

## 1 The Big Picture

The course is basically divided up into regular languages, context-free languages and Turing machines. You've had exam material on the first two, but not the last, so below are practice problems for the Turing machine part.

The final exam is cumulative, but with an emphasis on the last four or five weeks — i.e. the Turing machine stuff. If I were studying for this test, I would first review the notes from the Turing machine stuff, then try to do the practice problems. Then I would do the same for the early material with the 6-week exam problems, and the middle material with the 12-week exam problems.

### Regular languages

deterministic finite automata: informally and formally

non-deterministic finite automata: informally and formally

regular expressions: writing, understanding, converting to NDFAs

algorithms: union, intersection, complement, concatenation, Kleene star, conversion, DFA minimization ...

pumping lemma: following proofs, creating proofs, decomposing strings

perl regular expressions: general “what does it do” stuff

### Context-Free languages

context free grammars: derivations, parse trees, ambiguity, writing grammars

push-down automata: formally and informally, designing machines, converting grammars to PDAs

parsing: the parsing process, tokenization versus parsing, shift reduce parsing with PDAs

pumping lemma for CFLs: following proofs, decomposing strings

bison: general “what does it do” stuff

## 2 Since the 12 Week Exam

Here are some sample questions from the material we covered in the last four or five weeks. You have the 6 and 12 week exams to look over for questions covering the other material.

1. Check out the handout, and draw a diagram for the machine encoded as:

```
0010010001001001000110010101000100011001001010001011
```

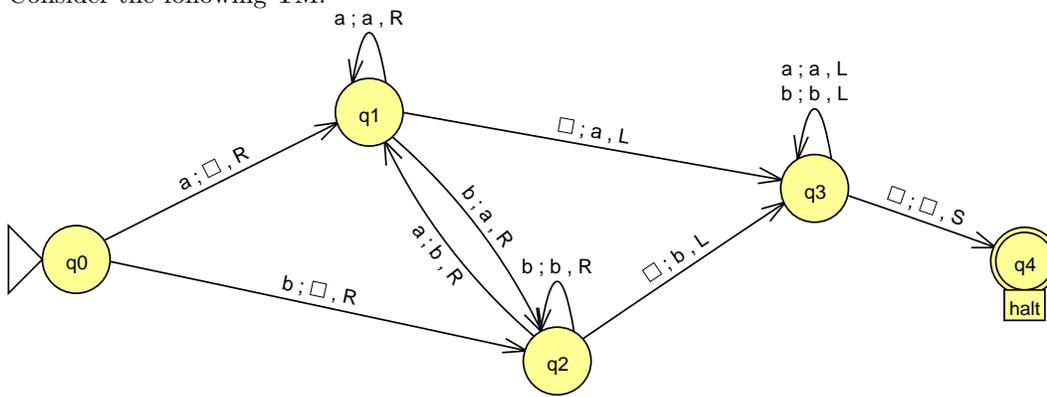
2. Draw a TM (according to our class's Turing machine model) with input alphabet  $\{a, b\}$  that simply erases its input and leaves the tape head in the first cell on the tape.

3. Define what is meant by “the language semi-decided by TM  $M$ .”
4. T/F (**Justify** your answer): A Turing machine is such a simple model, it can't tell us anything about sophisticated and complex modern computers.
5. T/F (**Justify** your answer): There is no limit on what can be done with computers.
6. Draw the TM defined by the following tuple:

$$\left( \{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1\}, \begin{array}{c|ccc} & 0 & 1 & \square \\ \hline q_0 & (q_1, \square, R) & (q_1, \square, R) & (h, \square, S) \\ q_1 & (q_1, 0, R) & (q_1, 1, R) & (q_2, \square, L) \\ q_2 & (q_2, \square, L) & (q_2, \square, L) & (h, \square, S) \end{array}, q_0 \right)$$

7. Getting back to the first square of a TM is always a big pain in the neck, because you have to take care to mark it, otherwise you'll fall off the end of the tape. Suppose we wanted to add a new feature to our Turing machine model. In addition to the Left, Right and Stay tape head moves we currently each transition, we'll allow a “Back-to-square-one” move which, in a single step, returns the tape head to the first square on the tape.
- Refer to the handout and give a formal definition of this augmented Turing machine model. It is enough to tell me what changes to the tuple/configuration/ $\implies$  text on the handout are necessary. You need not copy all the stuff that stays the same.

8. Consider the following TM:



- For input *abba* what *configuration* does this machine halt in? Your answer should be a configuration according to our definition! (see handout). (Hint: writing down the entire sequence of configurations in the computation would make partial credit a lot more likely!)

- Give a concise english description of what this machine accomplishes — i.e. for any input, not just *abba*.

9. Consider the following mystery algorithm:

**Input:** TM  $M_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, s_1)$ , TM  $M_2 = (Q_2, \Sigma_2, \Gamma_2, \delta_2, s_2)$ , where  $Q_1 \cap Q_2 = \emptyset$

**Output:** TM  $M = (\{\$ \} \cup Q_1 \cup Q_2, \{0, 1\} \cup \Sigma_1 \cup \Sigma_2, \{0, 1\} \cup \Gamma_1 \cup \Gamma_2, \delta, \$)$ , where

$$\delta(p, x) = \begin{cases} (s_1, x, R) & \text{if } p = \$ \text{ and } x = 0 \\ (s_2, x, R) & \text{if } p = \$ \text{ and } x = 1 \\ \delta_1(p, x) & \text{if } p \in Q_1 \text{ and } x \in \Sigma_1 \\ \delta_2(p, x) & \text{if } p \in Q_2 \text{ and } x \in \Sigma_2 \\ (p, x, S) & \text{otherwise Note that this causes an infinite loop!} \end{cases}$$

Give a concise english-language description of what this algorithm accomplishes, i.e. “Given two TMs  $M_1$  and  $M_2$ , this algorithm produces a machine  $M$  such that ...”. **Hint:** Suppose  $M_1$  replaces all *as* with *bs*, and suppose  $M_2$  replaces all *bs* with *as*. What would the machine  $M$  produced by this algorithm do for input *0abba*? How about for input *1abba*?

10. Let  $G$  be the grammar

$$\begin{aligned} S &\longrightarrow aTbb \mid \lambda \\ T &\longrightarrow bRa \mid ab \\ R &\longrightarrow Ta \mid cc \end{aligned}$$

with start symbol  $S$ . Let  $w = ababaabb$ . Decompose  $w$  into  $uvxyz$  as the pumping lemma for CFGs would do, i.e. so  $uv^kxy^kz \in L(G)$  for all  $k \geq 0$ . Show your work!