CS 39: Theory of Computation | Winter '23

PSET 2 — 2023-01-23

Prof. Chakrabarti | ***Student: Amittai Siavava***

Credit Statement

I discussed ideas for this homework assignment with Paul Shin.

I also referred to the following books:

(a) **Introduction to the Theory of Computation** by **Michael Sipser**.

(b) **A Mathematical Introduction to Logic** by **Herbert Enderton**.

Problem 1.

Construct NFAs for the languages generated by each of the following regular expressions.

(a) $10 \cup (0 \cup 11)0^*1$

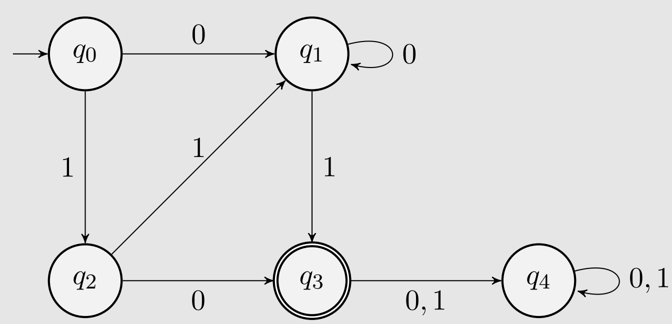

Figure 1.1. NFA to accept $10 \cup (0 \cup 11)0^*1$

(b) $(0 \cup 1)((0 \cup 1)(0 \cup 1))^* \cup ((0 \cup 1)(0 \cup 1)(0 \cup 1))^*$

Observation: the regular expression matches all strings of length n such that either $n \pmod 2 \equiv 1$ or $n \pmod 3 \equiv 0$.

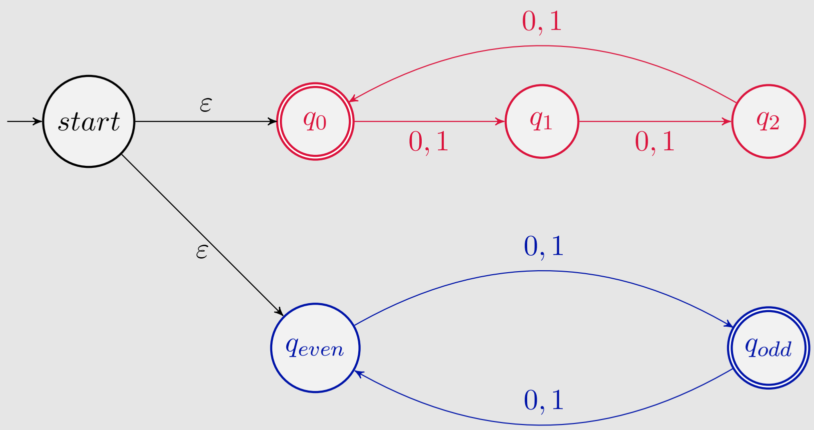

Figure 1.2. NFA to accept $(0 \cup 1)((0 \cup 1)(0 \cup 1))^* \cup ((0 \cup 1)(0 \cup 1)(0 \cup 1))^*$

Problem 2.

Give regular expressions for the following languages.

(a) $\{x \in \{0,1\}^* \ : \ x$ contains "000" or "111" (or both) as a substring $\}$.

$$(0 \cup 1)^*(000 \cup 111)(0 \cup 1)^*$$

(b) $\{x \in \{0,1\}^* \ : \ x$ contains both "000" and "111" as a substrings $\}$.

$$(0 \cup 1)^*((000(0 \cup 1)^*111) \cup (111(0 \cup 1)^*000))(0 \cup 1)^*$$

(c) $\{x \in \{0,1\}^* \ : \ x$ does not contain "111" as a substring $\}$.

$$(0^* \cup 0^*1 \cup 0^*11)(0^* \cup 00^*1 \cup 00^*11)^*$$

Problem 3.

