HW 11 Due: April 18^{th} 2025

1. Consider a new type of *deterministic* machine, having one read-only input tape and two stacks. The tape is read-only, it cannot be written, but the head can move left, right, or do nothing. Each stack operates, independently of the other, as in a deterministic pushdown automaton:

$$M = (Q, \Sigma, \Gamma_1, \Gamma_2, z_1, z_2, \delta, s)$$

where Q is a finite set of states, Σ is an input alphabet, Γ_1 and Γ_2 are two stack alphabets (Γ_1 for the first stack, Γ_2 for the second stack), $z_1 \in \Gamma_1$ and $z_2 \in \Gamma_2$ are the initial symbols for the two stacks, $s \in Q$ is the initial state. Just like in a Turing machine, h is a special halting state not in Q.

(a) Give an appropriate definition for the transition function δ, for a configuration of this machine, for the "yields in one step" operator, and for the language accepted by this machine.
Answer. The transition function can be defined as:

$$\delta : Q \times \Sigma \times \Gamma_1 \times \Gamma_2 \to (Q \cup \{h\}) \times \{\mathrm{L}, \mathrm{R}, \mathrm{S}\} \times \Gamma_1 \times \Gamma_2$$

Where:

- L, R, S indicate left, right, or stay for the tape head.

A configuration is a 5-tuple:

$$(q, x_1\underline{\alpha}y_1, \beta, \gamma)$$

Where:

- $q \in Q$ is the current state,
- $\alpha \in \Sigma^*$ is the content of the cell the tape head is pointing at,
- $x_1 \in \Sigma^*$ is the entire contents of the tape to the left of the head,
- $y_1 \in \Sigma^*$ is the entire contents of the tape to the right of the head up to the first # (excluded) of the infinite run of #'s on the tape
- $\beta \in \Gamma_1$ is the current content of the top of the stack 1,
- $\gamma \in \Gamma_2$ is the content of the top of the stack 2.

Yields in One Step ($\vdash$):

$$(q, x_1\underline{\alpha}y_1, \beta, \gamma) \vdash (q, x_2\underline{\alpha}y_2, \beta', \gamma')$$

- **(GO LEFT)**: $\delta(q_1, a_1, \beta_1, \gamma_1) = (q_2, L, \beta_2, \gamma_2)$, where $x_1 = x_2a_2 \wedge (a_1 = \# \wedge y_1 = \epsilon \wedge y_2 = \epsilon \vee a_1y_1 \neq \# \wedge y_2 = a_1y_1)$
- **(GO RIGHT)**: $\delta(q_1, a_1, \beta_1, \gamma_1) = (q_2, R, \beta_2, \gamma_2)$, where $x_2 = x_1a_1 \wedge (y_1 = \epsilon \wedge a_2 = \# \vee y_1 = a_2y_2 \neq \epsilon)$
- **(GO RIGHT)**: $\delta(q_1, a_1, \beta_1, \gamma_1) = (q_2, S, \beta_2, \gamma_2)$, where $(x_2 = x_1) \wedge (y_2 = y_1)$

(b) These machines can accept the same languages as a class of automata you already know: deterministic pushdown automata, pushdown automata, or Turing machines? Give a detailed *sketch* justifying your answer.

Answer. We will show this machine is equivalent in power to a Turing machine.

i. Use one stack to represent the left of the TM tape,

ii. Use the other stack to represent the right.

iii. Simulate left and right movements by popping/pushing between the stacks. To access a symbol from the first stack, we pop elements above it and push them onto the second stack—preserving all content which can be done as the tape head is moving. This setup simulates a Turing Machine's tape, where the head's position corresponds to the interface between the two stacks.

2. Use reduction to prove that the language $L = \{\rho(M_1)\rho(M_2) : L(M_1) \cup L(M_2) = \Sigma^*\}$ is not decidable (you may assume that M_1 and M_2 have the same input alphabet Σ).

   Answer. We have already established the undecidability of K_0, the problem of determining whether a Turing machine accepts a given input w.

$$K_0 = \{\rho(M)\rho(x) : M \searrow x\}.$$

By proving by contradiction, we will show that if L is decidable, then K_0 is also decidable.

