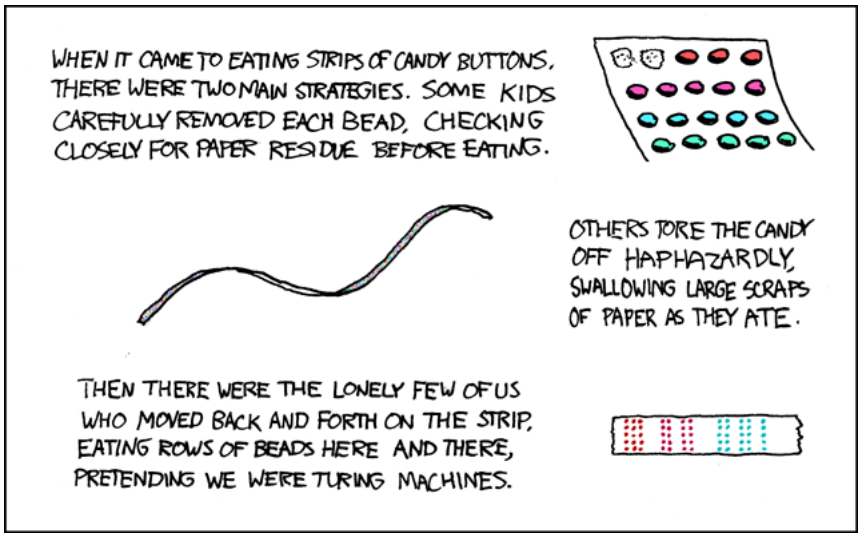
CS51 - Assignment 10
Finite Automata and Turing Machines

Due: December 4th, at 11:59pm

NO EXTENSIONS



https://xkcd.com/205/

For this assignment, we will:

- Gain more practice working with DFAs, NFAs, and Turing machines.

1 Getting started

We will use a Java-based program, called JFLAP, that was written by Professor Susan Rodger of Duke University. You can download JFLAP here `https://www.jflap.org/jflaptmp/july27-18/JFLAP7.1.jar`. Keep in mind that since it's a jar file, you might have to follow the same steps with assignment 4, when you downloaded Logisim Evolution.

Each problem, except for Problem 5, requires you to construct a finite state automaton or a Turing machine. When you start JFLAP, select "Finite Automaton" or "Turing Machine," as appropriate.

Construct your machine and use FJLAP to test it before saving it. Then save your JFLAP machine in a file named `assignment10-`⟨one-digit-problem-number⟩`.jff`

For example, the file for Problem 3 would be named `assignment10-3.jff`. Problem 5 is an exception; it asks for a text file named `aassignment10-5.txt`. When you have finished the assignment, submit each file separately (not a zip file) in the usual way on Gradescope.

Finite Automata

1.1 Problem 1

Use JFLAP to construct a DFA that accepts all non-empty strings over the alphabet $\{a, b\}$ in which every other letter in the string is an a, starting with the first letter. The empty string should not be accepted, but ab and aaa should be.

1.2 Problem 2

Use JFLAP to construct a six-state DFA which accepts a string over the one-letter alphabet $\{a\}$ if it has a number of a's that is a multiple of 2 or a multiple of 3 (or both).

1.3 Problem 3

Use JFLAP to construct a DFA which accepts a string of 0's and 1's if, when interpreted as a binary number, is a multiple of 7. The binary representation is read from left to right; that is, the *most* significant bit comes first. (This is not trivial; think about it before you start to build the DFA.)

Hint: The key is to think about this as a math problem and not a pattern-matching problem. The automaton is reading the symbols in the string one at a time from left to right.

1.4 Problem 4

Consider strings constructed from a's and b's. The string $aabbbab$ has two occurrences of the substring ab and one occurrence of the substring ba. The string aba has one occurrence of ab and one of ba; it does not matter that they share a character. Use JFLAP to construct a DFA that accepts strings for which the number of times ab appears is equal to the number of times that ba appears.

1.5 Problem 5

Two strings s and t are *distinguishable over a language* L if there is a third string u such that exactly one of su and tu are in L.

- Find six strings that are pairwise distinguishable over the language of Problem 2 and show that they are, in fact, distinguishable. With six strings, there are fifteen pairs to check.
- Explain why any DFA which accepts the language must have at least six states. For this problem, submit a separate file (plain text, not a Word document or a PDF file) named `assignment10-5.txt`.

1.6 Problem 6

Use JFLAP to create a *non-deterministic* finite automaton that has eight states and accepts the strings over a one-letter alphabet $\{a\}$ whose length is a multiple of 2 or a multiple of 5 (or both). Notice that the technique you used in Problem 2 leads to a DFA with ten states.

Turing Machines

Notice that JFLAP uses λ and $\square$ to denote the empty string and the blank character, respectively. The transition $(a, q; b, D, r)$ is specified by an arrow from state q to state r labeled with $a; b, D$. Remember that a Turing machine has two alphabets: the input alphabet and the tape alphabet. The blank character $\square$ is in the tape alphabet but not the input alphabet. You may add other characters to the tape alphabet (in fact, you will have to), but try not to get carried away. Keep your Turing machines as simple as possible.

Your Turing machines should have exactly one accepting state and one rejecting state, even though JFLAP does not enforce that requirement. In fact, JFLAP does not have explicit rejecting states; you can create a rejecting state by making a non-accepting state that has no transitions out of it.

1.7 Unary subtraction

Binary numbers consist of 0's and 1's. Unary numbers consist only of 1's. For example, we represent 1 as 1 in unary, 2 as 11, 3 as 111, etc. Write a Turing machine over the alphabet $\{1, -\}$ that accepts all unary number subtraction problems where the result is positive, i.e greater than 0. For example, "111-11" would be accepted while "11-111" would be rejected. Also, any string without a minus sign, or a string with more than one minus sign, will be rejected.

1.8 Balanced parentheses

Suppose that the input alphabet has exactly two symbols, the left and right parentheses. Create a Turing Machine that accepts the language of all strings of properly balanced parentheses.

For example, the strings

```
()
(())
```

```
(()(((()())))
```

are all accepted as members of the language, but

```
)(
(()
())(
```

are not. Also, the empty string *is* a string of balanced parentheses.

2 Feedback

Create a file named `feedback10.txt` that answers the questions:

- How long did you spend on this assignment?
- Any comments or feedback? What things did you find interesting, challenging, or boring?

3 When you're done

Submit your JFLAP files, problem 5 text file, and `feedback10.txt` online using Gradescope. Be sure you're uploading them to the right assignment. Note that you can resubmit the same assignment as many times as you would like up until the deadline.

4 Grading

	points
`Problem 1`	1
`Problem 2`	1
`Problem 3`	1
`Problem 4`	1
`Problem 5`	1
`Problem 6`	1
`Unary subtraction`	1
`Balanced parentheses`	1
total	10