Let L be the language over $\Sigma = \{a, b\}^*$ given by the regular expression $(ab \cup aab \cup aba)^*$.

(a) Design an NFA for L that has exactly 4 states, no ε-transitions, and exactly one a-transition out of its start state.

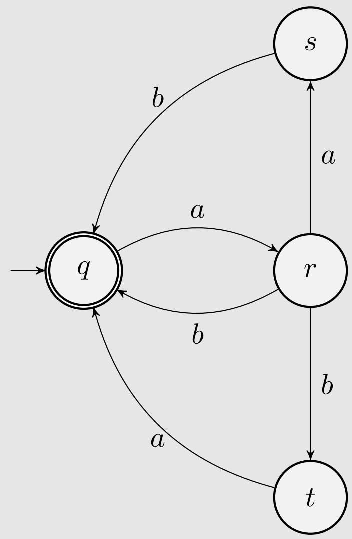

Figure 3.1. NFA to recognize $L \equiv (ab \cup aab \cup aba)^*$

(b) Convert the above NFA into a DFA for L by mechanically using the power set construction. For the sake of legibility, may avoid drawing transitions out of states that are unreachable from the start state of the resulting DFA. However, do draw every state of the power set construction. *I strongly recommend using state names like* rt*,* qst*, etc. in your drawing and not the more formal* $\{r, t\}$ *,* $\{q, s, t\}$*, etc.*

