FSM Toolkit Exercises

M. Mohri, F. Pereira, M. Riley

December 17, 1996

The following problems are designed to familiarize you with the FSM tools and methods. You should use the tools throughout to solve these problems and to test your solutions.

1. Given the alphabet $L = \{a, b, \ldots z, A, B, \ldots Z, \langle space \rangle\}$, create an automaton that:

    (a) accepts a letter in L (including space).

    (b) accepts a single space.

    (c) accepts a capitalized word (where a word is a string of letters in L excluding space and a capitalized word has its initial letter uppercase and remaining letters lowercase).

    (d) accepts a word containing the letter a.

2. Using the automata in Problem 1 as the building blocks, use appropriate FSM operations on them to create an automaton that:

    (a) accepts zero or more capitalized words followed by spaces.

    (b) accepts a word beginning or ending in a capitalized letter.

    (c) accepts a word that is capitalized and contains the letter a.

    (d) accepts a word that is capitalized or does not contain an a.

    (e) accepts a word that is capitalized or does not contains an a without using `fsmunion`.

3. Epsilon-remove, determinize, and minimize each of the automata in Problem 2. Give the number of states and arcs before and after these operations.

4. Consider the automaton:

0	1	1
0	2	2
1	1	1
2		
3	4	4
4	3	3
4		

(a) How many states can be reached from the initial state?

(b) How many states can reach a final state?

(c) Compile this automaton and then remove all useless states.

5. Given the alphabet $\{a, b, \ldots, z, \langle space \rangle\}$,

   (a) create a transducer that implements the *rot13* cipher – $a \rightarrow n, b \rightarrow o, \ldots, m \rightarrow z, n \rightarrow a, o \rightarrow b, \ldots, z \rightarrow m$.

   (b) encode and decode the message `"my secret message"` (assume $\langle space \rangle \rightarrow \langle space \rangle$).

6. Given the alphabet $\{0, 1, \ldots, 9\}$,

   (a) create an automaton that accepts numbers in the range $0 - 999999$

   (b) create a transducer that maps numbers (in the range $0 - 999999$) represented as strings of digits to their English read form, e.g.,
   $1 \rightarrow$ one
   $11 \rightarrow$ eleven
   $111 \rightarrow$ one hundred eleven
   $1111 \rightarrow$ one thousand one hundred eleven
   $11111 \rightarrow$ eleven thousand one hundred eleven

   (c) Randomly generate several numbers both as strings of digits and in their read form.

7. Given the alphabet $\{a, b, \ldots, z, \langle space \rangle\}$, create a spelling corrector transducer that implements the (imperfect) traditional rule – 'i before e except after c'. Use it to correct the inputs 'yeild' and 'reciept'.

8. Given the alphabet $\{I, V, X, L, C, D, M\}$,

   (a) create a weighted automaton that assigns to Roman numerals their numeric value (hint: use `fsmbestpath`).

   (b) Epsilon-remove, determinize, and minimize this automaton. Draw the automaton before and after these operations.

9. Given the alphabet $L = \{A, G, T, C\}$,

(a) create a transducer T that implements edit distance –

$$d(x,x) = 0, x \in L$$
$$d(x,y) = d(x,\epsilon) = d(\epsilon,y) = 1, x \neq y \in L \; .$$

(b) using T, find the best alignment between the strings 'AGTCC' and 'GGTACC'

(c) find the second best alignment

10. Consider the following words and their pronunciations (in ARPABET):

any	eh n iy
e.	iy
many	m eh n iy
men	m eh n
per	p er
persons	p er s uh n z
sons	s uh n z
suns	s uh n z
to	t uw
tomb	t uw m
too	t uw
two	t uw

(a) create a pronunciation lexicon L for these words – i.e., the closure of the pronunciation-to-word transducer.

(b) using L, find all possible word parsings of 't uw m eh n iy p er s uh n z'. Give the result as a graph and as a list of strings in order of fewest to greatest number of words per string.

(c) consider the (improbable) bigram language model that gives the cost of word β being followed by word α as:

$$Cost(\alpha|\beta) = \big|\, ||\alpha|| - ||\beta|| \,\big|,$$

where $||\gamma||$ is the number of phonemes in the pronunciation of the word γ. Create a weighted acceptor that implements this LM. Find the best parsing of the string in (b) when constrained by this language model.