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

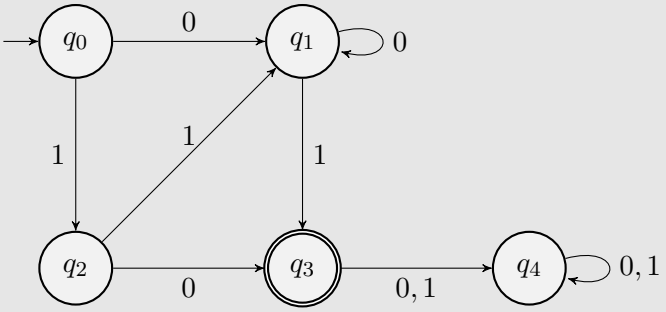

FIGURE 1.1. NFA to accept $10 \cup (0 \cup 11)0^*1$

(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 $n \pmod 3 \equiv 0$.

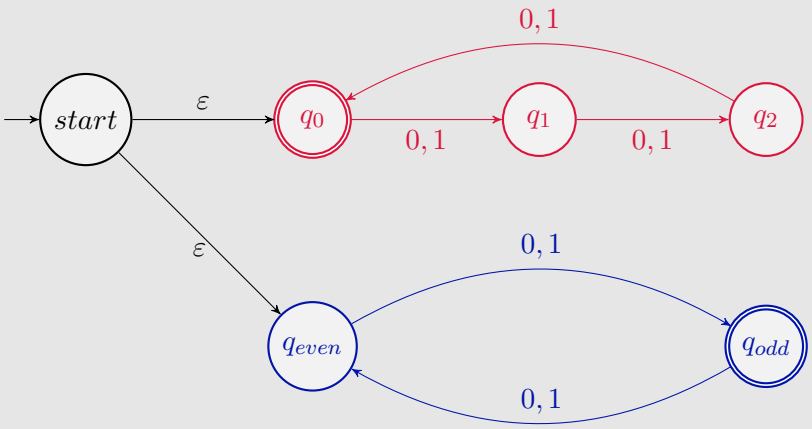

Figure 1.2. NFA to accept $(0 \cup 1)((0 \cup 1)(0 \cup 1))^* \cup ((0 \cup 1)(0 \cup 1)(0 \cup 1))^*$

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.

