

Homework #3

Due: Saturday, February 7th, 2015, 11:59 PM

Problem 1: Regex Crossword

Go to <http://regexcrossword.com/> and go through the tutorial to learn how to play *Regular Expressions Crossword* puzzles. You may want to go through the “Beginner”, “Intermediate” and “Experienced” puzzles for extra fun/practice.

Then, go to <http://regexcrossword.com/challenges/palindromeda/puzzles/3> (the third of the “Palindromeda” puzzles). Solve it, validate your answer, and print out your solution to a file `HW31.pdf`. Submit using bundle.

Problem 2: Pumping Lemma

Prove that the following language is not regular using the pumping lemma.

$$L = \{a^n b^m \mid n \leq m \leq 2n \text{ or } m \leq n \leq 2m\}$$

Submit your solution as `HW32.pdf` using bundle.

Problem 3: Extended regular expressions

It is often convenient in practice to extend basic regular expressions as studied in class, with a few additional operations, e.g.,

- “ $E?$ ”, denoting 0 or 1 occurrences of a word from $L(E)$. This can be used to denote that part of an expression is optional.
- “ E^+ ”, denoting 1 or more occurrences of words from $L(E)$. This is similar to E^* , but requires at least one word to be selected.
- “ $E_1 \langle \rangle E_2$ ”, denoting the concatenation of a two words from $L(E_1)$ and $L(E_2)$ in any order.

Show that all these extensions do not increase the power of regular expressions, in the sense that each expression making use of $?, ^+, \langle \rangle$ (in addition to the basic operations $\cup, \circ, *$) can be translated into an equivalent expression using only $\cup, \circ, *$. Specifically, for each of the above extended expressions, give a regular expression (built from subexpressions E, E_1, E_2) equivalent to it.

- Then apply all transformations to the following more complex example expression: $((a(b?)) \langle \rangle c)^+$

Submit your solution as `HW33.pdf` using bundle.

Problem 4: Regular Expressions in the Real World

In this problem, you are going to be working with real data (taken from the UCSD library system) and the UNIX command line tool `egrep`. You can refer to this link for a quick tutorial http://www.cs.columbia.edu/~tal/3261/fall107/handout/egrep_mini-tutorial.htm or read the manual page for `egrep` (`man egrep` from the command line) to see what format is used for regular expressions, and `egrep` command line options. `Egrep` works a line at a time and it matches at any point in the line, not necessarily at the beginning. That means, it acts as though the regular expression starts and ends with Σ^* to match any number of any character.

The data file `data.txt` can be downloaded from the course web page as part of this assignment. It contains data from the UCSD library (in plain text format). Take a look at the file to familiarize yourself with the format. You can use `egrep` to extract information from the file. For example, to find every entry that is in the SSH 6th or 7th floor, run the command:

```
egrep 'SSH (6|7)th Floor' data.txt
```

This should print out quite a lot of data, 191 lines worth, in fact. Instead of printing out all of the lines, we can just get a count of the lines by using the `-c` option to `egrep` as in the command:

```
egrep -c 'SSH (6|7)th Floor' data.txt
```

which prints out 191. You can also put the regular expression in a file `file.rex`, and then search for it using the command `egrep -f file.rex data.txt`

For each query below, give a corresponding `egrep` regular expression (e.g., `SSH (6|7)th Floor` in the above example) and check the number of matching lines using `egrep -c`. All of them will have at least one, so if you get zero, something went wrong. I suggest examining the output without the `-c` to ensure that you are finding the entries you want. Note that each book can have multiple call numbers in the library. You should consider each of those to be a separate book. In addition, each call number can have multiple copies, these can be treated as a single book.

- Modify the example above to find all books that are in the SSH 6th or 7th floor and have a year of 2002. Note that years are displayed as either 2002 or c2002. Make use of `.*` to skip over data you do not care about. How many are there?
- How many books have a call number whose first part is exactly QA1? That is, QA1 .L471 v.642 should be matched but QA171 .M648 1961 should not.
- How many books in the S&E Stacks are available? (Look at the data file to see how available books are represented; it should be very obvious.)
- How many books in the S&E Stacks have a year of 1990 or later?
- How many books in the S&E Stacks are available and have a year between 1975 and 1999, inclusive?

Write each expression in a separate file `HW34a.rex, ..., HW34e.rex`, so that they can be tested with the command `egrep -c -f HW34x.rex data.txt`. Each file should contain a single regular expression, given on a single line. Submit your `HW34x.rex` files using `bundle`. (You don't need to submit the numerical answers to the questions, just the files with the regular expressions.)