Ling 5801: Problem Set 1

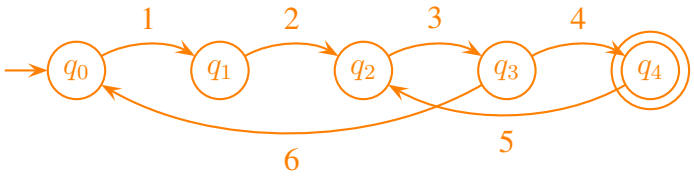
Due via Carmen dropbox at 11:59 PM 9/11.

1. [5 pts.] Draw an FSA, according to the definition in the lecture notes, that recognizes the following language: (0*|1*0). (The bar out-scopes the other operators, so this is equivalent to: (0*)|(1*0).) Try to define your FSA using the fewest states possible.

2. [5 pts.] Write a regular expression, according to the definition in the lecture notes, that recognizes any string of 1's and/or 0's containing an even number of 0's. Try to make your regular expression as concise as possible.

3. [5 pts.] Draw an FSA, according to the definition in the lecture notes, that recognizes any string of 1's and/or 0's containing an odd number of 0's and an odd number of 1's. Try to define your FSA using the fewest states possible.

4. [1 pt. extra credit – tricky!] What language does the following FSA generate? Answer using a regular expression. Try to make your regular expression as concise as possible.



5. [30 pts.] PROGRAMMING:

   (In general for your programming problems you should hand in the following as separate files – DO *NOT* submit a single zip or tar file or other collection, this will result in a 2 point deduction for each problem):

   - a copy of your Makefile (NOTE: please rename it 'Makefile.txt' so I can read the text!)
   - a copy of each script you write,
   - a representative sample of each source (input) file you use,
   - a representative sample of each target (output) you produce.

   You may use the 'sample text' file on the course web page, which has several big numbers in it.

   Write a Makefile that can do all of the following —

   (a) [10 pts.] Using the unix commands described in lecture notes, make a target '%.numlines' file, consisting of the *lines* in a corresponding source '%.txt' file that contain a number, defined here to be a maximal sequence of digits, commas, or decimal points ending in a digit. If a number in prose is immediately followed by a comma or period, you should not assume the comma or period is part of the number.

   So, for example, for the following text file:

   ```
   Kilograms are units of mass.
   ```

```
There are 2.2 pounds in a kilogram.
```

Your program should print:

```
There are 2.2 pounds in a kilogram.
```

(b) [10 pts.] Using the unix commands described in lecture notes, make a target '%.ints' file, consisting of the *non-decimal numbers themselves* (one per line) in a corresponding '%.txt' file. These are numbers (defined here to be a maximal sequence of digits, commas, or decimal points ending in a digit) that do not contain a decimal point.

So, for example, for the following text file:

```
There are 2.2 pounds in a kilogram.
That's less than 3.
```

Your program should print:

```
3
```

(c) [10 pts.] Using the unix commands described in lecture notes, make a target '%.numclass' file, by replacing every number in a corresponding source '%.txt' file (numbers defined here to be a maximal sequence of digits, commas, or decimal points ending in a digit) with the string 'BIGNUM' if the number is greater than or equal to 10,000, or the string 'SMALLNUM' if the number is less than 10,000. (Remember that numbers may include commas and decimal points.)

So, for example, for the following text file:

```
There are 2.20462 pounds in a kilogram.
```

Your program should print:

```
There are SMALLNUM pounds in a kilogram.
```