CSC4010: Introduction to Theoretical Computer Science
Problem Set 2

Remark: Each problem mentions which lectures it uses techniques from. If you would like feedback on your homework, you can either submit it in class or email it to both of us. If you find any errors, please let us know!

1. Show the following statements about decidable languages:
   (a) Show that if two languages L_1, L_2 are decidable then so is $L_1 \cap L_2 = \{x \mid x \in L_1, x \in L_2\}$.
   (b) Show that if two languages L_1, L_2 are decidable then so is $L_1 \circ L_2 = \{xy \mid x \in L_1, y \in L_2\}$.

   Note: The most important part of the proof is that you clearly explain what changes to the state set, alphabet, and/or transition function you have to make to prove these statements. You may use a variant of TM discussed in class to prove these.

2. In this course, the ordinary Turing machine has a tape that is infinite in both directions. A Turing machine with a one-way infinite tape is a variant whose tape has a leftmost cell and extends infinitely only to the right. Prove that ordinary (two-way infinite) TMs and one-way infinite TMs decide the same class of languages.

3. Give a Turing Machine that does the following:
   (a) On input 1^k, writes k in binary on the tape and halts. Describe the machine at the level of states and transitions.
   (b) Takes as input the binary representation $\langle x \rangle$ of a non-negative integer x, replaces it by $\langle x + 1 \rangle$, and halts.

4. In class, we described a Universal Turing Machine. Write a pseudocode describing the behavior of the Universal Turing Machine U on input $\langle M, w \rangle := \langle M \rangle w$, where $\langle M \rangle \in \{0,1\}^*$ is an encoding of a Turing machine M, and $w \in \{0,1\}^*$ is its input. Let $M = (Q, \Sigma, q_{\text{acc}}, q_{\text{rej}}, \delta, q_0, \sqcup)$, where $|Q| = 2^k$ and $|\Sigma| = 2^m$. Recall the following encoding conventions:

   1. $Q = \{0,1\}^k$.
   2. For every $b \in \Sigma$, $\langle b \rangle \in \{0,1\}^m$. In particular,$\langle 0 \rangle = 0^m, \langle 1 \rangle = 10^{m-1}, \langle \sqcup \rangle = 1^m$.
   3. $\langle L \rangle = 0$, and $\langle R \rangle = 1$.
   4. $\langle q, b \rangle = q\langle b \rangle$, and $\langle q, b, D \rangle = q\langle b \rangle\langle D \rangle$, for $q \in Q$, $b \in \Sigma$, and $D \in \{L, R\}$.
   5. If $\delta(q, b) = (q', b', D)$, then $\langle \delta(q, b) \rangle = q'\langle b' \rangle\langle D \rangle$.
   6. $\langle \delta \rangle$ is the concatenation of the records $\langle q, b \rangle\langle \delta(q, b) \rangle$ for every $q \in Q$ and $b \in \Sigma$, and
   7. $\langle M \rangle = 1^k 0 1^m 0 \langle \delta \rangle$.

   Assume that the encodings of q_0, q_{acc}, and q_{rej} are fixed and known to U. The transition table contains exactly one record for each pair (q, b). Write pseudocode for U that:

   1. decodes $\langle M, w \rangle$;
   2. simulates the transitions of M one at a time; and
   3. accepts, rejects, or runs forever exactly as M does on input w.

   You may assume that the input is a valid encoding. Alternatively, specify what U does on an invalid encoding.

Solutions

1. Let $M_i = (Q_i, \Sigma_i, \delta_i, \sqcup, q_{acc,i}, q_{rej,i}, q_{0,i})$ for $i = 1, 2$, be deciders for L_1, L_2, respectively.

   (a) We construct a two-tape TM M deciding $L = L_1 \cap L_2$, with tape i using alphabet Σ_i.

   Initially, M is in state q_0, with the input string $w \in \{0,1\}^*$ on the first tape and the second tape blank. It copies w onto the second tape, moves both heads to the blank immediately before their respective copies of w, and enters state $(q_{0,1}, q_{0,2})$. It then simulates M_1 on the first tape and M_2 on the second tape.

   The machine M is described as follows.

   (i) **State set.** Let $Q = Q_{\text{init}} \cup (Q_1 \times Q_2) \cup \{q_{acc}, q_{rej}\}$, where Q_{init} contains q_0 and the auxiliary states needed for copying the input on the second tape and repositioning the heads. The set of auxiliary states and the final halting state is disjoint from $Q_1 \times Q_2$.

