CS/ECE 374 A (Spring 2022)
Midterm 1 Solutions

---

1. (a) False. A counterexample: 11010 is accepted by the DFA but is not generated by $0^*(11)^*10(0+1)^*$.

    [Note: a correct regular expression for this DFA would be $(0+11)^*10(0+1)^*$.]

   (b) True. By Kleene's theorem, every regular language is recognized by some DFA.

    [Alternative explanation: in class, we have shown how to convert regular expressions to NFAs (by a recursive algorithm), and from NFAs to DFAs (by the subset or power-set construction).]

   (c) True. By the subset of power-set construction, L is accepted by a DFA with at most 2^n states. And the complement of L is accepted by a DFA with the same number of states (by switching the role of accepting and rejecting states).

   (d) False. A counterexample is $L_1 = \{1\}$ and $L_2 = \{0\}$. Here, 11 is in $(L_1 \cup L_2)^*$ but not in $(L_1 L_2)^*$.

   (e) True. A regular expression is $(0+1)^* \cdot \bigcup_{n=0}^{5}\{0^n 1^n 0^n\} \cdot (0+1)^*$. In fact, because $n = 0$ is not forbidden, the language is just $(0+1)^*$!

   (f) False. One counterexample is $L = \{0^{n^2} : n \geq 0\}$, with $\Sigma = \{0\}$. Here, L is not regular (as shown in the labs), but $\{xyz : x \in \{0\}^*,\ y \in L,\ z \in \{0\}^*\}$ is exactly 0^* and so is regular.

    [Another counterexample is $L = \{ww^R : w \in \{0,1\}^*, |w| \geq 374\}$ (with $\Sigma = \{0,1\}^*$). Here, L is not regular, but in a homework problem (Problem 4.1(c)), you have already shown that $\{xww^R z : x, w, z \in \{0,1\}^*, |w| \geq 374\}$ is regular!]

   (g) False. The minimum fooling set size is equal to the minimum number of states over all DFAs accepting the language, but this language has a DFA with 2022 states.

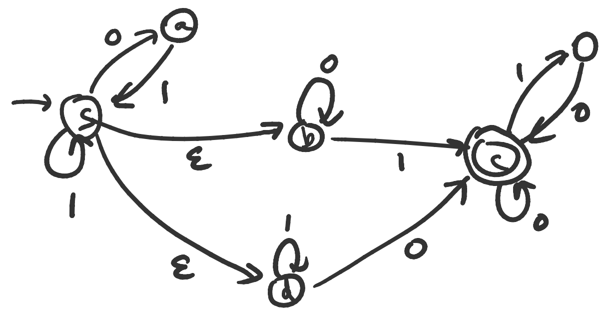
    [Alternatively, one can argue directly: if there is a fooling set F of size 2023, there must exist two distinct strings $x, y \in F$ with $x \equiv y \bmod 2022$ by the pigeonhole principle. But x and y are indistinguishable, since for any z, $|xz|$ is divisible by 2022 iff $|yz|$ is divisible by 2022.]

   (h) True. This is stated in class (we can convert any regular expression to a CFG directly, or alternatively, any DFA to a CFG).

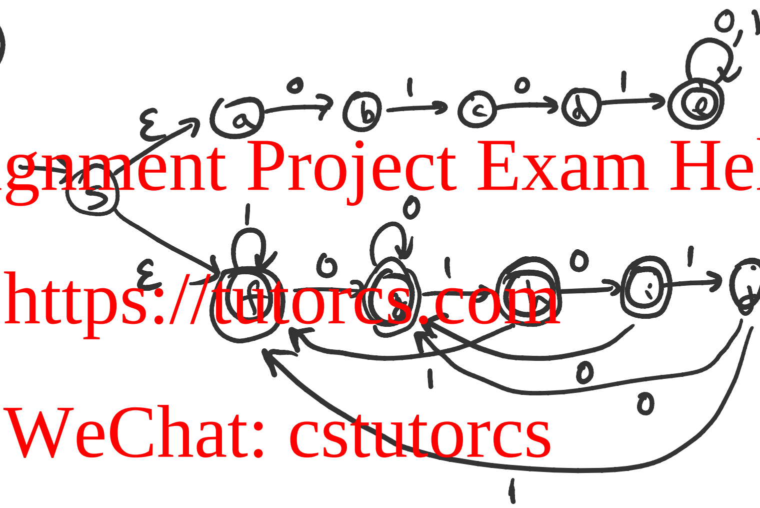
   (i) True. The grammar generates 0^*1^+, which is clearly regular.

February 21, 2022 5:29 PM

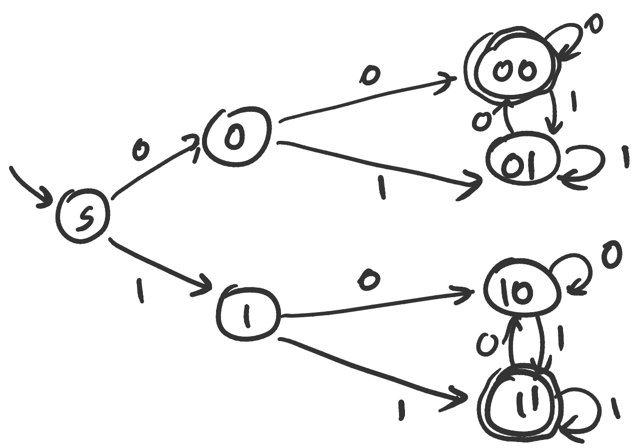
2. a)



b)



c)



2. (c) (Cont'd) Meaning of states:

- s: the start state.
- 0: read one 0.
- 1: read one 1.
- XY: first symbol is X, and last symbol read is Y.