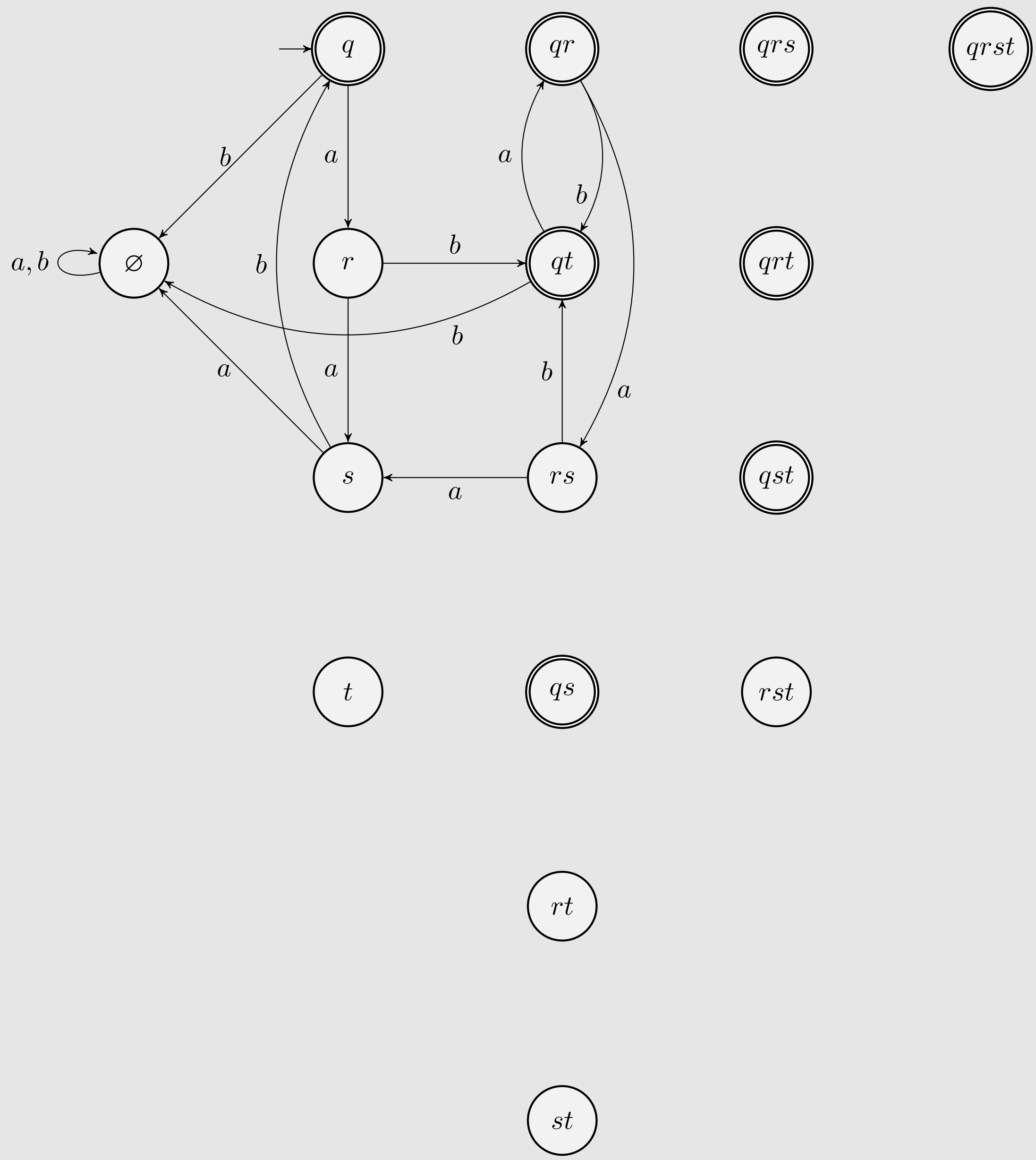

Figure 3.2. Seven-state DFA to recognize $L \equiv (ab \cup aab \cup aba)^*$

(c) Observe that exactly 7 states are reachable, so if you were to delete the unreachable ones, you would obtain a 7-state DFA for L. *Carefully observe this DFA and argue that two of its states can be replaced by a single state.* Do this and draw the resulting 6-state DFA for L.

The two states r and rs can be merged because the y are both non-accepting states and the transitions out of the two states are identical, i.e. $\delta(r, x) = \delta(rs, x)$ for all $x \in \Sigma$.

Resulting DFA:

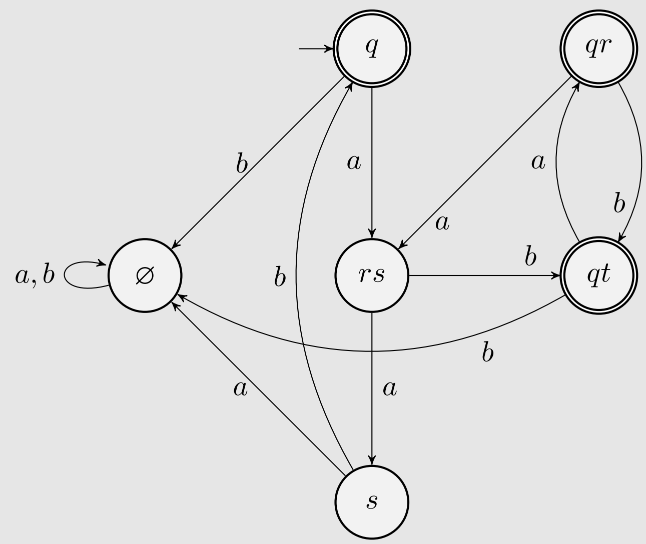

FIGURE 3.3. Six-state DFA to recognize $L \equiv (ab \cup aab \cup aba)^*$

(d) Give clear reasons why L cannot be recognized by a DFA with 5 or fewer states. For extra credit, write your reasoning as a formal proof.

Hint: Argue that any hypothetical DFA for L must treat the strings aa and ab differently in the sense that the state it reaches upon reading aa must be different from the state it reaches upon reading ab. Now extend this observation by identifying 6 specific strings (which may not all belong to L) that must all be treated differently.