   (ii) **Transition function.** If M is in a non-halting state, then the transition function is described as follows.

   $$\delta : (Q \setminus \{q_{acc}, q_{rej}\}) \times \Sigma_1 \times \Sigma_2 \longrightarrow Q \times \Sigma_1 \times \Sigma_2 \times \{L, R, S\}^3,$$

   where on S the head stays in the current position. Besides the initialization transitions described above, define the simulation transitions as follows, for $a \in \Sigma_1$ and $b \in \Sigma_2$.

   - **Neither simulated machine has halted.** If q_1, q_2 are nonhalting states and

   $$\delta_1(q_1, a) = (q_1', a', D_1), \qquad \delta_2(q_2, b) = (q_2', b', D_2),$$

   then $\delta((q_1, q_2), a, b) = ((q_1', q_2'), a', b', D_1, D_2)$.
   - **One of the simulated machines has rejected.** For any $q_1 \in Q_1$ and $q_2 \in Q_2$,

   $$\delta((q_{rej,1}, q_2), a, b) = (q_{rej}, a, b, S, S),$$

   $$\delta((q_1, q_{rej,2}), a, b) = (q_{rej}, a, b, S, S).$$

   - **Both simulated machines have accepted.** $\delta((q_{acc,1}, q_{acc,2}), a, b) = (q_{acc}, a, b, S, S)$.
   - **Only M_1 has accepted.** If q_2 is nonhalting and $\delta_2(q_2, b) = (q_2', b', D_2)$, then $\delta((q_{acc,1}, q_2), a, b) = ((q_{acc,1}, q_2'), a, b', S, D_2)$. Thus, the first tape and head remain unchanged while M_2 continues.
   - **Only M_2 has accepted.** If q_1 is nonhalting and $\delta_1(q_1, a) = (q_1', a', D_1)$, then $\delta((q_1, q_{acc,2}), a, b) = ((q_1', q_{acc,2}), a', b, D_1, S)$. Thus, the second tape and head remain unchanged while M_1 continues.

   Correctness and termination. Since M_1 and M_2 are deciders, each halts on w. If either simulated computation rejects, M rejects; otherwise, both eventually accept, and M accepts. Therefore, M halts on every input, and M accepts $w \iff w \in L_1$ and $w \in L_2 \iff w \in L_1 \cap L_2$. Hence M decides $L_1 \cap L_2$.

   (b) We construct a three-tape TM M deciding $L = L_1 L_2$, with tape $i+1$ using alphabet Σ_i for $i = 1, 2$. The first tape contains input w and a special symbol indicating the split currently being considered. Let $w = w_1 \cdots w_n$. The algorithm has at most $n+1$ phases. In phase k, where $1 \le k \le n+1$, it tests the split $u_k = w_1 \cdots w_{k-1}$, and $v_k = w_k \cdots w_n$, where $u_1 = \varepsilon$ and $v_{n+1} = \varepsilon$. It simulates M_1 on u_k on tape 2, and M_2 on v_k on tape 3.

   For $1 \le k \le n$, the split is recorded by replacing w_k on the first tape with $\bar{w}_k$. For $k = n+1$, the blank immediately after w is replaced with $\overline{\sqcup}$. Thus, the first tape uses the alphabet $\Sigma = \{0, 1, \sqcup, \bar{0}, \bar{1}, \overline{\sqcup}\}$. Within a phase, when we are simulating M_1, M_2 only heads two and three move, and the first head stays put.

   If both simulations accept, M accepts. If either simulation rejects, phase k fails and the other simulation is abandoned. If this is the final phase, M rejects; otherwise, it prepares for phase $k+1$.

   Initially, M marks the first input symbol, or the first input square with $\overline{\sqcup}$ if $w = \varepsilon$. It copies ε on tape 2, and w on tape 3, it places both heads two and three on the blank immediately before their respective strings. It places the first head on the marked symbol, and enters state $(q_{0,1}, q_{0,2})$.

   The machine M is described as follows.

(i) **State set.** Let $Q = Q_{\text{init}} \cup (Q_1 \times Q_2) \cup \{q_{acc}, q_{rej}\}$, where Q_{init} contains q_0, a state q_{next}, and the auxiliary states needed for copying, erasing the simulated workspaces, repositioning the heads, and preparing the next phase. The set of auxiliary states and the final halting state is disjoint from $Q_1 \times Q_2$.

(ii) **Transition function.** If M is in a non-halting state, then the transition function is described as follows.

$$\delta : (Q \setminus \{q_{acc}, q_{rej}\}) \times \Sigma \times \Sigma_1 \times \Sigma_2 \longrightarrow Q \times \Sigma \times \Sigma_1 \times \Sigma_2 \times \{L, R, S\}^2,$$

where on S the head stays in the current position. Besides the initialization and auxiliary transitions, we describe how to simulate the behavior of M_1, M_2 as follows: let $a \in \Sigma$, $b \in \Sigma_1$, and $c \in \Sigma_2$.

