HW6 : Computational Problems, Recognizability, Decidability

CSE105Sp23

Due: May 23th at 5pm (no penalty late submission until 8am next morning), via Gradescope

In this assignment: You will use general constructions and specific machines to explore the classes of recognizable and decidable languages. You will explore various ways to encode machines as strings so that computational problems can be recognized.

Resources: To review the topics you are working with for this assignment, see the class material from Week 6 through Week 7. We will post frequently asked questions and our answers to them in a pinned Piazza post.

Reading and extra practice problems: Chapter 4 exercises 4.1, 4.3, 4.4., 4.5. Chapter 4 Problems 4.29, 4.30.

Key Concepts: uring-recognizable languages, Turing-decidable languages, Church-Turing thesis, computational problems.

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 "hw6CSE105Sp23".

Assigned questions

1. **Explicit encodings** (8 points):
   In a computational problem, the elements of the language are encodings of machines. For example, consider the language

   $$E_{\text{DFA}} := \{\langle M\rangle \mid M \text{ is a DFA, and } L(M) = \varnothing\}$$

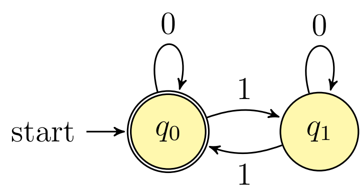
   where each string $\langle M\rangle$ in the language encodes a DFA $M = (Q, \Sigma, \delta, q_0, F)$. Usually, we purposefully drop the details about how this encoding is done because they can distract from the central computational properties of the language. In fact, any encoding can be used so long as there exists a decider for syntactic questions about the DFAs being encoded. In this question, we will build some specific explicit examples of encodings of DFAs to get more comfortable with these ideas.

(a) (*Graded for completeness*) [1] *Encoding with delimiters:* Perhaps the most straightforward way to create an encoding is to have it mirror the structure of the tuple $(Q, \Sigma, \delta, q_0, F)$ for the DFA. Your task: describe an encoding that maps each DFA M to a distinct string $\langle M \rangle$ that uniquely identifies M. That is, if you "decode" the encoding, you get the exact same machine back.

Hints, tips, notes of caution:

- You may use special characters like # and $ as delimiters in your encoding to separate the various components.
- Your encoding alphabet must be finite

(b) (*Graded for completeness*) Use your encoding from part (a) to produce the string encoding the DFA below:



(c) (*Graded for completeness*) Show that it is possible to have the same kind of delimited encoding without using special delimiter characters. In particular, prove that for every DFA M, we can assume that $\langle M \rangle \subseteq \{0, 1\}^*$.

Challenge; not graded:
For the delimited encoding schemes above, there are strings over the encoding alphabet (Σ) that nevertheless do not correspond to a valid DFA.

Prove/disprove: There exists an encoding scheme for which this is not true; that is,

$$\{\langle M \rangle \mid M \textit{ is a DFA}\} = \Sigma^*.$$

2. **Closure** (18 points):
Let $\Sigma = \{0, 1\}$ and $\Gamma = \{0, 1, 2\}$. Recall the functions

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

[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.

(a) (*Graded for correctness*) [2] Prove that, given any deterministic decider over Σ, M_L, there is a deterministic decider over Γ that recognizes

$$\text{Rep}(\ L(M_L)\ )$$

In other words, you will prove that for any Turing-decidable language L over Σ, $\text{Rep}(L)$ is also Turing-decidable. A complete answer will include both a precise construction of the machine and a (brief) justification of why this machine works as required.

(b) (*Graded for correctness*) Prove that, given any nondeterministic Turing machine over Γ, N_L, there is a nondeterministic Turing machine over Γ that recognizes

$$\text{Substring}(\ L(N_L)\ )$$

In other words, you will prove that the class of Turing-recognizable languages over Γ is closed under the Substring operation. A complete answer will include both a precise construction of the machine and a (brief) justification of why this machine works as required.

(c) (*Graded for completeness*) Give a different proof that the class of Turing-recognizable languages over Γ is closed under the Substring operation, this time using only deterministic Turing machines. A complete answer will include both a precise construction of the machine and a (brief) justification of why this machine works as required.

3. **Computational problems** (24 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*) For each regular language K, the language

$$\{\langle M\rangle \mid M \text{ is a DFA and } L(M) = K\}$$

is decidable.

(b) (*Graded for correctness*) For each regular language L, the language

$$\{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are both DFA and } L(M_1) \subseteq L \text{ and } L(M_2) \subseteq \overline{L}\}$$

is decidable.

(c) (*Graded for correctness*) Let $\text{Model} \in \{\text{DFA}, \text{NFA}, \text{REX}, \text{CFG}, \text{PDA}\}$. If EQ_{Model} is decidable, then E_{Model} is decidable.

Challenge; not graded: Let $\text{Model} \in \{\text{DFA}, \text{NFA}, \text{REX}, \text{CFG}, \text{PDA}\}$*. If* A_{Model} *is decidable, then* EQ_{Model} *is decidable.*

[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.