**Spring 2020 CIS 262**

# Automata, Computability and Complexity
# Jean Gallier

# Solutions for the First Review Session

February 24, 2020
Solutions

**Problem 1.** (1) An NFA with a single $\epsilon$-transition accepting $L = \{aa, bb\}^*$ whose transition table

| | $\epsilon$ | $a$ | $b$ |
|---|---|---|---|
| 0 | $\emptyset$ | 1 | 2 |
| 1 | $\emptyset$ | 3 | $\emptyset$ |
| 2 | $\emptyset$ | $\emptyset$ | 3 |
| 3 | 0 | $\emptyset$ | $\emptyset$ |

is shown below:

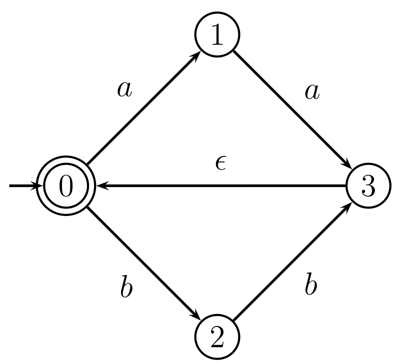


Figure 1: NFA for $L = \{aa, bb\}^*$

(2) Convert the NFA of question (a) to a DFA.

When we apply the subset construction, we get:

| | | $a$ | $b$ |
|---|---|---|---|
| $A$ | $\{0\}$ | $B$ | $C$ |
| $B$ | $\{1\}$ | $D$ | $E$ |
| $C$ | $\{2\}$ | $E$ | $D$ |
| $D$ | $\{0,3\}$ | $B$ | $C$ |
| $E$ | $\emptyset$ | $E$ | $E$ |

The final states are $A$ and $D$ and the start state is $A$.

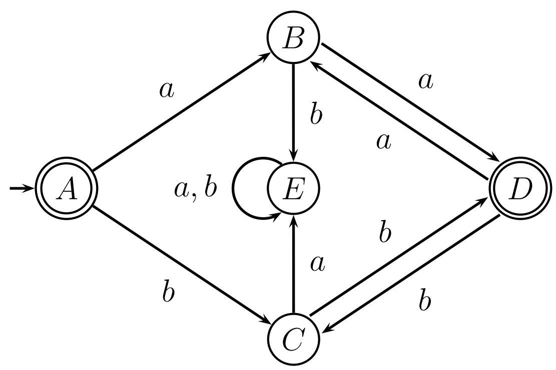


Figure 2: DFA for $L = \{aa, bb\}^*$

**Problem 2.** (1) Given any DFA, $D = (Q, \Sigma, \delta, q_0, F)$, let $D'$ be the DFA, $D' = (Q \cup \{q_0'\}, \Sigma, \delta', q_0', F')$, where $q_0'$ is a new state not in $Q$, with $F' = F$ if $q_0 \notin F$ else $F' = F \cup \{q_0'\}$, and with the transition function $\delta'$ defined as follows:

For all $a \in \Sigma$, if $p \in Q$ then

$$\delta'(p, a) = \delta(p, a)$$

else

$$\delta'(q_0', a) = \delta(q_0, a).$$

Clearly, there are no incoming transitions into $q_0'$ and since the transitions from $q_0'$ are identical to the transitions from $q_0$ and all the other transitions are the same as in $D$, we have $L(D') = L(D)$.

(2) It is false that a DFA accepts a finite language iff its contains no underlying cycle. This is because, given any DFA, there must be a transition from every state on every input and as a DFA is finite, *every DFA has a cycle*! For example, the following DFA over the alphabet $\{a\}$ only accepts $\epsilon$, yet it has a cycle:

