Homework 7: Unit 2 Reflection and TM Warmup

CSCI 161

Due 9pm, Sunday, November 8, 2020

Krehbiel

Overview. This week's homework will be a reflection on the course so far and a self-paced introduction to Unit 3. The first question is a freeform reflection on not only how Unit 2 was different for you than Unit 1 but also on your goals for the rest of the course. The second question has you apply the technical definition of a Turing machine and its computation model (discussed in class and Section 3.1, and typed up for reference here) to a specific example. The third question asks high level reading questions about Sections 3.2 and 3.3 of the text, which you will not be tested on explicitly but are important for the perspective they offer on this unit.

Note that HW7 is due at the end of the weekend to give you time to read; like HW4 after Midterm 1, it is light on problem-solving. HW8 will be posted before HW7 is due and will be due next Thursday evening. As always, turn in a modified version of this tex file with the associated compiled pdf for 5% extra credit.

Question 0. What collaboration, if any, did you use for Questions 2 and 3? Elaborate to the extent helpful.

Question 1. (4pts) What did you do differently in Unit 2 vs Unit 1, and how do you think it paid off or didn't? What have been your main takeaways from the course so far? What do you hope to walk away with from the course, and what will you focus on in Unit 3 to achieve those goals?

Question 2. (4pts) List the 7 missing configurations in the computation history of the Turing machine from Example 3.9 on the input 01#01. Formal definitions from Section 3.1 typed up below.

Definition 3.3. A Turing machine is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ with finite state set Q, finite input alphabet $\Sigma \not\ni \sqcup$, finite tape alphabet $\Gamma \supseteq \Sigma \cup \{\sqcup\}$, $q_0, q_{\text{accept}}, q_{\text{reject}} \in Q$ with $q_{\text{accept}} \neq q_{\text{reject}}$ as the start, accept, and reject states, resp., and transition function $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$.

The status of a Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ can be expressed as configuration uqv for $q \in Q$ and $u, v \in \Gamma^*$, where u denotes the tape contents to the left of the head, v denotes the tape contents under and to the right of the head, and q denotes the current state. There are two options for how one configuration may yield another according to δ:

- uaq_ibv yields uq_jacv if $\delta(q_i, b) = (q_j, c, L)$ for $a, b, c \in \Gamma,\ u, v \in \Gamma^*,\ q_i, q_j \in Q$;
- uaq_ibv yields $uacq_jv$ if $\delta(q_i, b) = (q_j, c, R)$ for $a, b, c \in \Gamma,\ u, v \in \Gamma^*,\ q_i, q_j \in Q$.

M accepts an input $w \in \Sigma^*$ if there exists a sequence of configurations $C_1, \ldots, C_k$ for some $k \geq 1$ such that

- $C_1 = q_0w$ (start at the start configuration),
- C_i yields C_{i+1} for each $i = 1, \ldots, k-1$ (the transitions obey δ), and
- $C_k = uq_{\text{accept}}v$ for some $u, v \in \Gamma^*$ (reach an accept configuration).

Similarly, M rejects an input $w \in \Sigma^*$ if there exists a configuration sequence as above but with $C_k = uq_{\text{reject}}v$ for some $u, v \in \Gamma^*$ (a reject configuration).

The language of TM M is $L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$. A Turing machine is called a decider if for every string in Σ^* is either accepted or rejected. $L(M)$ is called the language recognized by M, and if M is a decider we may also call $L(M)$ the language decided by M.

Definitions 3.5 and 3.6. A language is called Turing-recognizable if some Turing machine recognizes it. A language is called decidable if some Turing machine decides it.

Example 3.9. A Turing decider for $\{w\#w \mid w \in \{0,1\}^*\}$.

High-level algorithmic idea:

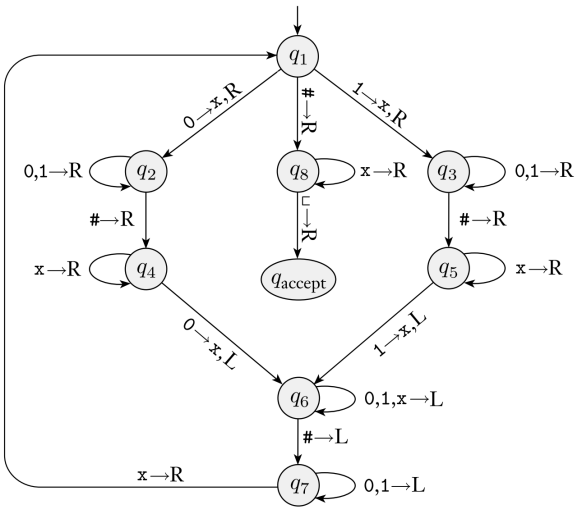
- Zig-zag across the tape to corresponding positions on either side of the # symbol to check whether these positions contain the same symbol. If not, or it no # is found, *reject*. Cross off symbols as they are checked to keep track.
- When all symbols to the left of the # have been crossed off, check for any remaining symbols to the right. *Reject* if symbols remain and *accept* otherwise.

Low-level pseudocode and state diagram defining formal machine:

Turing decider $M = (\{q_1, \ldots, q_8, q_{\text{accept}}, q_{\text{reject}}\}, \{0,1,\#\}, \{0,1,\#,x,\sqcup\}, \delta, q_1, q_{\text{accept}}, q_{\text{reject}})$

- While tape head is 0 or 1:
  - Replace tape head b with x and move R.
  - Scan R for as long as the tape is 0 or 1.
  - If tape head = $\sqcup$: *reject*.
  - Move R past # and scan R past every x.
  - If tape head $\neq b$: *reject*.
  - Replace b with x and move L.
  - Scan L until # and then until first x.
  - Move R to the next unmatched character.
- If tape head = #:
  - Move R and scan R until next non-x.
  - If tape head = $\sqcup$: *accept*.

  Else: *reject*.



Q2 task: Complete the missing configurations in the computation history of M on input $01\#01$.

start config	first match	second match	final scan	accept config
	$C_2 = xq_2 1\#01$	$C_9 =$		
	$C_3 = x1q_2\#01$	$C_{10} =$		
	$C_4 = x1\#q_4 01$	$C_{11} =$	$C_{16} = xx\#q_8xx$	
$C_1 = q_1 01\#01$	$C_5 = x1q_6\#x1$	$C_{12} =$	$C_{17} = xx\#xq_8x$	$C_{19} = xx\#xx \sqcup q_{\text{accept}}$
	$C_6 = xq_7 1\#x1$	$C_{13} =$	$C_{18} = xx\#xxq_8$	
	$C_7 = q_7 x1\#x1$	$C_{14} =$		
	$C_8 = xq_1 1\#x1$	$C_{15} =$		

Question 3. (4pts) Mark the following statements about Sections 3.2 and 3.3 of the text as true or false.

a) A multitape Turing machine can have instructions that do not move the tape head, even if it is not over the leftmost memory cell.

b) There exists a language that is recognized by a nondeterministic Turing machine but is not recognized by any deterministic Turing machine.

c) For any string in any Turing recognizable language, any enumerator for that language will eventually print the string.

d) There exists some Turing-recogizable nonregular language that has an enumerator that will print the entire language in finite time.

e) Any Turing machine can be implemented in C++.

f) Any C++ program can be simulated on a Turing machine.

g) High-level pseudocode describing a Turing machine can access arbitrary points in memory.