- **Neither simulated machine has halted.** If q_1, q_2 are non-halting states and

$$\delta_1(q_1, b) = (q_1', b', D_1), \qquad \delta_2(q_2, c) = (q_2', c', D_2),$$

then $\delta((q_1, q_2), a, b, c) = ((q_1', q_2'), a, b', c', S, D_1, D_2)$.

- **One of the simulated machines has rejected.** For any $q_1 \in Q_1$ and $q_2 \in Q_2$,

$$\delta((q_{rej,1}, q_2), a, b, c) = (q_{\text{next}}, a, b, c, S, S, S), \qquad \delta((q_1, q_{rej,2}), a, b, c) = (q_{\text{next}}, a, b, c, S, S, S).$$

Thus, rejection by one of the simulated machines causes the current phase to fail, rather than immediately causing M to reject.

- **Both machines have accept.** $\delta((q_{acc,1}, q_{acc,2}), a, b, c) = (q_{acc}, a, b, c, S, S, S)$.
- **Only M_1 accepts.** If q_2 is nonhalting and $\delta_2(q_2, c) = (q_2', c', D_2)$, then $\delta((q_{acc,1}, q_2), a, b, c) = ((q_{acc,1}, q_2'), a, b, c', S, S, D_2)$.
- **Only M_2 accepts.** If q_1 is nonhalting and $\delta_1(q_1, b) = (q_1', b', D_1)$, then $\delta((q_1, q_{acc,2}), a, b, c) = ((q_1', q_{acc,2}), a, b', c, S, D_1, S)$.

(iii) **Going to the next phase.** In state q_{next}, the first head is still on the marked symbol.
If this symbol is $\overline{\sqcup}$, then the failed phase was phase $n+1$. Every split of w has been tested, so M enters q_{rej}.
Otherwise, M erases the contents of tapes 2 and 3 and resets their heads. It restores the marked symbol $\bar{w}_k$ to w_k and moves the mark one square to the right: the next symbol is replaced by its marked version if it is 0 or 1, and by $\overline{\sqcup}$ if it is blank.
It then copies $w_1 \cdots w_k$ onto tape 2 and $w_{k+1} \cdots w_n$ onto tape 3, interpreting marked input symbols as their unmarked versions. It places both heads two and three on the blank immediately before u_{k+1} and v_{k+1}, returns the first head to the marked symbol, enters state $(q_{0,1}, q_{0,2})$ and proceeds with phase $k+1$.

Halting. Each initialization, copying, and cleanup routine takes finitely many steps. Since M_1 and M_2 are deciders, each phase finishes in finitely many steps: either one simulation rejects, or both eventually accept. There are at most $n+1$ phases, so M halts on every input.

Correctness. In phase k, M faithfully simulates M_1 on u_k and M_2 on v_k. Thus, that phase succeeds exactly when $u_k \in L_1$ and $v_k \in L_2$. The machine tests every split of w unless an earlier split succeeds. Therefore,

$$M \text{ accepts } w \iff \exists k \in \{1, \ldots, n+1\} \text{ such that } u_k \in L_1 \text{ and } v_k \in L_2 \iff w \in L_1 L_2.$$

Hence M decides $L_1 L_2$.

2. Let M_p be the two-way infinite tape Turing Machine. At any instant the tape of M_p has the form

$$\cdots \ \sqcup \ \ a_{-n} \ \cdots \ a_{-1} \ \ a_0 \ \ a_1 \ \cdots \ a_m \ \ \sqcup \ \cdots$$

where a_0 is the symbol in the cell scanned by M_p initially. M_o represents these tape contents on a single one-way tape with *two tracks*:

a_0	a_1	a_2	$\cdots$	a_m	B	$\cdots$
$\#$	a_{-1}	a_{-2}	$\cdots$	a_{-n}	B	$\cdots$

The upper track holds the initial cell and everything to its right; the lower track holds the cells to the left of the initial cell, in *reverse* order, beginning with the marker $\#$. When M_p is at or to the right of the initial cell, M_o simulates it on the upper track; when M_p is to the left of the initial cell, M_o simulates it on the lower track, moving in the *opposite* direction (since that half of the tape is stored reversed). The initial configuration, on input $a_0 a_1 \cdots a_n$, is

a_0	a_1	$\cdots$	a_n
B	B	$\cdots$	B

Formally, $M_o = (K', \Sigma, \Gamma', \delta', q_0', F')$, where

$$K' = \{q_0'\} \cup (K \times \{1, 2\}), \qquad \Gamma' = \Gamma \times (\Gamma \cup \{\#\}), \qquad F' = \{(q, 1), (q, 2) \mid q \in F\}.$$

The second component of a state records which track the head is on: 1 for the upper track, 2 for the lower track.