Let M be a DFA that recognizes L. Let us define the function δ^* as follows:

$$\delta^* : Q \times \Sigma^* \to Q$$

$$\delta^*(q, x) = \begin{cases} q & \text{if } x = \varepsilon. \\ \delta^*(\delta(q, s), t) & \text{otherwise, where } q = st, s \in \Sigma, t \in \Sigma^*. \end{cases}$$

Then M must differentiate between six distinct strings as follows:

s	$\delta^*(q_0, s)$	$\delta^*(q0, sa)$	$\delta(q_0, sb)$	$\delta^*(q_0, sab)$
a	rejecting	rejecting	accepting	accepting
b	rejecting	rejecting	rejecting	
aa	rejecting	rejecting	accepting	rejecting
ab	accepting	accepting	rejecting	
aab	accepting	rejecting	rejecting	
aba	accepting	rejecting	accepting	

TABLE 1. Summary of the δ-transitions for M

As we can see, the state that M transitions to upon reading any of the six strings must be different from the state it transitions to upon reading any of the other strings because of a combination of different accepting/rejecting status and different outgoing transitions.

Problem 4.

(a) Let L be a nonempty language recognized by an NFA N. Prove that there exists an NFA N_1 that has exactly one accept state and recognizes the same language L.

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA that recognizes L.

We define $N_1 = (Q \cup \{a\}, \Sigma, \delta_1, q_0, F_1)$, where a is a new state that is not in Q, and:

$$F_1 = \{a\}$$

$$\delta_1(q, x) = \begin{cases} \delta(q, x) \cup \{a\} & \text{if } q \in F \text{ and } x = \varepsilon \\ \delta(q, x) & \text{otherwise.} \end{cases}$$

Claim 4.1. N_1 is an NFA that recognizes L.

Proof. Recall that the original NFA N recognizes L.

We aim to show that $\mathcal{L}(N_1) = L$; essentially, that $\mathcal{L}(N_1) \supseteq L$ (completeness) and $\mathcal{L}(N_1) \subseteq L$ (soundness).

(i) **Completeness** ($\mathcal{L}(N_1) \supseteq L$)

Let $w = w_1 w_2 \dots w_k : w_i \in \Sigma$ for all $0 < i \leq k$ be a string in L. Since N recognizes L, there exists a computation path $p = q_0 q_1 \dots q_n$ of N on w such that $q_n \in F$ and $q_i \neq a$ for all $0 \leq i \leq n$. Since $\delta_1(q, x) = \delta(q, x)$ when $q \notin F$ or $x \neq \varepsilon$, N_1 will have the same computation path p on the sequence $w_1, \dots, w_n$. After processing w_n, N_1 will be in state $q_n \in F$, so $\delta_1(q_n, \varepsilon) = \delta(q_n, \varepsilon) \cup \{a\}$. a is in F_1, so N_1 will accept w.

(ii) **Soundness** ($\mathcal{L}(N_1) \subseteq L$) Suppose N_1 accepts a string $w = w_1 \dots w_n$, $n \geq 1$, $q_n \in \Sigma \cup \{\varepsilon\}$. Let the computation path of N_1 on w be $p = q_0 q_1 \dots q_n$, then $q_n = a$. The only transition into a is an epsilon-transition from a state $q \in F$. Therefore, the sequence $w_1 \dots w_n$ is in L, since the corresponding computation path ends at a state in F.

□

(b) Prove, by giving a concrete counterexample, that the analogous result does not hold for DFAs, i.e., that for a nonempty language L recognized by a DFA M, there might not exist any 1-accept-state DFA that recognizes L. Explain clearly why your chosen language L has this property.

For the alphabet $\Sigma = \{0, 1\}$, let $L = \{x \in \Sigma^* : |x| \in \{2, 3\}\}$ Consider the DFA that recognizes L:

$$
\begin{aligned}
M &= (Q, \Sigma, \delta, q_0, F) \quad \text{where} \\
Q &= \{q_i : 0 \le i \le 4\} \\
\delta(q_i, x) &= q_{i+1} \\
q_0 &= q_0 \\
F &= \{q_2, q_3\}
\end{aligned}
$$

Note that the DFA has two accepting states, q_2 and q_3. However, we have that $\delta(q_2, x) = q_3$ but $\delta(q_3, x) = q_4$, i.e. M transitions from q_2 to an accepting state (q_3), but M transitions from q_3 to a non-accepting state (q_4). Therefore, we cannot merge q_2 and q_3 into a single state. Furthermore, a DFA cannot move to another state without getting any valid input symbol $x \in \Sigma$, so we cannot add epsilon-transitions from q_2 and q_3 into a new accepting state.

