

Homework 6 – Due Thursday, March 19, 2026 at 11:59 PM

Reminder Collaboration is permitted, but you must write the solutions by yourself without assistance, and be ready to explain them orally to the course staff if asked. You must also identify your collaborators and write “Collaborators: none” if you worked by yourself. Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden. Collaboration is not allowed on problems marked “INDIVIDUAL.”

Problems There are 4 required problems.

1. (Nondeterministic Turing machines)

- (a) Give a high-level description of a **nondeterministic** (multi-tape) TM deciding the following language over $\{0, 1, \#\}$: $\{a_1\#a_2\#\dots\#a_k \mid k, m \geq 0, a_i \in \{0, 1\}^m \text{ for every } i = 1, \dots, k, \text{ and there exists a subset } S \subseteq [k] \text{ such that for every } j \in [m], \text{ we have } (a_i)_j = 1 \text{ for exactly one index } i \in S\}$.

For example, the string $010\#110\#001\#111$ is in the language because we can choose $S = \{2, 3\}$ as our subset of indices. (It would also work to choose $S = \{4\}$.) On the other hand, the string $010\#110\#100\#000$ is not in the language because there is no way to choose such an appropriate S .

Explain why your NTM is correct.

Note: It is possible to do this with a deterministic TM, but we want to give you practice with the concept of nondeterminism. So your solution must use an NTM’s ability to nondeterministically guess in a meaningful way.

- (b) For languages A, B defined over the same alphabet Σ , define the operation

$$\text{Interlace}(A, B) = \{x_1y_1x_2y_2\dots x_ky_k \mid k \geq 0, x_1, \dots, x_k \in A, y_1, \dots, y_k \in B, \text{ where every } x_i, y_i \in \Sigma^*\}.$$

Given Turing machines recognizing languages A and B , give a high-level description of a nondeterministic (multi-tape) TM recognizing $\text{Interlace}(A, B)$. Again, your solution must use nondeterminism in a meaningful way. Explain why your NTM is correct.

- (c) Explain why part (b) implies that the Turing-recognizable languages are closed under the Interlace operation.
- (d) Explain (briefly!) what changes you could make to the construction or analysis in parts (b) and (c) to show that the **decidable** languages are closed under the Interlace operation.

Hint: Recall that a nondeterministic TM is a decider if it halts on every input, on every computation branch. The class of languages decided by NTMs is exactly the class of decidable languages.

2. (ALL_{REX}) Consider the following computational problem: Given a regular expression R (over some alphabet Σ), is the language generated by R equal to Σ^* ?

- (a) Formulate this problem as a language ALL_{REX} , in the style of the languages described in Sipser Chapter 4.1.
- (b) Show that ALL_{REX} is decidable. Give a high-level description of a Turing machine that decides this language, as well as an explanation of why it works.

Hint: Following the examples in Sipser Chapter 4.1, you may assume that the procedures we've seen in class for converting back and forth between automata and regular expressions can be implemented on Turing machines, and just cite these constructions. You may also use the fact that

$$E_{\text{DFA}} = \{\langle D \rangle \mid D \text{ is a DFA such that } L(D) = \emptyset\}$$

is decidable as stated in Theorem 4.4.

3. (**Universal TM**) In this short programming exercise, you will (sort of) build the universal Turing machine by essentially recreating Morphet. Hopefully this makes the idea of designing a Turing machine that takes another Turing machine as input less disturbing. Recall the language $A_{\text{TM}} = \{\langle M, w \rangle \mid M \text{ is a TM accepting input } w\}$. Determining membership in this language corresponds to the computational problem of determining whether TM M accepts input w .

The file `universal_TM.py` contains starter code that will help you implement a ~~Turing machine~~ Python program recognizing this language. Implement a program that prompts the user for an (appropriately encoded) TM-with-stay-put M working on singly infinite tape (that is, the input is padded with infinitely many trailing blank spaces to its right only) and a binary string w , outputting i) the sequence of configurations M enters when run on w , and ii) whether M accepts or rejects input w , if it terminates. Your program is (necessarily) allowed to run forever if M runs forever on w . The starter code file describes the expected syntax for the input and output of your solution.

This is not a software engineering class, so your program is allowed to fail arbitrarily (including failing silently) if its inputs do not correctly encode a TM and a binary string.

If you really don't like Python, you can implement your program in another high-level programming language (Java, C++, Haskell, ...) that the grading staff can read. (No Malbolge, please.) The downside is that you won't have the starter code to parse the input for you, and our autograder won't give you "instant" feedback on your submission.

4. (**Code as data**) Consider the following description of a three-tape TM H .

Algorithm 0: $H(\langle M, N \rangle)$

Input : Binary encoding of two basic TMs M and N , i.e., $\langle M, N \rangle \in \{0, 1\}^*$

1. Copy the string $\langle M \rangle$ to tape 2.
 2. Copy the string $\langle N \rangle$ to tape 3.
 3. Repeat the following three steps forever:
 4. Simulate N for one step on tape 2.
 5. Simulate M for one step on tape 3.
 6. If either machine accepts, accept. If both machines have rejected, reject. Else, continue.
-

Also, define the following three TMs, whose descriptions will be supplied as input to H .

Algorithm 1: $M_1(x)$

Input : String $x \in \{0, 1\}^*$

1. Ignore the input x and reject.
-

Algorithm 2: $M_2(x)$

Input : String $x \in \{0, 1\}^*$

1. If $x = \langle M_1 \rangle$, accept. Otherwise, reject.
-

Algorithm 3: $M_3(x)$

Input : String $x \in \{0, 1\}^*$

1. Let M be the TM such that $\langle M \rangle = x$.
 2. For $i = 1, 2, 3, \dots$:
 3. For each string y where $|y| \leq i$:
 4. Run M on input y for i steps. If it accepts, accept.
-

- (a) What is $L(M_1)$?
- (b) What is $L(M_2)$?
- (c) What is $L(M_3)$?
- (d) Is $\langle M_1, M_2 \rangle \in L(H)$? Explain why or why not.
- (e) Is $\langle M_1, M_3 \rangle \in L(H)$? Explain why or why not.
- (f) What is the language $L(H)$ recognized by H ?
- (g) Is H a decider for the language $L(H)$? Explain why or why not.