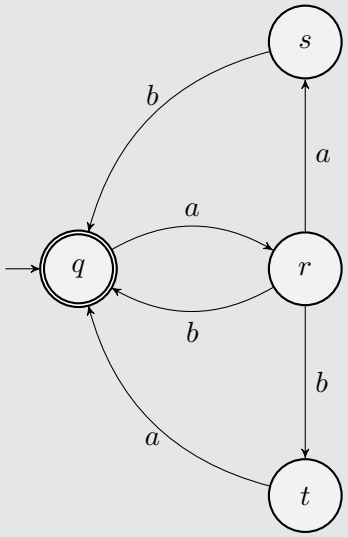

Figure 3.1. NFA to recognize $L \equiv (ab \cup aab \cup aba)^*$

(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.*

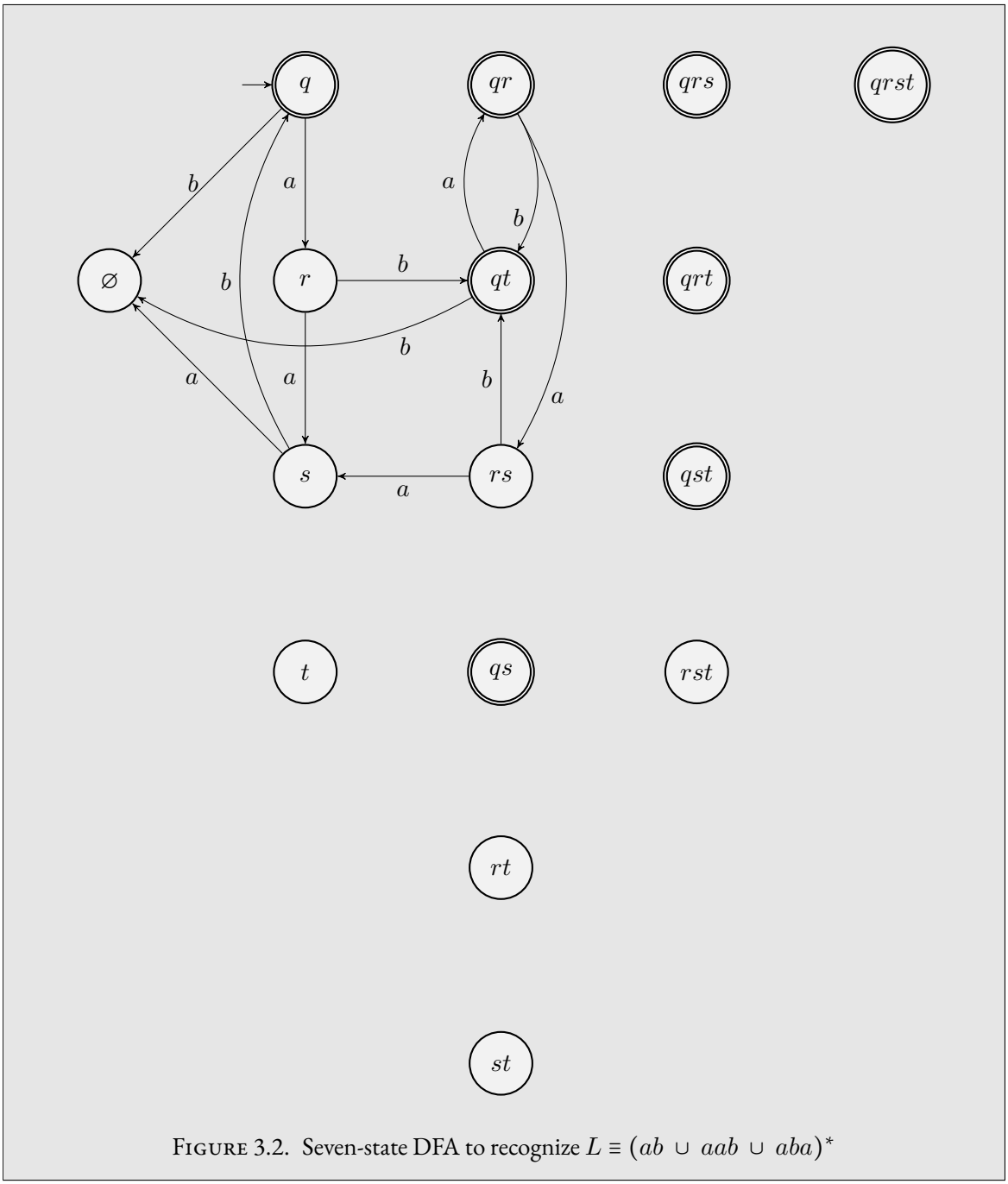

Figure 3.2. Seven-state DFA to recognize $L \equiv (ab \cup aab \cup aba)^*$

(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:

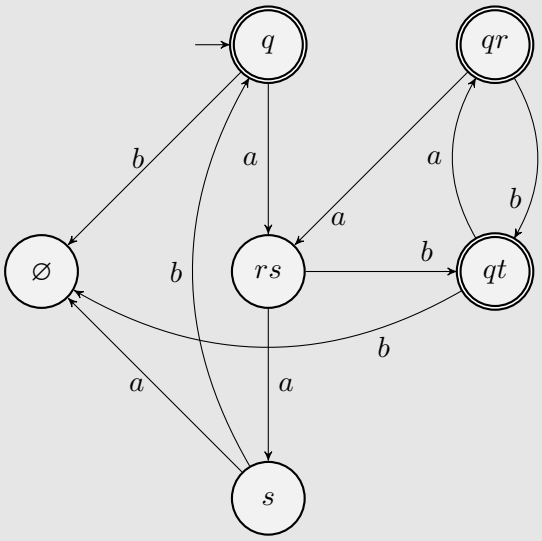

Figure 3.3. Six-state DFA to recognize $L \equiv (ab \cup aab \cup aba)^*$

(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.

Any DFA that recognizes L must differentiate between 6 unique strings, each of which may or may not belong to L but has a unique set of δ-transitions. These strings are:

(i) aa and ab: The string aa does not belong to L, but the string ab does. Any DFA that recognizes L must therefore treat these two strings differently.

(ii) aab and aba: While both aab and aba belong in L, note that upon reading a aab followed by b, any DFA that recognizes L must transition to a reject state (since $aabb \notin L$) while the same DFA must transition to an accept state upon reading aba followed by b (since $abab \in L$). Therefore, the DFA may not transition to the same state upon reading aab vs. aba, since its transitions from that state must be distinct from each other.

(iii) $aaab$ and $abab$: The string $aaab$ does not belong to L, but the string $abab$ does. Any DFA that recognizes L must therefore treat these two strings differently.

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 $N_1 = (Q \cup F_2, \Sigma, \delta_1, q_0, F_2)$, where a is a new state that is not in Q, and:

$$F_2 = \{a\}$$

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

Lemma 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) $\mathcal{L}(N_1) \supseteq L$ (completeness) Let $w = w_1 w_2 \dots w_k : w_i \in \Sigma \text{ for all } 0 < i \leq k$ 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 $q_n \in F$. Note that $q_i \neq a$ for all $0 < i \leq n$, since $a \notin Q$. Consider the computation path of N_1 on w,

(ii) $\mathcal{L}(N_1) \subseteq L$ (soundness)

□

(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.

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 $s = s_1 \dots s_n, n \geq 1, s_i \in \Sigma$ be a string in L such that $s \notin \textsc{Min}(L)$, then $\exists\ t = s_1 \dots s_k, k < n$, such that $t \in L$. Let $p_s = p_0, \dots, p_n$ 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 $p_0, \dots, p_k$ 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 $\lceil Min \rceil L$, 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) $\textsc{Max}(L)$

Note that $\textsc{Max}(L) \subseteq L$. Let $s = s_1 \dots s_k, k \geq 1, s_i \in \Sigma$ be a string in L such that $s \notin \textsc{Max}(L)$, then $\exists\ t = s_1 \dots s_k s_{k+1} \dots s_n, k < n$, such that $t \in L$. We wish to prune all such s where the set $T = t \in \Sigma^* : st \in L$ is non-empty.

Problem 6.

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