Computer Science 52

Assignment 10

Due Friday, December 8, 2017, at 5:00 PM

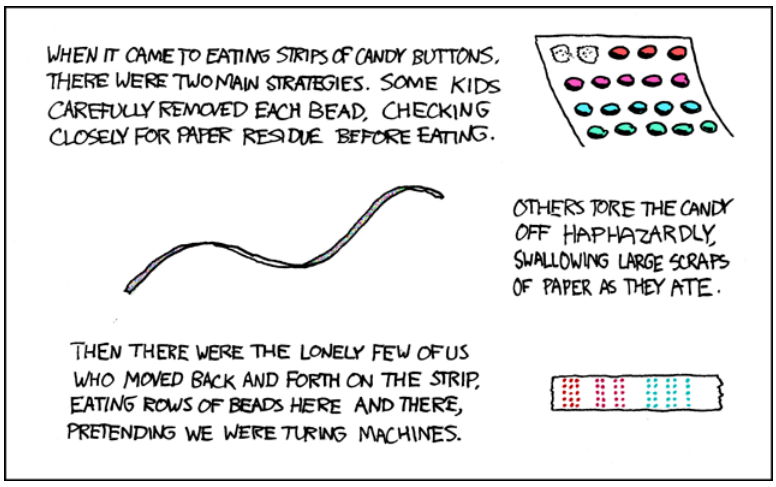
Important note on scheduling Because of the intervening Thanksgiving Recess, we have a bit of overlap on Assignments 9 and 10. The former would normally be due on November 24, the day after Thanksgiving. We are giving you a little extra flexibility and making it due a week later. However, Assignment 10 is will be due one week later, on December 6. We strongly encourage you to finish Assignment 9 early and move on to Assignment 10!

Reading and resources The material for this assignment is in Sections II through VIII of the course document *Languages and Models of Computation.*

We will use a Java-based program, called JFLAP, that was written by Professor Susan Rodger of Duke University. In the Computer Science laboratories, you can start the program with the command

```
/common/cs/cs052/bin/jflap
```

If you want to work outside the laboratories, see the course Resources page for a link to the JFLAP website. Be sure to get JFLAP 7.0. The "Thin" version should work, but move to the "Full" version if you encounter problems. There is also a link to the JFLAP documentation on the Resources page. The diagrams in the course document *Languages and Models of Computation* were constructed using JFLAP.



https://xkcd.com/205

Submission Each problem, except for number 5, requires you to construct a finite state automaton or a Turing machine. When you start JFLAP, select "Finite State Automaton" or "Turing Machine," as appropriate. Construct your machine and use JFLAP to test it before saving it. Then save your JFLAP machine in a file named

`asgt10-`⟨one-digit-problem-number⟩`.jff`

For example, the file for Problem 3 would be named `asgt10-3.jff`. Problem 5 is an exception; it asks for a text file named `asgt10-5.txt`. When you have finished the assignment, submit each file separately (not a zip file) in the usual way. When selecting the file to submit via the web interface you can select multiple files at once by holding down the command key on a Macintosh or the control key on Windows. You can also submit files one at a time.

Finite Automata

1. [2 points] 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.

2. [2 points] 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).

3. [2 points] 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. In Assignment 6, problem 1, when you were processing decimal numbers from left to right, you were able to calculate a partial value for the number so far. Think of trying to do something similar, since the automaton is reading the bits in the string one at a time from left to right.

4. [3 points] 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.

5. [3 points] 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.

a. Find six strings which 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.

b. 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 `asgt10-5.txt`.

6. [2 points] Use JFLAP to create a *nondeterministic* 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 there are two alphabets for a Turing machine, 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.

7. [Unary subtraction, 3 points] 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. For example, "111-11" will be accepted and "11-111" will be rejected. Also, any string without a minus sign, or a string with more than one minus sign, will be rejected.

8. [Balanced parentheses, 3 points] Suppose that the input alphabet has exactly two symbols, the left and right parentheses. Create a Turing machine that accepts the language 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.

CS fifty-two
was a hoot. Each assignment
proved to be so cute

SML, assembly,
circuits, oh my! JFLAP gave
a strain to my eye

Sam Rubin '19, CS52 poet laureate,
reflecting at the end of the semester