We therefore *have to* retain q_2 and q_3 as separate states in order to recognize L.

Problem 5.

Recall the definitions of $\textsc{Max}(L)$ and $\textsc{Min}(L)$ from Homework 1. Prove that, for every regular language L, the languages $\textsc{Max}(L)$ and $\textsc{Min}(L)$ are both regular.

$$\textsc{Min}(L) = \{x \in \Sigma^* : x \in L \text{ and no proper prefix of } x \in L\},$$

$$\textsc{Max}(L) = \{x \in \Sigma^* : x \in L \text{ and } x \text{ is not a proper prefix of any string in } L\}.$$

Since L is regular, there exists a DFA M that recognizes L. Let $M = (Q, \Sigma, \delta, q_0, F)$ be such a DFA.

(a) $\textsc{Min}(L)$

Note that $\textsc{Min}(L) \subseteq L$. Let $s = s_1 \ldots s_n, n \geq 1, s_i \in \Sigma$ be a string in L such that $s \notin \textsc{Min}(L)$, then $\exists\ t = s_1 \ldots s_k, k < n$, such that $t \in L$. Let $p_s = p_0, \ldots, p_n$ be the computation path of M on s such that $p_0 = q_0$ and $p_i = \delta(p_{i-1}, s_i)$ for all i, and $p_n \in F$. Then p_t, the computation path of M on t, is equivalent to $p_0, \ldots, p_k$ for some $k < n$ since the first k letters in s are identical to the first k letters in t and both computation paths start at q_0 yet DFAs may only transition to a single state given an input letter and a state. Therefore, the computation path of M on a string $s \notin \textsc{Min}(L)$ always enters and leaves at least one accepting state before reaching the final state. To prune s, redirect all transitions *out of* accepting states to a new trap-state r that does not accept any strings.

$$M_2 = (Q \cup \{r\}, \Sigma, \delta_2, q_0, F) \text{ where}$$

$$r \notin Q$$

$$\delta_2(q, a) = \begin{cases} r & \text{if } q \in F \cup \{r\} \\ \delta(q, a) & \text{otherwise} \end{cases}$$

(i) $\mathcal{L}(M_2) \supseteq \textsc{Min}(L)$ (Completeness).

Suppose a string s is in $[Min]L$, then s must not have a proper prefix that happens to be in L. Therefore, the computation path of M_2 on s never enters *and leaves* an accepting state, and M_2 does identical transitions to those on M on s since $\delta_2(q, x) = \delta(q, x)$ for all $q \in Q \setminus F$ and $x \in \Sigma$. Since $s \in L$, M accepts s, so M_2 also accepts s.

(ii) $\mathcal{L}(M_2) \subseteq \textsc{Min}(L)$ (Soundness).

Suppose a string s is accepted by M_2, then the computation path of M_2 on s never enters *and leaves* an accepting state, since $\delta_2(q, x) = r$ for all $q \in F$ and $x \in \Sigma$. Therefore, s must not have a proper prefix that happens to be in L.

(b) $\textsc{Max}(L)$

Let M be a DFA that recognizes L.

We define:

$$\delta^* : Q \times \Sigma^* \to Q$$

$$\delta^*(q, x) = \begin{cases} q & \text{if } x = \varepsilon. \\ \delta^*(\delta(q, s), t) & \text{otherwise, where } q = st, s \in \Sigma, t \in \Sigma^*. \end{cases}$$

$$R = \{q \in F : \delta^*(q, s) \in F \text{ for some } s \in \Sigma^* \setminus \{\varepsilon\}\}.$$

Observation 5.1. R is the set of all *accepting states* in M and have some non-empty transition sequence to some accepting state (which could be the same state).