[One alternative solution is to first draw an NFA (which requires just 4 states) and then apply the subset or power-set construction.]

3. (a) $(0^5)^*(1^5)^* + 0(0^5)^*1111(0^5)^* + 00(0^5)^*111(0^5)^* + 000(0^5)^*11(0^5)^* + 0000(0^5)^*1(0^5)^*$.

(b) Define the following DFA $M = (Q, \{0,1\}, s, \delta, A)$:

$$
\begin{aligned}
Q &= \{i : 0 \le i \le 2022\} \cup \{(i,k) : 1 \le i \le 2022, 0 \le k < i\} \cup \{\text{ERR}\} \\
s &= 0 \\
A &= \{(i,k) \in Q : k \ne 0\} \\
\delta(i,0) &= i+1 \qquad \text{if } i \in \{0,\ldots,2021\} \\
\delta(i,1) &= (i,1) \qquad \text{if } i \in \{1,\ldots,2022\} \\
\delta((i,k),1) &= (i,(k+1) \bmod i) \qquad \text{if } i \in \{1,\ldots,2022\} \\
\delta((i,k),0) &= \text{ERR} \\
\delta(0,1) &= \text{ERR} \\
\delta(\text{ERR},0) &= \text{ERR} \\
\delta(\text{ERR},1) &= \text{ERR}
\end{aligned}
$$

Meaning of states:

- ERR is the error state.
- State i means that we have read i 0's (and no 1's).
- State (i,k) means that we have read 0^i1^j for some $j \equiv k \bmod i$.

4. (a) Choose $F = \{10^i : i \ge 0\}$.
Let x and y be two arbitrary distinct strings in F.
Then $x = 10^i$ and $y = 10^j$ for some $i \ne j$.
Choose $z = 10^i1$.
Then $xz = 10^i10^i1 \in L$.
On the other hand, $yz = 10^i10^j1 \notin L$, because $i \ne j$ (so the middle symbol is not 1 but is 0).
Thus, F is a fooling set.
Since F is infinite, L cannot be regular.

[Alternate Proof: Choose $F = \{0^i : i \ge 1\}$.
Let x and y be two arbitrary distinct strings in F.
Then $x = 0^i$ and $y = 0^j$ for some $i, j \ge 1$ with $i \ne j$.
Choose $z = 10^j$.

Then $xz = 0^i 1 0^j \in L$, since the first, middle, and last symbols are all 0's if $i \neq j$.
And $yz = 0^j 1 0^j \notin L$, since the first/last symbol is 0 but the middle symbol is 1.
Thus, F is a fooling set.
Since F is infinite, L cannot be regular.]

(b)

$$
\begin{aligned}
S &\rightarrow 0A0 \mid 1B1 \\
A &\rightarrow 0A0 \mid 0A1 \mid 1A0 \mid 1A1 \mid 0 \\
B &\rightarrow 0B0 \mid 0B1 \mid 1B0 \mid 1B1 \mid 1
\end{aligned}
$$

Meaning of non-terminals:

- A generates all odd-length strings whose middle symbol is 0.
- B generates all odd-length strings whose middle symbol is 1.
- S generates all strings in the given language (since $0A0$ covers the case where left, middle, and right symbols are 0, and $1B1$ covers the case where left, middle, and right symbols are 1).

5. (a) Since $L = \{\varepsilon, 01, 0101, 010101, 01010101, 0101010101, \ldots\}$,
DELETE-FIFTH$(L) = \{00101, 0100101, 010100101, \ldots\}$, which can be described by the regular expression (01)*00101.

(b) Let L be a regular language over $\Sigma = \{0, 1\}$. By Kleene's theorem, L is accepted by some DFA $M = (Q, \Sigma, s, \delta, A)$. We construct an NFA $M' = (Q', \Sigma, s', \delta', A')$ accepting DELETE-FIFTH(L) (which would imply that DELETE-FIFTH(L) is regular by Kleene's theorem). The construction is as follows:

$$
\begin{aligned}
Q' &= \{(q, \textsc{before}) : q \in Q\} \cup \\
&\quad\ \{(q, i, \textsc{after}) : q \in Q,\ i \in \{0, 1, 2, 3, 4\}\} \\
s' &= (s, \textsc{before}) \\
A' &= \{(q, 4, \textsc{after}) : q \in A\}
\end{aligned}
$$

$$
\begin{aligned}
\delta'((q, \textsc{before}), c) &= \{(\delta(q, c), \textsc{before})\} && \forall q \in Q,\ c \in \Sigma \\
\delta'((q, \textsc{before}), \varepsilon) &= \{(\delta(q, a), 0, \textsc{after}) : a \in \Sigma\} && \forall q \in Q \\
\delta'((q, i, \textsc{after}), c) &= \{(\delta(q, c), i + 1, \textsc{after})\} && \forall q \in Q,\ c \in \Sigma,\ i \in \{0, 1, 2, 3\}
\end{aligned}
$$

(All unspecified values of $\delta'(\cdot, \cdot)$ are $\emptyset$.)

Explanation: The idea is to divide the process into two phases: BEFORE (reading the prefix x) and AFTER (reading the suffix y). We use nondeterminism to guess when to switch from the BEFORE phase to the AFTER phase, via an ε-transition, and we also use nondeterminism to guess the symbol a. At the same time, we simulate M on the string xay.