HW5 : Turing Machines

CSE105Sp23

Due: May 16th at 5pm (no penalty late submission until 8am next morning), via Gradescope

In this assignment: You will practice designing and working with Turing machines and their variants.

Resources: To review the topics you are working with for this assignment, see the class material from Week 5 through Week 6. We will post frequently asked questions and our answers to them in a pinned Piazza post.

Reading and extra practice problems: Chapter 3. Chapter 3 exercises 3.1, 3.2.

Key Concepts: Formal definitions of Turing machines, computations of Turing machines, halting computations, implementation-level descriptions of Turing machines, high-level descriptions of Turing machines, recognizable languages, decidable languages, variants of Turing machines, enumerators, nondeterministic Turing machines, Church-Turing thesis.

For all HW assignments: Weekly homework may be done individually or in groups of up to 3 students. You may switch HW partners for different HW assignments. The lowest HW score will not be included in your overall HW average. Please ensure your name(s) and PID(s) are clearly visible on the first page of your homework submission and then upload the PDF to Gradescope. If working in a group, submit only one submission per group: one partner uploads the submission through their Gradescope account and then adds the other group member(s) to the Gradescope submission by selecting their name(s) in the "Add Group Members" dialog box. You will need to re-add your group member(s) every time you resubmit a new version of your assignment. Each homework question will be graded either for correctness (including clear and precise explanations and justifications of all answers) or fair effort completeness. You may only collaborate on HW with CSE 105 students in your group; if your group has questions about a HW problem, you may ask in drop-in help hours or post a private post (visible only to the Instructors) on Piazza.

All submitted homework for this class must be typed. You can use a word processing editor if you like (Microsoft Word, Open Office, Notepad, Vim, Google Docs, etc.) but you might find it useful to take this opportunity to learn LaTeX. LaTeX is a markup language used widely in

computer science and mathematics. The homework assignments are typed using LaTeX and you can use the source files as templates for typesetting your solutions. To generate state diagrams of machines, we recommend using Flap.js or JFLAP. Photographs of clearly hand-drawn diagrams may also be used. We recommend that you submit early drafts to Gradescope so that in case of any technical difficulties, at least some of your work is present. You may update your submission as many times as you'd like up to the deadline.

Integrity reminders

- Problems should be solved together, not divided up between the partners. The homework is designed to give you practice with the main concepts and techniques of the course, while getting to know and learn from your classmates.
- You may not collaborate on homework with anyone other than your group members. You may ask questions about the homework in office hours (of the instructor, TAs, and/or tutors) and on Piazza (as private notes viewable only to the Instructors). You *cannot* use any online resources about the course content other than the class material from this quarter – this is primarily to ensure that we all use consistent notation and definitions (aligned with the textbook) and also to protect the learning experience you will have when the 'aha' moments of solving the problem authentically happen.
- Do not share written solutions or partial solutions for homework with other students in the class who are not in your group. Doing so would dilute their learning experience and detract from their success in the class.

You will submit this assignment via Gradescope (https://www.gradescope.com) in the assignment called "hw5CSE105Sp23".

Assigned questions

1. REP(**Describing a Turing Machine for** REP) (18 points):
   Recall that for $L \subseteq \Sigma^*$ with $\Sigma = \{0, 1\}$, we define the language over $\Gamma = \{0, 1, 2\}$

   $$\begin{aligned}\textsc{Rep}(L) &:= \{w \in \Gamma^* \mid \text{between every pair of successive 2's in } w \text{ is a string in } L\} \\ &= \{w \in \Gamma^* \mid \text{for all } v \in \Sigma^* \text{ if } 2v2 \in \textsc{Substring}(\{w\}), \text{ then } v \in L\}\end{aligned}$$

   where for all languages $K \subseteq \Gamma^*$ we let

   $$\textsc{Substring}(K) := \{w \in \Gamma^* \mid \text{there exist } a, b \in \Gamma^* \text{ such that } awb \in K\}.$$

   In this question, you will give three separate descriptions of a Turing machine which recognizes the language $\textsc{Rep}(\{0^n1^n \mid n \geq 0\})$.

   This may seem like a somewhat tedious process, but we think that it is important to see all the different descriptions in action at least once for a single language.

With each description, give a brief justification connecting the description to the language it recognizes.

(a) (*Graded for completeness*) [1] High-level description: description of algorithm (precise sequence of instructions), without implementation details of machine. Your description can use data structures and refer to specific positions in the input strings (without specifying memory management).

(b) (*Graded for correctness*) [2] Implementation-level description: English prose that describes the Turing machine head movements relative to contents of tape, and conditions for accepting / rejecting based on those contents.