The transition function δ' is defined as follows. Write a move of M_p as $\delta(q, a) = (q', c, D)$ with $D \in \{L, R\}$, and let $\bar{D}$ denote the opposite direction ($\bar{L} = R$, $\bar{R} = L$).

1. *Initial move.* M_o marks the first cell with $\#$ and moves right, choosing the track according to the first move of M_p:

$$\begin{aligned} \delta'(q_0', [a_0, B]) &= ((q', 1), [A, \#], R) && \text{if } \delta(q_0, a_0) = (q', A, R), \\ \delta'(q_0', [a_0, B]) &= ((q', 2), [A, \#], R) && \text{if } \delta(q_0, a_0) = (q', A, L). \end{aligned}$$

2. *M_p to the right of the initial cell (upper track).* For all $b \in \Gamma$:

$$\delta'((q, 1), [a, b]) = ((q', 1), [c, b], D).$$

   M_o simulates the move on the upper track, in the same direction, leaving the lower track unchanged.

3. *M_p to the left of the initial cell (lower track).* For all $a \in \Gamma$ and $b \in \Gamma$:

$$\delta'((q, 2), [a, b]) = ((q', 2), [a, c], \bar{D}).$$

   M_o simulates the move on the lower track, in the opposite direction, leaving the upper track unchanged.

4. *Crossing the initial cell (the cell containing $\#$).* Whichever track mode M_o is in, scanning $[a, \#]$ means M_p is scanning the initial cell $a_0 = a$. For $i \in \{1, 2\}$:

$$\begin{aligned} \delta'((q, i), [a, \#]) &= ((p, 1), [A, \#], R) && \text{if } \delta(q, a) = (p, A, R), \\ \delta'((q, i), [a, \#]) &= ((p, 2), [A, \#], R) && \text{if } \delta(q, a) = (p, A, L). \end{aligned}$$

   In both cases M_o moves right: if M_p moves right, M_o continues on the upper track; if M_p moves left, M_o switches to the lower track, where a rightward step of M_o corresponds to a leftward step of M_p.

3. (a) The TM is given in Figure 1. The machine starts in the initial state q_{start} with the head on the blank immediately before the input. The machine replaces $\square$ (the blank symbol) with the separator symbol $\#$. It initializes a binary counter to 0 on its left, and moves to the first input cell and the state q_{scan}. The machine repeatedly marks the leftmost unmarked unary 1 as X and increments the binary counter by one. When no more 1's remain, it enters the state q_{erase}. The machine erases all Xs and the separator $\#$, then halts in q_h with only $\langle k \rangle$ remaining and the head on its leftmost digit.

   (b) The corresponding TM is given in Figure 2.

4. In class, we described the Universal Turing Machine with four tapes:

   1. **Input Tape:** This contains $\langle M, w \rangle = \langle M \rangle w$, where M is the Turing machine we want to simulate, and w is the input to M.

