

CS 273: Intro to Theory of Computation, Fall 2007

Quiz 3 Solutions

1. (2 points) Is every context-free language Turing decidable? (Yes or No)

Solution: Yes.

2. (2 points) Let $L = \{\langle M, w \rangle \mid M \text{ is a Turing machine and } M \text{ accepts } w\}$. Is L Turing recognizable, Turing decidable, or neither?

Solution: Turing recognizable.

3. (3 points) Briefly explain the difference between a Turing recognizable language and a Turing decidable language.

Solution: A decidable language needs to be recognized by a Turing machine that **halts on all inputs**. If the language is only Turing recognizable, it might be that none of the Turing machines recognizing it is guaranteed to halt on all inputs. That is, they run forever, rather than rejecting, some inputs not in the language.

An important subtlety (which I did not deduct points for) is that a decidable language L needs to have at least one TM which recognizes it and always halts. But there might also be some other TMs which recognize L but don't always halt. So it's not the case that all TMs recognizing L must halt. (In fact, if you give me a decider for any language, it's easy to modify its code to make a TM that's similar but loops on inputs not in the language.)

Also notice that the question asked when a language was decidable. Some of you answered the closely related question: when is a TM a decider. Or, you said things like "a Turing-decidable language always halts." (It's the machine that halts, not the language.) This kind of confusion will eventually lose you points (in this or some later class).

4. (2 points) Suppose that a Turing machine M contains the transition $\delta(r, f) = (p, c, L)$ (r and p are states; f and c are tape symbols).

If M is now in configuration $aadrfb$, what will its next configuration be?

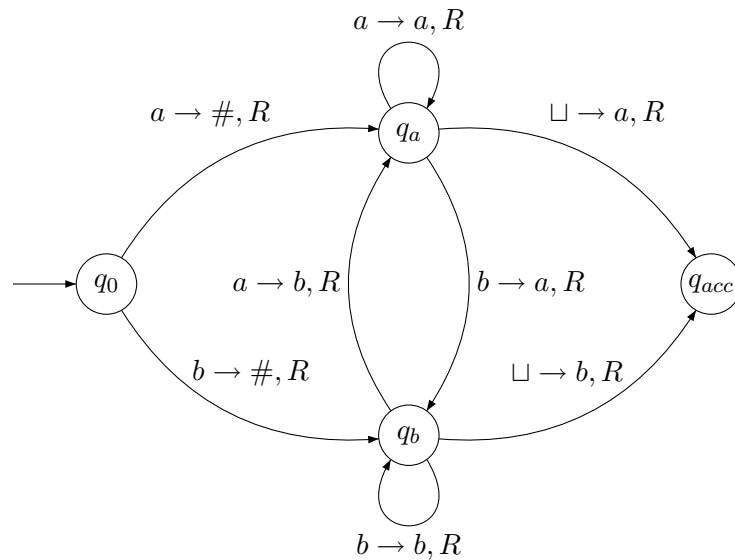
Solution: That is, currently the tape contains $aadf b$, the head is on the f , and the machine is in state r . After the transition, the tape contains $aadcb$, the head is on the d , and the machine is in state p . So the new configuration is $aapdcb$.

5. (4 points) Draw the state diagram for a Turing machine with input alphabet $\{a, b\}$ which writes a $\#$ in the first tape position, shifting the input string one position rightwards, and then accepts leaving its head on first blank after the shifted input. You can assume that the input always contains at least one non-blank character (i.e. the TM will reject the empty input).

Your diagram should not include the reject state or transitions to it. Hint: it can be done with only 4 non-reject states.

Solution: Suppose \sqcup is the blank symbol and q_{acc} is the accept state. As we move rightwards along the input, we need to pick up each character and remember it in our state, so we can write it into the next tape position. The state q_a remembers the character a and the state q_b

remembers the character b . When we write the last shifted character, we move right so that we end up on the first blank after the shifted input.



A common minor mistake was to forget to write out the last remembered character when you hit the blank at the end of the input.

Here's another reasonable approach, more complicated but worth full credit when correct. Move the head all the way to the right. Then move left shifting each character one space right, leaving a blank where it used to be. If you shift left from this new blank but still see a blank, you're at the lefthand end, so write a sharp sign and then move to the righthand end of the tape.

6. (4 points) Let L and K be two languages. Suppose that K reduces to L . I.e. if a decider for language L exists, this decider can be used to construct a decider for language K . Based on this information, state whether each of the following claims is true or false:
- (a) If L is undecidable, then K is undecidable. **Solution:** False
 - (b) If K is undecidable, then L is undecidable. **Solution:** True
 - (c) If K is TM-decidable, then L is TM-decidable. **Solution:** False
 - (d) If L is TM-decidable, then K is TM-decidable. **Solution:** True

Notice that (d) must be true, because it's essentially a repeat of what the problem said was true. (a) is false, because our information only talks about what's true if L is decidable and we've said nothing about what happens when L isn't decidable.

(b) is the contrapositive of (d), so they have the same truth value. Similarly, (a) is the contrapositive of (c), so they have the same truth value.

7. (4 points) Suppose that $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ is a Turing machine. M is defined to accept a string w if there is a sequence of configurations of C_1, C_2, \dots, C_k which satisfy three conditions. The first condition is that C_1 is q_0w . What are the other two conditions?

Solution: (1) C_k contains q_{accept} and (2) C_i yields C_{i+1} for every i from 0 to $k - 1$.

Many people said that $C_k = xq_{accept}$ which is almost right, except that there might be more stuff on the tape after the head position. Turing machines can halt with the head in any position on the tape.

Many people got partial credit for equations that weren't right but expressed the idea behind one or both conditions.

It was ok, but not necessary, to spell out the definition of "yields" in terms of changes to configurations. (See Sipser for the details.)

8. (4 points) Suppose that we are given an encoding of a DFA M . Sketch an algorithm for deciding whether $L(M)$ is empty.

Solution: $L(M)$ is empty if there is a path through the state diagram from the start state to some final state. We can use any standard graph search algorithm (e.g. breadth-first search), starting at the start state and following the transitions of the DFA. States are marked "explored" when they are pushed onto the search queue, so that they can't be put back onto the queue a second time. If the search finds a final state, the algorithm should return the answer "no" (that is, $L(M)$ is not empty). If the search queue empties without a final state having been found, the algorithm should return the answer "yes" (that is $L(M)$ is empty).

Another approach that works, though it's a bit brute-force, is to enumerate all strings whose length is less than the number of states in M , and check each one to see if the DFA accepts it. By the pumping lemma, if none of these shorter strings is in the language, then the language must be empty.

A number of people had essentially correct solutions, but reversed the conditions for accepting vs. rejecting. This is easy to do under test conditions. A number of people adopted the very sensible strategy of having their algorithm report "empty" or "not empty", so it was more obvious how to interpret its answers.