Let M_2 be defined as follows:

$$M_2 = (Q, \Sigma, \delta, q_0, F \setminus R)$$

Claim 5.2. $\mathcal{L}(M_2) = \textsc{Max}(L)$

Proof.

(i) **Completeness** ($\mathcal{L}(M_2) \supseteq \textsc{Max}(L)$) Suppose a string s is in $\textsc{Max}(L)$, then $s \in L$, since $\textsc{Max}(L) \subseteq L$. Let $s = s_1 \dots s_n, n \geq 1, s_i \in \Sigma$, and let $p = p_0, \dots, p_n$ be the computation path of M on s where $p_0 = q_0$ and $p_i = \delta(p_{i-1}, s_i)$ for all i. Then $p_n \in F$ since $s \in L$.

However, s is not a proper substring of any string in L (since $s \in \textsc{Max}(L)$), therefore $p_n \notin R$. This implies that $p_n \in F \setminus R$. Since M_2 uses the same transition function as M, the computation path of M_2 on s is equivalent to $p_0, \dots, p_n$ and p_n is an accepting state in M_2 so M_2 accepts s.

(ii) **Soundness** ($\mathcal{L}(M_2) \subseteq \textsc{Max}(L)$) Let s be a string accepted by M_2. Let $s = s_1 \dots s_n, n \geq 1, s_i \in \Sigma$, and $p = p_0, \dots, p_n$ be the computation path of M_2 on s where $p_0 = q_0$ and $p_i = \delta(p_{i-1}, s_i)$ for all i. Then $p_n \in F \setminus R$ since M_2 accepts s. This implies two things:

- $p_n \in F$, meaning M, the original DFA, accepts s. Therefore, s is a member of the language L.
- $p_n \notin R$, meaning p_n does not have a non-empty transition sequence to some accepting state. Therefore, there does not exist any non-empty string $y \in \Sigma^*$ such that $\delta^*(p_n, y) \in F$. This

implies that there is no string in Σ^8 that, when appended to s, results in a string that is accepted by M (that is, a string in L). This is the same as saying that s is not a proper substring of any string in L, so $s \in \textsc{Max}(L)$.

$\square$

Problem 6.

Recall the definition of $\textsc{Half}(L)$ from Homework 1. Prove that if L is regular, so is $\textsc{Half}(L)$.

$$\textsc{Half}(L) = \{x \in \Sigma^* : \exists y \in \Sigma^* \text{ such that } xy \in L\}$$

High Level Idea

Since L is regular, let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA that recognizes L. We can track whether a string s is a subsequence of another string t in L by tracking the set of states reachable from q that eventually reach an accepting state. We can start this tracking with the start state, q_0, and the set of accepting states, F, Each time we take a step in our DFA, we replace the set S of tracked states by those states that have a transition into *at least* one member of S. Consequently, every time we transition in our DFA, we also backtrack one step. If we reach a state after processing a string $s \in \textsc{Half}(L)$, then the state q_s must be in the set of tracked states.

Formal Definition

Let $\psi : \mathcal{P}(Q) \to \mathcal{P}(Q)$ be a function as follows:

$$\psi(S) = \bigcup_{s \in S} \{q \in Q : \exists y \in \Sigma, \delta(q, y) = s\}.$$

We may also define function exponentiation as repeated function application, i.e.

$$\psi^n(S) = \begin{cases} S & \text{if } n = 0 \\ \psi^{n-1}(\psi(S)) & \text{if } n \geq 0 \end{cases}$$

Define a new DFA M_2 as follows:

$$\begin{aligned} M_2 &= (Q \times \mathcal{P}(Q), \Sigma, \delta_2, q_{0_2}, F_2) \text{ where} \\ q_{0_2} &= (q_0, F) \\ \delta_2\left((q, S), x\right) &= (\delta(q, x), \psi(S)) \\ F_2 &= \{(q, S) \in Q \times \mathcal{P}(Q) : q \in S\}. \end{aligned}$$

Claim 6.1. $\mathcal{L}(M_2) = \textsc{Half}(L)$.

