Alphabet Dance* by Ross Eckler (St. Martin's Press, 1996). Keep in mind while you read that Mr. Eckler uses different terminology for graphs than our textbook does; for example, he uses the terms *point* and *line* instead of *vertex* and *edge*. Choose at least three of the following and discuss them:
 - (a) The second paragraph, marked (\star), describes a particular characterization of word graphs and how this characterization can be used to determine whether a word graph has a particular property. What is this paragraph talking about? Restate the main ideas of this paragraph in your own words, using terms and concepts we discussed in Chapter 6.
 - (b) A graph which cannot be drawn on a piece of paper without edges crossing is usually called a *nonplanar* graph. Mr. Eckler uses the term *eodermdrome* to refer to a word that produces such a graph. Where does this word come from, and why does it mean what it does? (Hint: Draw the word graph for it.)
 - (c) Choose a word from the third column of Figure 24b, Eodermdromes That Reduce to $K(3,3)$, and draw its word graph. Explain what makes this word an eodermdrome.
 - (d) This article claims that *supercalifragilisticexpialidocious* is *not* an eodermdrome. Draw a word graph of *supercalifragilisticexpialidocious* without crossing edges to show this claim is true. (The clarity of your graph is important. You will probably have to draw it experimentally a few times to figure out how to keep edges from crossing. In the graph you hand in, try to position the letters in such a way that most, if not all, of the edges are straight lines.)
 - (e) Choose a word that is not an eodermdrome and draw a word molecule for it. (Draw carefully; a compass will probably be helpful. I will take into account the "interestingness" of the word molecule when grading; choosing a very short word, like *cat*, or a word with no repeated letters, like *farmhouse*, is not very interesting.)
 - (f) Invent (or research) some other kind of wordplay that relates somehow to graphs, tilings, polyominoes, or something else we've talked about in this course. Describe it and give some examples.

2. **Deterministic finite automata.** The basic idea of a *deterministic finite automaton* (abbreviated DFA, and sometimes called a *deterministic finite state machine*) is not as complicated as the name makes it sound. Essentially a DFA is a model for a simple kind of computer that can answer a yes/no question about an input *string*, i.e., a sequence of characters (letters, numbers, or other symbols). The set of allowable symbols in the input string is called the *alphabet* for the DFA, even if it contains symbols other than letters. Working with large alphabets (such as the 26 letters of the English alphabet) makes things more complicated, so often we will use the alphabet that consists only of the digits 0 and 1, for simplicity.

Here is an example of a DFA that can be used to answer the question, "Does the input string contain an odd number of zeroes?"



This picture is interpreted as follows. There are two *states*, labeled *A* and *B* in this picture. The state *A* is the *start state*, as indicated by the arrow pointing to *A* from nowhere. The state *B* is the *accept state*, as indicated by the fact that it is drawn with a double circle. (In general a DFA may have several accept states; this DFA has just one. Also, it is possible for a state to be both the start state and an accept state.) The directed edges between the states are labeled with symbols from the alphabet for the DFA, which in this case consists of the digits 0 and 1. Each state has exactly one outgoing edge for each symbol in the alphabet (note that the edges labeled 1 in this picture happen to be loops).

To use a DFA, we perform the following steps:

1. Start at the start state.
2. Use the first symbol in the string to choose an outgoing edge from the start state. Follow this edge to a new state.
3. Have you used all of the symbols in the string? If so, decide your answer and stop: If you are in an accept state, the answer is “yes”; otherwise, the answer is “no.”
4. If there are more symbols in the string, use the next symbol to choose an outgoing edge from the current state, and follow this edge to a new state. Go back to step 3.

For example, suppose we want to use the DFA above to decide whether the string 010110 has an odd number of zeroes. We start in the start state, which in this case is state *A*. We use the first symbol in the string, which is 0, to choose the outgoing edge from state *A* labeled with 0; following this edge takes us to state *B*. Then we use the second symbol in the string, which is 1, to choose the outgoing edge from state *B* labeled with 1, which is a loop, so we stay in state *B*. Continuing in this manner, we find that the third symbol, 0, takes us back to state *A*; the fourth symbol, 1, keeps us in state *A*; the fifth symbol, 1, also keeps us in state *A*; and the sixth symbol, 0, takes us back to state *B*. We have now reached the end of the string. We find ourselves in state *B*, which is an accept state (since it is drawn with a double circle), so the answer to our question (“Does the string 010110 have an odd number of zeroes?”) is yes.

Conceptually, the states of a DFA are used to keep track of some kind of information. As the DFA processes an input string character by character, it moves from state to state according to what it has seen so far. In this example, the state *A* represents the situation that so far we have seen an even number of zeroes, and the state *B* represents the situation that so far we have seen an odd number of zeroes. Therefore, if we reach the end of the string and we have seen an odd number of zeroes (i.e., if we are in state *B*), the answer to our question is yes; otherwise, if we have seen an even number of zeroes (i.e., if we are in state *A*), the answer to our question is no.

In general, a DFA can have many states, and one or more of them can be designated as accept states. There must be exactly one state designated as the start state (the start state may also be an accept state), and every state must have exactly one outgoing edge for each symbol in the alphabet (some of these edges may be loops).

Deterministic finite automata are useful because there is a straightforward method for automatically designing one that will recognize a given pattern, and once designed a DFA is quick to use. There is a computer program called **grep** that is used to search a file for lines matching a specified pattern; for example, to find all lines in the file whose tenth letter is an A, or to find all lines that contain the word *output* somewhere. Essentially, **grep** works by designing a DFA for the specified pattern and then using it for each line of the file to decide whether the line matches the pattern.

Discuss the following:

- (a) Use the DFA shown above to decide whether the strings 101000 and 01110100 have an odd number of zeroes. Explain the steps you are following.
- (b) How can the DFA above be changed so that it answers the question, “Does the input string contain an even number of zeroes?”
- (c) Design a DFA that answers the question, “Does the input string end with a 1?” Assume that the alphabet consists only of the digits 0 and 1. Explain how your DFA works. In particular, what do the states represent? Give a couple of examples showing your DFA in action (at least one for which the answer is yes, and at least one for which the answer is no), and explain each step.

3. **Dijkstra's algorithm.** Research Dijkstra's algorithm, which can be used to find the shortest distance between two vertices in a weighted graph (in which the weights represent distances). Describe how the algorithm works, and give an example, explaining each step. (Make up your own example; don't take the example itself from somewhere else.) Why is a shortest-distance algorithm such as Dijkstra's algorithm useful in a practical scenario?
4. **Traffic lights.** Research the history of traffic lights and signals. Describe something interesting about them, such as early experiments and problems that had to be overcome, variations around the world, or methods of coordinating related signals. Explain how some of the ideas we have talked about in class can be used to help time traffic lights in order to keep traffic moving efficiently. Ideas from both graph theory and scheduling might be relevant. Can one-way streets be beneficial? Can they be detrimental? How do differing speed limits affect things? Make your ideas concrete by relating them to traffic lights in downtown Lincoln.
5. **Movie production.** Research the process of making a movie or a television show. Relate this process to the scheduling concepts we discussed in Chapter 7. What tasks need to be done? What order requirements are there between these tasks? In what order are the tasks actually done? Why are they done in this order? What things are done at the same time? What things cannot be done at the same time, and why? What is a typical finishing time for the production of a movie or TV show? Feel free to include other interesting or pertinent information, and feel free to draw a picture (such as an order-requirement digraph) if it will help to explain things.