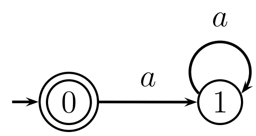


Figure 3: DFA for $\{\epsilon\}$

**Problem 3.** By definition, $L^R = \{w^R \mid w \in L\}$. Recall that it was proved that

$$(uv)^R = v^R u^R \quad \text{and} \quad (w^R)^R = w,$$

for all $u, v, w \in \Sigma^*$. We have

$$\begin{aligned} w \in (L_1L_2)^R \quad & \text{iff} \quad w^R \in L_1L_2 \\ & \text{iff} \quad (\exists u \in L_1)(\exists v \in L_2)(w^R = uv) \\ & \text{iff} \quad (\exists u \in L_1)(\exists v \in L_2)(w = v^R u^R) \\ & \text{iff} \quad (\exists x \in L_1^R)(\exists y \in L_2^R)(w = yx) \\ & \text{iff} \quad w \in L_2^R L_1^R, \end{aligned}$$

which proves that

$$(L_1L_2)^R = L_2^R L_1^R.$$

We claim that

$$(L^n)^R = (L^R)^n, \qquad \text{for all } n \geq 0.$$

This is proved by induction. For $n = 0$, we have

$$(L^0)^R = \{\epsilon\}^R = \{\epsilon\} = (L^R)^0,$$

so the base case holds.

Assume the induction hypothesis holds for any $n \geq 0$. Using $(L_1L_2)^R = L_2^R L_1^R$, we get

$$(L^{n+1})^R = (L^n L)^R = L^R (L^n)^R = L^R (L^R)^n = (L^R)^{n+1},$$

establishing the induction step.

Then, we get

$$(L^*)^R = \left( \bigcup_{n \geq 0} L^n \right)^R = \bigcup_{n \geq 0} (L^n)^R = \bigcup_{n \geq 0} (L^R)^n = (L^R)^*,$$

so

$$(L^*)^R = (L^R)^*,$$

as claimed.

**Problem 4.** Let $\Sigma = \{a, b\}$.

(1) A DFA accepting

$$L_1 = \{w \in \Sigma^* \mid w \text{ contains an even number of } a\text{'s}\}.$$

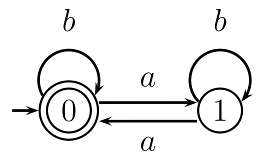


Figure 4: DFA for $L_1$

(2) A DFA accepting

$$L_2 = \{w \in \Sigma^* \mid w \text{ contains a number of } b\text{'s divisible by } 3\}.$$

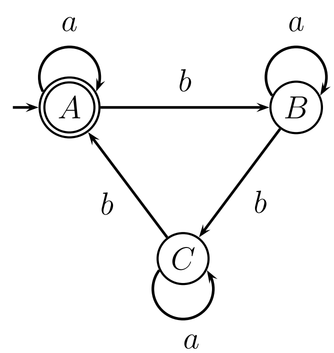


Figure 5: DFA for $L_2$

(3) A DFA accepting $L_3 = L_1 \cap L_2$.

The cross-product construction (for intersection) yields:

| | $a$ | $b$ |
|---|---|---|
| $(0, A)$ | $(1, A)$ | $(0, B)$ |
| $(0, B)$ | $(1, B)$ | $(0, C)$ |
| $(0, C)$ | $(1, C)$ | $(0, A)$ |
| $(1, A)$ | $(0, A)$ | $(1, B)$ |
| $(1, B)$ | $(0, B)$ | $(1, C)$ |
| $(1, C)$ | $(0, C)$ | $(1, A)$ |

The start state $(0, A)$ is also the only final state.

**Problem 5.**

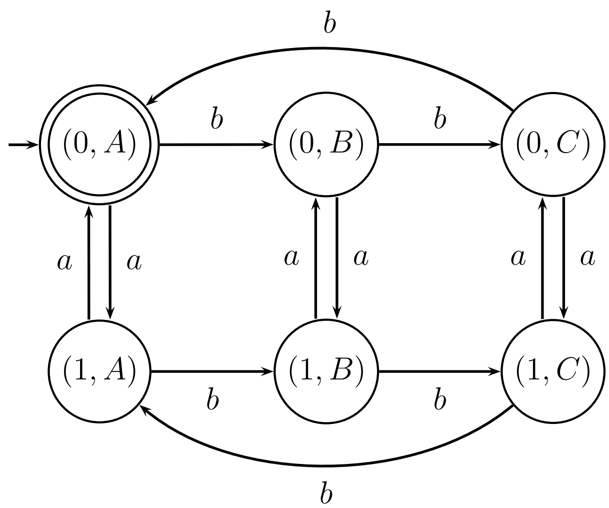


Figure 6: DFA for $L_1 \cap L_2$

Let $\Sigma = \{a, b\}$. Describe a method taking as input any DFA $D$ (over $\{a, b\}$) and testing whether

