600.405 — Finite-State Methods in NLP
Assignment 1: Getting Started

Prof. J. Eisner — Fall 2000
Handed out (behind schedule): Fri., Nov. 10, 2000
Due: Preferably at the Tue., Nov. 14 lecture, for your good;
but will accept until noon on Friday, Nov. 17
(to NEB 224 mailbox or jason@cs.jhu.edu).

A number of important ideas and nuances will be introduced through the homework exercises rather than in lectures. So even if you're just sitting in, I encourage you to consider and discuss the theoretical questions, and to try some of the practical exercises, since they will help you develop your intuitions.

For enrolled students: As stated on the course web page, you are encouraged to work in pairs on the homework, *provided* that each of you makes a real effort on each problem; that you indicate who you worked with; and you write up your work separately.

You are welcome to send me questions and even to use the class mailing list for discussion, within reason.

The $\star$ symbol denotes a difficult problem. It may be iterated, i.e., the difficulty level is indicated as an element of $\star^*$. Aren't regexps useful? :-)

1. Recall that a complete deterministic finite-state automaton (complete DFA) is specified as a tuple $(\Sigma, Q, i, F, \delta)$, where

   - Σ is the **alphabet**;
   - Q is the finite set of **states**;
   - $q_0 \in Q$ is the **initial state**;
   - $F \subseteq Q$ is the set of **final states**;
   - $\delta : Q \times \Sigma \rightarrow \Sigma$ is the **transition function**. For example, $\delta(q_1, a) = q_2$ means that the a arc from state q_1 goes to state q_2.

(a) The term **complete** means that from each state, there exist $|\Sigma|$ different arcs, one to read each symbol in Σ. How would you have to change the definition above so as to allow incomplete automata with fewer than $|\Sigma|$ outgoing arcs per state?

(b) How would you have to change the definition to allow nondeterminism—i.e., multiple arcs leaving the same state and reading the same symbol?

(c) How would you change the definition to allow ϵ-transitions, i.e., transitions that read the empty string?

(d) How would you change the definition so as to associate an output string or weight with each arc?

(e) For a fixed alphabet of size $k = |\Sigma|$, how many distinct complete DFAs are there with the state set $Q = \{q_1, q_2, \ldots q_n\}$? (You may count unequal automata as distinct even if they are isomorphic.)

(f) ⋆Some of the automata in the previous question are equivalent in that they accept the same **language** (set of strings). Assume $k \geq 2$. Asymptotically (i.e., for large n), about how many *different* languages are accepted by such automata? Can you get reasonably tight lower and upper bounds? Give your answer in asymptotic ("big-Oh") form: $O(f(n))$ or $e^{O(f(n))}$, where k appears as a constant in $f(n)$.[1] (*Note:* You may want to review the simplest minimization algorithm[2] or at least try problem 6 first.)

(g) ⋆⋆ Extra credit: Same question for $k = 1$.

2. Learn the three software packages, in order, by following the instructions at `http://www.cs.jhu.edu/~jason/405/software.html`. What was the most interesting thing you learned or realized about finite-state methods during this exercise? Also, what do you think of each package—what's good and what's annoying?

   (a) FSA Utilities

   (b) xfst

   (c) fsm + lextools

[1] Note: $z = e^{O(f(n))}$ means that $\log z = O(f(n))$. This notation is useful because $e^{2x+5} \neq O(e^x)$ but $e^{2x+5} = e^{O(x)}$.

[2] See Hopcroft & Ullman §3.4. There is also a nice concise illustrated explanation at `http://www.cs.engr.uky.edu/~lewis/essays/compilers/min-fa.html`.

For the remaining problems on this assignment, you may use the tool of your choice. (But you can do most of the work this week without any software at all.)

3. Your questionnaire asked:

   > Write a regular expression that accepts only binary numbers that are divisible by 4.

   Here are some of the answers from the class. For each answer, say whether it is a correct answer. If not, give a (short) string on which the regular expression does the wrong thing. You may use the software tools to help you.

