HW 2 Solution

CS/ECE 374 B: Algorithms & Models of Computation, Spring 2020

Submitted by:

- ≪**Michael Jiang**≫: ≪**minhaoj2**≫
- ≪**Yumai Sun**≫: ≪**yumais2**≫
- ≪**Shuchen Zhang**≫: ≪**szhan114**≫

4 Problem 4:

Solution 4.a:

The strings that do not contain 011 as a substring: 1*0*(100*)*(1 + ϵ).

Since the string cannot contain 011 as a substring, for all substring 01, there has to be a 0 after that. (100*) guarantees that every occurrence of 01 would have a 0 followed by that. Therefore, this regex is correct.

Solution 4.b:

The strings that do not contain 011 as a subsequence: 1*+1*0*10*.

Since we do not want the subsequence of 011, there could only be one 1 after any number of 0s in the string. The other situation is a string with only 1s. Combining the two situations we get this regular expression.

Solution 4.c:

The strings that start with 00 and have 001 as a substring: 000*1(0+1)*

Because the string starts with 00, we put 00 at the beginning, then it could be followed by any number of 0s, but should be followed by 1 immediately to let 001 be a substring, then the rest of the string could be arbitrary.

Solution 4.d:

The strings that contain either the substring 10 or the substring 01, but not both: $1^+0^+ + 0^+1^+$

To satisfy the condition, when the string contain substring 10, then it cannot contain substring 01. Thus, there can't be any 0s before the substring 10 and there can't be any 1s after the substring 10. Similarly, when the string contain substring 01, then it cannot contain subtring 10. Thus, there can't be any 1s before the substring 01 and there can't be any 0s after the substring 01.

Solution 4.e:

The strings that in which every nonempty maximal substring of consecutive 0s is of even length: $(1 + 00)^*$

To satisfy the condition, every run of 0s must be of even length.

5 Problem 5:

Solution 5.A:

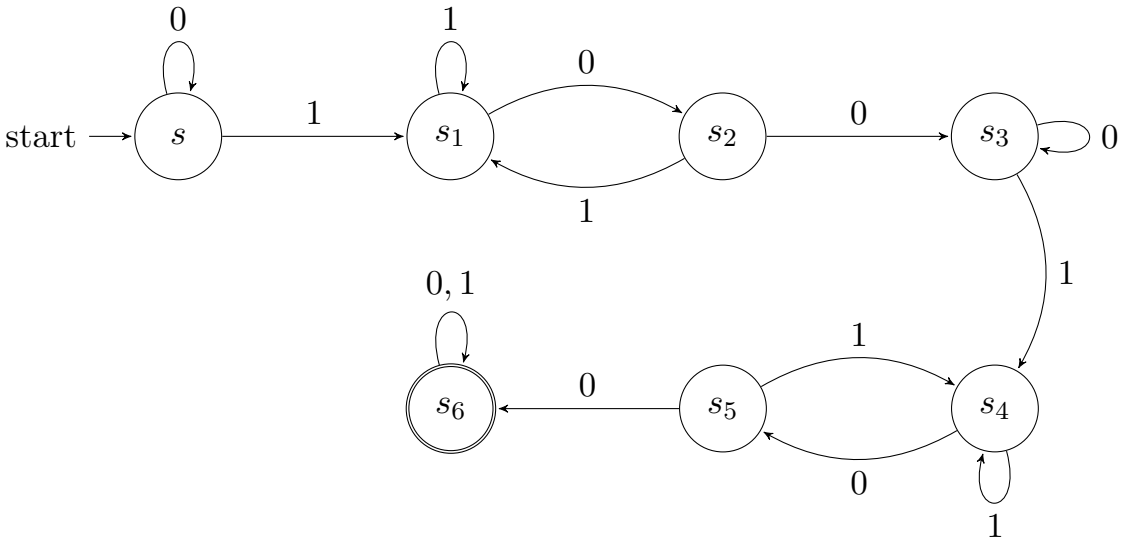

Figure 1: DFA graph

The DFA above is the solution to this problem, which contain all the strings that have 100 as a substring for at least twice. In this case, the automata can be defined as below, M = $\{\mathbb{Q}, \sum, \delta, s, \mathbf{A}\}$. As defined in the context, the input alphabet $\sum = \{0, 1\}$. The state set $\mathbb{Q}$ is $\{s, s_1, s_2, s_3, s_4, s_5, s_6\}$, where s is the start state **S** and s_6 is the acceptance state **A**. And the transition function δ is indicated as in the graph.

Solution 5.B:

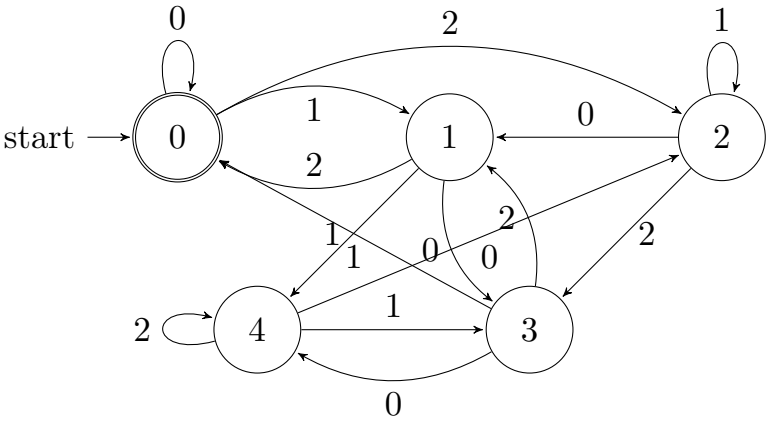

Figure 2: DFA graph

The DFA above is the solution to this problem. $\sum = \{0, 1, 2\}$ and the states $\mathbb{Q} = \{0, 1, 2, 3, 4\}$ represents the remainder of the input ternary numbers when divided by 5. The others are stated in the graph.