$$L(D) = \{a\}^* b\{a, b\}^*.$$

The regular language, $\{a\}^* b\{a, b\}^*$ is accepted by a two-state DFA, $D'$, with $Q = \{0, 1\}$, start state 0 and final state, 1, and with

$$\begin{aligned} \delta(0, a) &= 0 \\ \delta(0, b) &= 1 \\ \delta(1, a) &= 1 \\ \delta(1, b) &= 1. \end{aligned}$$

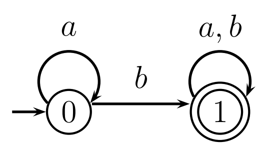


Figure 7: DFA for $\{a\}^* b\{a, b\}^*$

As $L(D) = \{a\}^* b\{a, b\}^* = L(D')$ iff $L(D) \subseteq L(D')$ and $L(D') \subseteq L(D)$, from the hint,

$$L(D) = \{a\}^* b\{a, b\}^* = L(D') \quad \text{iff} \quad L(D) - L(D') = \emptyset \quad \text{and} \quad L(D') - L(D) = \emptyset.$$

We know that the cross-product constructions for relative complements yields DFA's, $D_1$ and $D_2$, so that $L(D_1) = L(D) - L(D')$ and $L(D_2) = L(D') - L(D)$. Thus, we can test whether $L(D) = \emptyset$ by testing whether $L(D_1) = \emptyset$ and $L(D_2) = \emptyset$. However, this holds iff no final state of $D_1$ is reachable and no final state of $D_2$ is reachable, which can be tested by computing the reachable states of $D_1$ and $D_2$ using the algorithm described in the notes. The set of final states of $D_1$ is $F \times \overline{F'}$ and the set of final states of $D_2$ is $\overline{F} \times F'$. So no state in $F \times \overline{F'}$ should be reachable in $D_1$ and no state in $\overline{F} \times F'$ should be reachable in $D_2$.

**Problem 6.**

Let $D = (Q, \Sigma, \delta, q_0, F)$ be a DFA and assume that $Q$ contains $n \geq 1$ states. Prove that if there is some string $w \in \Sigma^*$ such that $w \in L(D)$ and $|w| \geq n$, then there is some string $u \in \Sigma^*$ such that $u \in L(D)$ and $|u| < n$.

*Claim*. The sequence $q_0, q_1, \ldots, q_m$ of states in the computation from $q_0$ on input $w$ (with $m = |w|$) with $q_m \in F$ must contain two identical states $q_h = q_k$, for $0 \leq h < k \leq n$.

Since $|w| \geq n$, we have $m \geq n$. The sequence

$$q_0, q_1, \ldots, q_n$$

has $n + 1$ elements, but $Q$ contains $n$ distinct states, so by the pigeonhole principle, two of the sates in the sequence must be identical, say $q_h = q_k$, for $0 \leq h < k \leq n$.

Consider a string $u \in \Sigma^*$ of *minimal length* such that $u \in L(D)$.

Assume by contradiction that $|u| \geq n$. Then, by the claim, the sequence $q_0, q_1, \ldots, q_m$ of states in the computation from $q_0$ on input $w$ (with $m = |w|$) with $q_m \in F$ must contain two identical states $q_h = q_k$, for $0 \leq h < k \leq n$. Thus we can write $u = xyz$, where $x$ is the string that takes us from $q_0$ to $q_h$, $y$ is the string that takes us from $q_h$ to $q_k = q_h$, and $z$ is the string that takes us from $q_q$ to $q_m$, and by construction, $0 < |y| \leq n$. Then by skipping the sequence of states from $q_h$ back to $q_k = q_h$, we obtain the sequence

$$q_0, q_1, \ldots, q_h, q_{k+1}, \ldots, q_m$$

with $q_m \in F$, showing that $xz \in L(D)$. But since $0 < |y|$, we have

$$|xz| < |xyz| = |w|$$

with $xz \in L(D)$, contradicting the minimality of $u$. Therefore, $|u| < n$, as claimed.