Prove by Contradiction. Let R be a TM that decides L. We'll construct TM S to decide K_0 by working in the following manner:

$S =$ "On input $\langle M_1, w\rangle$, where M_1 is a TM and w is a string:

1. Run R on input $<M_1, M_2>$, where M_2 is a TM that rejects w and accepts everything else.
    1. If R accepts, *accept*;
    2. if R rejects, *reject*."

If R accepts, then $L(M_1) \cup L(M_2) = \Sigma^*$. Since $L(M_2) = \Sigma^* \backslash w$, this implies $w \in L(M_1)$, so S accepts. If R rejects, then $w \notin L(M_1)$ and S rejects. Thus, S decides K_0 using R, implying K_0 is decidable—contradicting the known undecidability of K_0. Therefore, L is not decidable.

3. Are the following languages Turing-decidable, Turing-acceptable but not Turing-decidable, or not even Turing-acceptable? Justify your answer carefully in each case (e.g., using a reduction). These exercises are not as simple as they look.

(a) $L_1 = \{\rho(M)\rho(C) : M \searrow \rho(M),\ C \text{ is the computation history of } M \text{ on } \rho(M)\}$

Answer. We construct a decider R for L_1:

1. On input $\langle w\rangle$:
    1. Parse the prefix of w as $\rho(M)$, the encoding of a Turing machine M.
    2. Verify the prefix matches a valid TM encoding. If not, reject.
    3. Simulate M on input $\rho(M)$, step by step.
    4. At each step, compare the configuration with the corresponding segment of the suffix of w.
    5. If the simulation completes and matches all of w, accept. Otherwise, reject.

(b) $L_2 = \{\rho(M)\rho(C) : M \searrow \rho(M), C \text{ is the computation of } M \text{ on } \rho(M) \vee M \nearrow \rho(M), C = \epsilon\}$

Answer. We have already established the undecidability of K_1, the problem of determining whether a Turing machine accepts its own encoding.

$$K_1 = \{\rho(M) : M \searrow \rho(M)\}.$$

By proving by contradiction, we will show that if L_2 is decidable, then K_1 is also decidable.

Prove by Contradiction. Assume, for contradiction, that L_2 is decidable. Then there exists a TM R that decides L_2.

We'll construct TM S to decide K_1 by working in the following manner:

$S =$ "On input $\langle \rho(M) \rangle$:

i. Run R on input $\langle \rho(M)\rho(\epsilon) \rangle$.
ii. If R accepts; reject.
iii. If R rejects; accept.

If $M \nearrow \rho(M)$, then R will accept $\rho(M)\rho(\epsilon)$ since the computation history is empty. If $M \searrow \rho(M)$, then the correct computation history C must differ from ϵ, so R will reject the input. Thus, S decides K_1, which contradicts its known undecidability. Therefore, L_2 must be undecidable.

Now, we have already established the unrecognizability of $\overline{K_1}$, the set of Turing machines that loop forever on their own encoding:

$$\overline{K_1} = \{\rho(M) : M \nearrow \rho(M)\}.$$

We will now prove by contradiction that if L_2 is recognizable, then $\overline{K_1}$ is also recognizable.

Proof by Contradiction. Assume, for contradiction, that L_2 is recognizable. Then there exists a TM R that recognizes L_2.

We construct a TM S to recognize $\overline{K_1}$ as follows:

$S =$ "On input $\langle \rho(M) \rangle$:

i. Run R on input $\langle \rho(M)\rho(\epsilon) \rangle$.
ii. If R accepts, then accept.

If $M \nearrow \rho(M)$, then $\rho(M)\rho(\epsilon) \in L_2$ and R accepts. Thus, S accepts. Therefore, S recognizes $\overline{K_1}$, contradicting the fact that $\overline{K_1}$ is not recognizable. Hence, L_2 is not recognizable.

(c) $\overline{L_2}$:

Answer. Since L_2 is not decidable, its complement, $\overline{L_2}$, is not even Turing recognizable.

Note: recall that a computation is a finite sequence of Turing machine configurations, each of them of the form (q, u, a, v), such that one configuration follows from the previous one through one application of the δ function, and such that the last configuration is a halting configuration. The format for the particular encoding $\rho(C)$ of the computation C is left unspecified because it is clearly irrelevant for your proofs: we only need to agree (and I hope we do) that such a format can be defined.