   (a) $1(0+1)^*00$

   (b) $(0+1)^*100$

   (c) $(0+1)^*00+0$

   (d) ^{*00}

   (e) $(1^*0^*)^*00$

4. Draw a finite-state automaton that accepts the above language. If it is not deterministic, also draw a deterministic (and preferably minimal) version. Produce at least one of these drawings by using the software tools.

5. Your questionnaire asked:

   > A binary number is divisible by 3 iff the number of 1's in even positions = the number of 1's in odd positions (mod 3). For example, $1010111 = 87 = 29 \cdot 3$ has four 1's in even positions and one 1 in an odd position. Draw a finite-state machine that accepts only binary numbers that are divisible by 3.

   Here are some of the answers from the class. For each answer, say whether it is a correct answer. If not, give a (short) string on which the machine does the wrong thing.

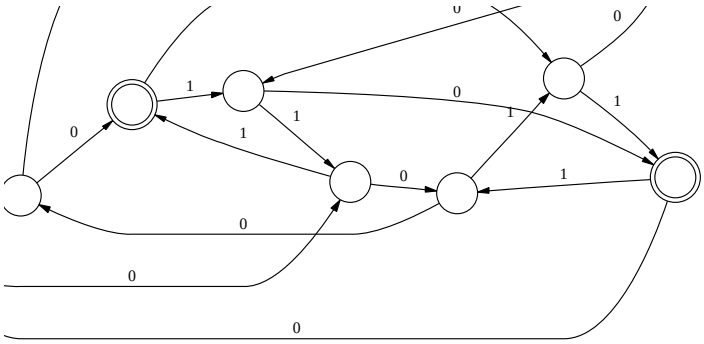
   (a)

(b)

(c) _____

(d)

(e)



6. (a) The *minimal* DFA for problem 5 is not necessarily shown above; what is it? (My first and second guesses accepted the right language but weren't minimal!) You may use the Myhill-Nerode theorem to help you construct it and prove that it is minimal.[3] If you have trouble seeing the answer or want to check your work, you may use the software tools to help you (e.g., to minimize or check an automaton).

   (b) Now, for each state in your DFA, succinctly describe the class of prefixes on which the DFA reaches that state. Does your description imply that the DFA correctly tests divisibility-by-3? Can the correctness of your description be proved by induction, as desired?

   (c) $\star$ In general, divisibility by k in base b can be decided by a DFA. Can you say anything about how to construct the minimal DFA to perform this task, and how many states it will have?

7. (a) Write a finite-state transducer that deterministically reads a binary number n *from right to left* (i.e., least significant bit first) and outputs (only) the binary representation of $n + 1$, also from right to left. Test it using software.

[3] Given an arbitrary language L, two strings u and v are said to be **L-indistinguishable** if $(\forall x \in \Sigma^*) ux \in L \Leftrightarrow vx \in L$. Only then could a DFA accepting L correctly reach the same state on both u and v. Myhill-Nerode says that L is regular iff L-indistinguishability partitions Σ^* into *finitely* many equivalence classes. If so, the *minimal* DFA for L has one state per equivalence class; it reaches that state when reading any member of the class. Again, see Hopcroft & Ullman §3.4.

(b) One way to solve the above is to write and compile an appropriate regular expression. Do so and test using software.

(c) Reverse your transducer (i.e., reverse the direction of each arc, or simply reverse the regular expression and recompile) so that it reads and writes binary numbers from left to right. Can this transducer be determinized? (Try it in software!) Why or why not? If it is nondeterministic, why doesn't it have multiple outputs per input?

(d) Invert your transducer (i.e., exchange the input and output labels on each arc). What relation does the resulting transducer implement? What happens on a zero input?

(e) Do you think that base-b multiplication by an arbitrary fixed k can be implemented with a finite-state transducer? Does right-to-left vs. left-to-right matter?