HW5: Recognizability, Decidability, Undecidability, and Reductions

CSE105Sp22

Due: 5/26/22 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. You will use general constructions and specific machines to explore the classes of recognizable, decidable, and undecidable languages. You will use computable functions to relate the difficult levels of languages via mapping reduction.

Resources: To review the topics you are working with for this assignment, see the class material from Weeks 6, 7, 8. 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, 4.32. Chapter 5 exercises 5.4, 5.5, 5.6, 5.7. Chapter 5 problems 5.10, 5.11, 5.16, 5.18.

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, computational problems, diagonalization, undecidability, unrecognizability, computable function, mapping reduction.

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 we will use this quarter 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 "HW5CSE105Sp22".

Assigned questions

1. (*Graded for correctness*[1])

    (a) Give an example of a decidable language L_1 whose complement is also decidable. A complete solution will include either (1) a precise definition of the example language L_1 and an explanation of why it is decidable and why its complement is decidable, or (2) a sufficiently general and correct argument for why there is no way to choose an example language to satisfy this requirement. All justifications and arguments should connect to the relevant definitions and the specific concepts being discussed.

    (b) Give an example of a decidable language L_2 and a Turing machine M_2 such that $L(M_2) = L_2$ but M_2 does not decide L_2. A complete solution will include either (1) precise definitions of L_2 and M_2 and justifications for why $L(M_2) = L_2$ and why M_2 does not decide L_2, or (2) a sufficiently general and correct argument for why there is no way to choose such a language and machine. For any machines you discuss, you can choose whether to use high-level descriptions, implementation level descriptions, or formal definitions. All justifications and arguments should connect to the relevant definitions and the specific concepts being discussed.

2. (*Graded for fair effort completeness*[2])

    Recall that a set X is said to be **closed** under an operation OP if, for any elements in X, applying OP to them gives an element in X. For example, the set of integers is closed under multiplication because if we take any two integers, their product is also an integer.

    Suppose M_1 and M_2 are Turing machines. Consider the following high-level descriptions of machines that give general constructions based on M_1 and M_2.

    (a) Consider the following construction of a nondeterministic Turing machine:

    "On input w
    1. Nondeterministically split w into two pieces, i.e. choose x, y such that $w = xy$.
    2. Simulate running M_1 on x.
    3. Simulate running M_2 on y.
    4. If both simulations in steps 2 and 3 accept, accept."

---

[1]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.

[2]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.

Can this construction be used to prove that the class of Turing-recognizable languages is closed under concatenation? Briefly justify your answer.

(b) Consider the following construction of an enumerator:

"Without any input
1. Build an enumerator E_1 that is equivalent to M_1.
2. Build an enumerator E_2 that is equivalent to M_2.
3. Start E_1 running and start E_2 running.
4. Initialize a list of all strings that have been printed by E_1. Declare the variable n_1 to be the length of this list (initially $n_1 = 0$).
5. Initialize a list of all strings that have been printed by E_2 so far. Declare the variable n_2 to be the length of this list (initially $n_2 = 0$).
6. Every time a new string x is printed by E_1:
7.   Add this string to the list of strings printed by E_1 so far.
8.   Increment n_1 so it stores the current length of the list.
9.   For $j = 1 \ldots n_2$,
10.   Let w_j be the jth string in the list of strings printed by E_2
11.   Print xw_j.
12. Every time a new string y is printed by E_2:
13.   Add this string to the list of strings printed by E_2 so far.
14.   Increment n_2 so it stores the current length of the list.
15.   For $i = 1 \ldots n_1$,
16.   Let u_i be the ith string in the list of strings printed by E_1
17.   Print $u_i y$."

Can this construction be used to prove that the class of Turing-recognizable languages is closed under concatenation? Briefly justify your answer.