Proof.

(i) **Completeness** ($\mathcal{L}(M_2) \supseteq \textsc{Half}(L)$)

Suppose a string u is a member of $\textsc{Half}(L)$, then there exists a string $t \in L$ and a string $v \in \Sigma^*$ such that $|u| = |v|$ and $t = uv$. Let $t = t_1, \ldots, t_n, \ldots, t_{2n}$ where $t_i \in \Sigma$ for all $0 < i \leq 2n$, and let $p = p_0, \ldots, p_n, \ldots p_{2n}$ be the computational path of M on t such that $p_0 = q_0$ and $p_i = \delta(p_{i-1}, t_i)$. Since $t \in L$, M recognizes t, $p_n \in F$. Furthermore, $t = uv$ and $|u| = |v|$, so we can write $u = t_1, \ldots, t_n$ and $v = t_{n+1}, \ldots, t_{2n}$.

Consider the computational path of M_2 on u. M_2 starts in state (q_0, F) and processes each of the characters t_1 up to t_n in sequence, transitioning from the states (q_0, F) to $(q_1, \psi(F))$, to $(q_2, \psi^2(F))$, all the way to $(q_n, \psi^n(F))$ and so on.

Recall that $t \in L$ and $p = p_1 \ldots p_n, \ldots p_{2n}$ is the computational path of M on t. Therefore, $p_{2n} \in F$. Since $\delta(p_{2n-1}, t_n) = p_{2n}$, we have that $p_{2n-1} \in \psi(F)$. Similarly, $\delta(p_{2n-2}, t_{2n-1}) = p_{2n-1}$, so $p_{2n-2} \in \psi^2(F)$, and $p_{2n-i} \in \psi^i(F)$ for all $0 < i \leq n$. Therefore, $p_n \in \psi^n(F)$, and $(q_n, \psi^n(F))$ is an accepting state of M_2.

(ii) **Soundness** ($\mathcal{L}(M_2) \subseteq \textsc{Half}(L)$)

Suppose a string s is accepted by M_2. Let $s = s_1, \ldots, s_n$ where $s_i \in \Sigma$ for all $0 < i \leq n$. Let $p = p_0, \ldots, p_n$ be the computational path of M_2 on s, then $p_n \in \psi^n(F)$ (by definition of the accepting states of M_2). That implies that there exists a state $p_{n+1} \in Q$ such that $\delta(p_n, x) = p_{n+1}$ for some $x \in \Sigma$. Therefore, $p_{n+1} \in \psi^{n-1}(F)$. Similarly, there must exist a state $p_{n+2} \in Q$ such that $\delta(p_{n+1}, x_1) = p_{n+2}$ for some $x_1 \in \Sigma$. This implies again that $p_{n+2} \in \psi^{n-2}(F)$, so there must exist a state $p_{n+3} \in Q$, such that $\delta(q_{n+2}, x_3) = p_{n+3}$ for some $x_3 \in \Sigma$. More generally, for each $0 < i \leq n$, there must exist a state $p_{n+i} \in Q$ such that $\delta(p_{n+i-1}, x) = p_{n+i}$ for some $x \in \Sigma$. Since it took n transitions to reach the accepting state $(p_n, \psi^n(F))$, there must be n states $p_{n+1}, \ldots, p_{2n}$ such that $\delta(p_{n+i-1}, x) = p_{n+i}$ for all $0 < i \leq n$, and $p_{2n} \in F$, meaning the corresponding string $s' = s_1 \ldots s_{2n}$ is a member of L. Therefore, $s \in \mathcal{L}(M_2)$.

$\square$