LING 570: Hw3
Due date: 11:45pm on Oct 20

Goal: Become familiar with FST.
All the example files mentioned below are under **hw3/examples**/.

Q1 (12 points): Manually create FSTs for the following regular relations and save the FSTs in Carmel format as files "fst1", "fst2", "fst3" under **hw3_dir/q1**/.

- fst1 for $\{(a^{2n}, b^{n}) \mid n >= 0\}$
- fst2 for $\{(a^{n}, b^{2n}c) \mid n >= 0\}$
- fst3 for $\{(a^{n} d^{*}, (bc)^{n} g) \mid n >= 0\}$

Q2 (18 points): Use Carmel to build a FST acceptor, **fst_acceptor.sh**.

- The format of the command line is: fst_acceptor.sh fst_file input_file > output_file
- fst_file is an FST in the Carmel format (e.g., "**examples/fst0**", "**examples/wfst1", "examples/wfst2"**)
- Each line in the input file is a string (e.g., "**examples/ex**", "**examples/ex2**")
- Each line in the output_file has the format "x => y prob" (e.g., "**examples/ex.fst0**"), where
    - x is the string from the input file.
    - y is the output string if x is accepted by the FST, or *none* if x is not accepted by the FST.
    - prob is the probability of the path whose yield is x.
    - The probability of a path is the product of the probabilities of the edges in the path.
    - If there are multiple paths for an input string x, y is the output string of the path with the highest probability (for paths with the same probabilities, Carmel breaks the tie somehow)

- Run your fst_acceptor.sh with the FSTs in Q1 and hw3/examples/ex as input file, save the output files in ex.fst[1-3], respectively, under **hw3_dir/q2/.**

```
fst_acceptor.sh   hw3_dir/q1/fst1     hw3/examples/ex >  hw3_dir/q2/ex.fst1
…
fst_acceptor.sh   hw3_dir/q1/fst3     hw3/examples/ex >  hw3_dir/q2/ex.fst3
```

Q3 (55 points): Build **fst_acceptor2.sh** WITHOUT using Carmel.

- fst_acceptor2.sh has the same command line format and functionality as fst_acceptor.sh.

- The only difference is that fst_acceptor2.sh CANNOT use Carmel; for example, the code will need to read in the fst_file, store the FST in some data structure, and determine whether each line in the input_file is accepted by the FST and if so what the output string and the probability should be.

- Note that the input FST might be nondeterministic. Unlike FSA, not every non-deterministic FST can be converted to a deterministic one. So your code needs to follow multiple paths for an input string and check whether any of the paths ends at a final state. If there are multiple paths that end at a final state, choose the one with the highest probability. Your algorithm can be an extension of the algorithm in Figure 2.19 on Page 35 of J&M.

- In your note file, briefly explain what data structure you use to store the input FST.

- Run fst_acceptor2.sh with the fst_input files created in Q1 and store the output files under **hw3_dir/q3**/.

Q4 (5 points) Run the following commands and save the output files under **hw3_dir/q4/.**

fst_acceptor.sh hw3/examples/wfst**1** hw3/examples/ex**2** > hw3_dir/q4/ex2.wfst1

fst_acceptor.sh hw3/examples/wfst**2** hw3/examples/ex**2** > hw3_dir/q4/ex2.wfst2

Q5 (10 points) Run the following commands and save the output files under **hw3_dir/q5/.**

fst_acceptor**2**.sh hw3/examples/wfst1 hw3/examples/ex2 > hw3_dir/q5/ex2.wfst1

fst_acceptor**2**.sh hw3/examples/wfst2 hw3/examples/ex2 > hw3_dir/q5/ex2.wfst2

The submission hw3_dir/ should include:

- The hw3 note file that includes your description of FST in Q3.

- The source and shell scripts for Q2 and Q3, and any scripts called by them.
- hw3_dir/q1/ includes the three FSTs for Q1 (fst1, fst2, and fst3).
- hw3_dir/q2/ and hw3_dir/q3/: the files, ex.fst[1-3], created in Q2 and Q3.
- hw3_dir/q4/ and hw3_dir/q5/: the files, ex2.fst[1-2], created in Q4 and Q5.