(c) Consider the following construction of a Turing machine:

"On input w
1. Let $n = |w|$.
2. Create a two dimensional array of strings $s_{m,j}$ where $0 \le m \le n$ and $0 \le j \le 1$.
3. For each $0 \le m \le n$, initialize $s_{m,0}$ to be the prefix of w of length m and $s_{m,1}$ to be the suffix of w of length $n - m$. In other words, $w = s_{m,0}s_{m,1}$ and $|s_{m,0}| = m$, $|s_{m,1}| = n - m$.
4. For $i = 1, 2, \ldots$
5.   For $k = 0, \ldots, i$
6.   Run M_1 on $s_{\min(k,n),0}$ for (at most) i steps.
7.   Run M_2 on $s_{\min(k,n),1}$ for (at most) i steps.
8.   If both simulations in steps 6 and 7 accept, accept."

Can this construction be used to prove that the class of Turing-recognizable languages is closed under concatenation? Briefly justify your answer.
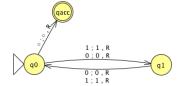
3. (*Graded for fair effort completeness*) Recall that

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a Turing machine, } w \text{ is a string, and } w \in L(M)\}$$

and

$$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a Turing machine, } w \text{ is a string, and } M \text{ halts on } w\}$$

Consider the Turing machines below, with input alphabet $\Sigma = \{0, 1\}$, tape alphabet $\{0, 1, \_\}$, and state diagrams (with the usual conventions):

(a) Give an example string that is in both A_{TM} and $HALT_{TM}$ and that is related to one of the two Turing machines whose state diagrams are given above, or explain why there is no such string.

(b) Give an example string that is in A_{TM} and is not in $HALT_{TM}$ and that is related to one of the two Turing machines whose state diagrams are given above, or explain why there is no such string.

(c) Give an example string that is not in A_{TM} and is in $HALT_{TM}$ and that is related to one of the two Turing machines whose state diagrams are given above, or explain why there is no such string.

4. (*Graded for correctness*) Fix $\Sigma = \{0, 1\}$ for this question. For each part below, you can choose sets from the following list:

$$\emptyset, A_{TM}, \overline{A_{TM}}, HALT_{TM}, \overline{HALT_{TM}}, E_{TM}, \overline{E_{TM}}, EQ_{TM}, \overline{EQ_{TM}}, \Sigma^*$$

You may use each set from the list **at most once** in the examples below. In particular, you can't choose $A = B = C = D = X = Y = \Sigma^*$.

(a) Find sets A, B for which the computable function

$$F = \text{``On input } x$$

1. Output $\langle$ $, 00\rangle$."

witnesses the mapping reduction $A \leq_m B$. Justify your answer by proving that, for all strings x, $x \in A$ iff $F(x) \in B$. If no such sets exist, justify why not.

(b) Find sets C, D for which the computable function

$G =$ "On input x

  1. Check if $x = \langle M, w \rangle$ for M a Turing machine and w a string. If so, go to step 3.

  2. If not, output $\langle$ [diagram] , [diagram] $\rangle$.

  3. Construct the Turing machine $M'_x =$ "On input y,

      1. If y has a positive and odd length, reject.

      2. Else, if y has a positive and even length, accept.

      3. Otherwise, run M on w and, if the computation halts, accept y."

  4. Output $\langle M'_x, $ [diagram] $\rangle$."

witnesses the mapping reduction $C \leq_m D$. Justify your answer by proving that, for all strings x, $x \in C$ iff $G(x) \in D$. If no such sets exist, justify why not.

(c) Find sets X, Y for which the computable function

$H =$ "On input x

  1. Check if $x = \langle M, w \rangle$ for M a Turing machine and w a string. If so, go to step 3.

  2. If not, output $\langle$ [diagram] $\rangle$.

  3. Construct the Turing machine $M'_x =$ "On input y,

      1. If $y \neq w$, reject.

      2. Otherwise, run M on w.

      3. If M accepts, accept. If M rejects, reject."

  4. Output $\langle M'_x \rangle$."

witnesses a mapping reduction $X \leq_m Y$. Justify your answer by proving that, for all strings x, $x \in X$ iff $H(x) \in Y$. If no such sets exist, justify why not.