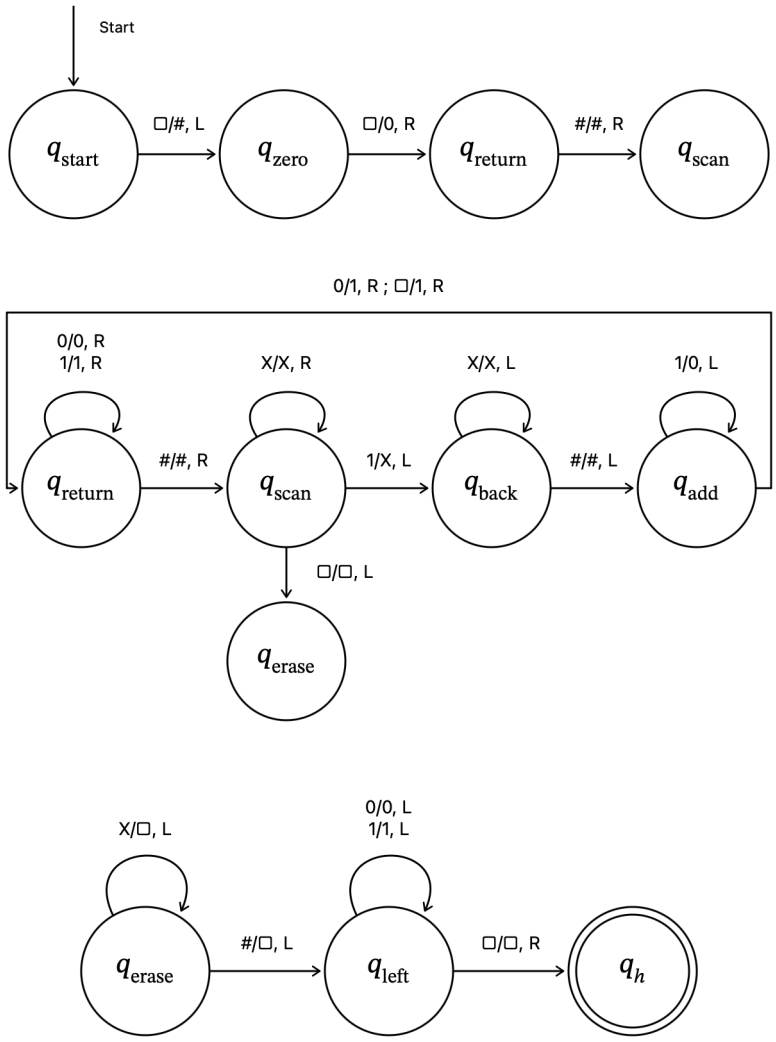


Figure 1: Unary to binary TM

2. **The state tape:** When we are simulating the behavior of M on w, this always contains the current state in which M is.
3. **Machine code tape:** This contains the encoding $\langle\delta\rangle$ of the transition function of the turing machine M.
4. **Simulation Tape:** This contains the the tape contents of M when it is being simulated on w.

Given input $\langle M\rangle w$, we first describe how U decodes it, and prepares to simulate M on w:

1. First, it deduces the number of states 2^k by counting the number of 1's until the first 0. It also deduces the alphabet-size of M (denoted 2^m) by counting the number of 1's between the first and the second 0.
2. At this point the U knows that $\langle\delta\rangle$ is given in the next $2^{k+m}(2k+2m+1)$ bits following the second 0. So it copies these entries to the machine code tape.
3. Having done this, it knows where w starts, so it copies w to the simulation tape. Since M's alphabet has size m, if $w_i = 0$, then we write 0^m in the m-bit block representing w_i. Similarly, if $w_i = 1$, we write 10^{m-1} in the m-bit block representing w_i.

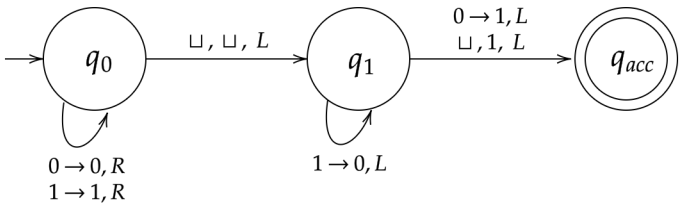


Figure 2: Add 1 TM

4. Finally, to simulate the initial state of M, we write down $q_0 \in \{0,1\}^k$ on the state tape. It also moves the simulation tape head to the beginning of the string $\langle w_1 \rangle \langle w_2 \rangle \cdots \langle w_n \rangle$.

At this point, U's state and simulation tapes exactly reflect M's initial configuration when simulated on w. Now suppose, inductively, U's state and simulation tapes exactly reflect M's configuration at time i. Suppose this non-halting configuration is $uqv = u_1 u_2 \cdots u_k q v_1 v_2 \cdots v_l$. This implies on U's state tape, q is written down and on its simulation tape, $\langle u_1 \rangle \langle u_2 \rangle \cdots \langle u_k \rangle \langle v_1 \rangle \cdots \langle v_l \rangle$ is written down. The simulation tape head is on the first symbol of the m-bit block of $\langle v_1 \rangle$. Suppose $\delta(q, v_1) = (q', v_1', D)$. To simulate the resulting configuration of M, the machine code head will locate the block containing $q\langle v_1 \rangle$. The next $2k + 2m + 1$ bits will contain $q' \langle v_1' \rangle \langle D \rangle$. It will write q' on the state tape after erasing q. It will replace $\langle v_1 \rangle$ with $\langle v_1' \rangle$ and move to the beginning of the adjacent block in direction D. If $q' = q_{acc}$ or $q' = q_{rej}$, then it will do the steps mentioned above and halt.

By induction we have shown: 1) that we can simulate the initial configuration of M. 2) If can sucessfully simulate the ith configuration of M, then we can simulate the $i + 1$th configuration of M. Therefore, if M rejects/accepts/loops, then the same is true of U.