(c) (*Graded for correctness*) Formal definition: Give the 7-tuple of parameters

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$$

describing the Turing Machine. Represent the transition function δ by drawing the state diagram of the Turing machine. You may use the following conventions: omit the reject state from the diagram; any missing transitions in the state diagram are assumed to go to the reject state.

2. **This Turing Machine is broken... or is it?** (12 points):
Let's consider a variant of a Turing Machine which models computation where the data keeps getting corrupted. A Corrupted Turing Machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ has some of the usual Turing Machine components: set of states (Q); input alphabet (Σ); transition function (δ); start state (q_0); accepting state (q_{acc}), rejecting state (q_{rej}). Unlike a normal Turing Machine, the tape alphabet $\Gamma \supseteq \Sigma$ has two (rather than one) special characters $\sqcup$ and $\lightning$:

- $\sqcup \in \Gamma$, $\sqcup \notin \Sigma$: This is the usual blank symbol.
- $\lightning \in \Gamma$, $\lightning \notin \Sigma$: This symbol indicates a cell that has been corrupted. The machine cannot write over any corrupted cell; that is, for each $q \in Q$,

$$\delta(q, \lightning) \in \{(r, \lightning, D) \mid r \in Q, D \in \{L, R\}\}$$

Computation in the corrupted Turing Machine proceeds as normal except that sometimes when writing a tape symbol as intended, a $\lightning$ symbol is written instead. Thankfully, the pattern of corruption is predictable: the first write is corrupted, and then every other write thereafter is corrupted. That is, the first, third, fifth,... etc. writes are corrupted.

[1] This means you will get full credit so long as your submission demonstrates honest effort to answer the question. You will not be penalized for incorrect answers. To demonstrate your honest effort in answering the question, we ask that you include your attempt to answer *each* part of the question. If you get stuck with your attempt, you can still demonstrate your effort by explaining where you got stuck and what you did to try to get unstuck.

[2] This means your solution will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

(a) (*Graded for correctness*) Prove that for every **regular** language L, there exists a corrupted Turing Machine M that recognizes L.

(b) (*Graded for completeness*) It will turn out that the Corrupted Turing Machine is no less powerful than our usual definition of a Turing Machine. Let's break the proof into a few steps. First, define the alphabet $\Sigma_{\text{pairs}} := \Sigma \times \Sigma = \{(a, b) \mid a \in \Sigma, b \in \Sigma\}$ of pairs of symbols in Σ. Give the construction of a Corrupted Turing Machine that takes input $a = a_1 a_2 \cdots a_n \in \Sigma^*$ and rewrites it as pairs of symbols interspersed by corrupted symbols. To be precise, the Corrupted Turing Machine should take starting tape configuration

a_1	a_2	$\ldots$	a_{n-1}	a_n	$\sqcup$	$\sqcup$	$\ldots$

to the tape configuration

↯	(a_1, a_2)	↯	(a_3, a_4)	↯	$\ldots$	↯	(a_{n-1}, a_n)	↯	$\sqcup$	$\sqcup$	$\ldots$

with the head once again pointing to the first cell. For simplicity, you may assume that n is even.

(c) (*Graded for completeness*) Starting from the tape configuration we created in the previous step, describe the implementation of a Corrupted Turing Machine which simulates the computation of any normal/uncorrupted Turing Machine.

Challenge; not graded: A key idea in the construction above was to increase the number of symbols in our tape alphabet. Can you do the same construction *without* increasing the size of the tape alphabet? That is, for every language L recognized by a normal Turing machine over alphabet Σ and tape alphabet $\Sigma \cup \{\sqcup\}$, is there a Corrupted Turing Machine with tape alphabet $\Sigma \cup \{\sqcup, ↯\}$ that recognizes L?

3. **True/False enumerator** (20 points):
For each of the following statements, determine if it is true or false. Clearly label your choice by starting your solution with **True** or **False** and then provide a brief (3-4 sentences or so) justification for your answer.

(a) (*Graded for correctness*) Every enumerator enumerates an infinite language.

(b) (*Graded for correctness*) Let E be any enumerator and M be any Turing machine. If $L(E) = L(M)$, then M enters the reject state for all strings not in $L(E)$.

(c) (*Graded for correctness*) Let E be any enumerator over Σ. Suppose $a, b \in \Sigma^*$ and $a, b \in L(E)$. If E prints a before b, then $|a| \leq |b|$.

(d) (*Graded for correctness*) Let M decide language L over Σ such that M halts on all inputs $w \in \Sigma^*$ in $|w|^{2023}$ steps. There exists an enumerator E with the following properties: $L(E) = L(M)$; and if $a, b \in L$ and $|a| < |b|$, then E prints a before b.

(e) (*Graded for correctness*) Let N be a nondeterministic Turing machine. There is an enumerator E that enumerates the set of all and only strings accepted by N that have odd length.