6 Problem 6:

Solution 6.A:

Lemma #1: $\delta^*(q, xa) = \delta(\delta^*(q, x), a)$ (from proof in the lecture notes)
Let L' represent the set of all prefixes of all the strings in L(M), that is, $L' = \{x | x$ is prefix of any string $\omega = x \bullet y$ that $\forall \omega \in L(M), x, y \in \sum^*\}$. And our claim is, $L(M') = \sum^* \setminus L'$. In other words, $L(M')$ is the complement of the set of all prefixes of any string in L(M).
And we will prove this claim with two cases: $L(M') \subseteq (\sum^* \setminus L')$ and $(\sum^* \setminus L') \subseteq L(M')$.

First case: $L(M') \subseteq (\sum^* \setminus L')$.

Suppose a string $\omega \in L(M')$. By the definition of DFA M', we can get that $\delta^*(s, \omega) = B(M)$ where s is the starting state of M'. Since for all upcoming inputs for bad states would result in bad state, for strings ω which can reach bad state cannot be the prefix of any string in L(M). In this case, $\omega \notin L'$. Therefore, $\omega \in$ the complement of L'. Therefore, $\omega \in \sum^* \setminus L'$. Thus, $L(M') \subseteq (\sum^* \setminus L')$.

Second case: $(\sum^* \setminus L') \subseteq L(M')$.

We will prove this by contradiction.
Suppose a string $\omega \in \sum^* \setminus L'$. By the definition of set difference, $\omega \notin L'$. And we also assume that $\omega \notin L(M')$. Therefore, $\delta^*(s, \omega) \neq B(M)$. By the definition of strings that are not in bad state, we know that for some string $\lambda \in \sum^*, \delta^*(\delta^*(s, \omega), \lambda) \in A$. And we will prove by induction on $|\lambda|$ that $\delta^*(\delta^*(s, \omega), \lambda) = \delta^*(s, \omega \bullet \lambda)$.

Base Case: When $\lambda = \epsilon$, that is, no input for $\delta^*(s, \omega)$. Therefore, the equation holds naturally.

Inductive Hypothesis: For some $k \in \mathbb{N}$, that k ≥ 1, for all strings a with length $\leq$ k satisfy that $\delta^*(\delta^*(s, \omega), a) = \delta^*(s, \omega \bullet a)$.

Inductive step: For some string λ with length k + 1, we can represent it as $k = a \bullet b$ where b is a symbol. Therefore,
$\delta^*(\delta^*(s, \omega), \lambda)$
$= \delta^*(\delta^*(s, \omega), a \bullet b)$
$= \delta^*(\delta^*(s, \omega \bullet a), b)$ (by inductive hypothesis)
$= \delta^*(s, \omega \bullet \lambda)$ (by lemma 1).

Therefore, we proved that $\delta^*(\delta^*(s, \omega), \lambda) = \delta^*(s, \omega \bullet \lambda)$. Therefore, for some string $\omega \bullet \lambda$, it can reach an accepting state in M. Therefore, ω is a prefix of a string in L(M), which is a contradiction with the assumption that $\omega \in \sum^* \setminus L'$. In this case, ω must be in L(M'), and $(\sum^* \setminus L') \subseteq L(M')$.

Conclusion: With the two cases above, we have proved that, $L(M') = \sum^* \setminus L'$. In other words, we proved that $L(M')$ is the complement of the set of all prefixes of any string in L(M).

Solution 6.B:

From proof of the second case in 6.A, we can see that $\delta^*(s, xy) = \delta^*(\delta^*(s, x), y)$.
Let $x \in L(M'), y \in \Sigma^*$ be given.
Then we have $\delta^*(s, xy) = \delta^*(\delta^*(s, x), y)$, by the proof in 6.A
Since $x \in L(M')$, $\delta^*(s, x) = q$, where $q \in B(M)$ by the definition of M'.

Then $\delta^*(q, y) \in B(M)$, by the definition of bad state.
Therefore, we can conclude that if $x \in L(M'), y \in \Sigma^*$, then $xy \in L(M')$.

Solution 6.C:

Let's define a DFA $M = \{Q, \sum, \delta, s, A\}$ where
$Q = Q_2 \times Q_1$
s = (s_2, s_1)
$A = A_2 \times$ $(Q_1 - A_1)$
$\delta = Q \times \sum \to Q$: $\delta((q_2, q_1), a) = \begin{cases} (\delta_2(q_2), \delta_1(q_1)) & \text{if } q_1 \notin A_1 \\ (\delta_2(q_2), q_1) & \text{if } q_1 \in A_1 \end{cases}$

Since we don't want any string with prefix in L_1 as in our accepting state, that is, $\delta^*(q, \omega)$ will not reach any state $a_1 \in A_1$. Therefore, we can modify the transition function so that once $\delta^*(q, \omega) \in A_1$, any input after ω should remain in L_1. Also, since we want the string $\in L_2$, we need to maintain the properties of δ_2. In this case, after removing A_1 from our accepting state, all accepting strings in this new automata would not have prefix in L_1. And the others can remain the same with the product construction of two DFAs.