# HW 2 : Extra problems

**CS/ECE 374: Algorithms & Models of Computation, Fall 2017** Version: **1.0**

The following problems are not for submission or grading. No solutions for them will be provided but you can discuss them on Piazza (however, some of them already contain a solution).

**1.** Suppose $N_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $N_2 = (Q_2, \Sigma, \delta_2, s_2, A_2)$ are NFAs. Formally describe a DFA that accepts the language $L(N_1) \setminus L(N_2)$. This combines subset construction and product construction to give you practice with formalism.

**2.** Suppose $M = (Q, \Sigma, \delta, s, A)$ is a DFA. For states $p, q \in Q$ ($p$ can be same as $q$) argue that $L_{p,q} = \{w \mid \delta^*(p, w) = q\}$ is regular. Recall that $\text{PREFIX}(L) = \{w \mid wx \in L, x \in \Sigma^*\}$ is the set of all prefixes of strings in $L$. Express $\text{PREFIX}(L(M))$ as $\cup_{q \in Z} L_{s,q}$ for a suitable set of states $Z \subseteq Q$. Why does this prove that $\text{PREFIX}(L(M))$ is regular whenever $L$ is regular?

**3.** For a language $L$ let $\text{MID}(L) = \{w \mid xwy \in L, x, y \in \Sigma^*\}$. Prove that $\text{MID}(L)$ is regular if $L$ is regular.

**4.** (a) Draw an NFA that accepts the language $\{w \mid$ there is exactly one block of 0s of even length$\}$. (A "block of 0s" is a maximal substring of 0s.)

(b) i. Draw an NFA for the regular expression $(010)^* + (01)^* + 0^*$.

ii. Now using the powerset construction (also called the subset construction), design a DFA for the same language. Label the states of your DFA with names that are sets of states of your NFA.

**5.** This problem is to illustrate proofs of (the many) closure properties of regular languages.

(a) For a language $L$ let $\text{FUNKY}(L) = \{w \mid w \in L$ but no proper prefix of $w$ is in $L\}$. Prove that if $L$ is regular then $\text{FUNKY}(L)$ is also regular using the following technique. Let $M = (Q, \Sigma, \delta, s, A)$ be a DFA accepting $L$. Describe a NFA $N$ in terms of $M$ that accepts $\text{FUNKY}(L)$. Explain the construction of your NFA.

(b) In Lab 3 we saw that $\mathit{insert1}(L)$ is regular whenever $L$ is regular. Here we consider a different proof technique. Let $r$ be a regular expression. We would like to show that there is another regular expression $r'$ such that $L(r') = \mathit{insert1}(L(r))$.

i. For each of the base cases of regular expressions $\emptyset, \epsilon$ and $\{a\}, a \in \Sigma$ describe a regular expression for $\mathit{insert1}(L(r))$.

ii. Suppose $r_1$ and $r_2$ are regular expressions, and $r'_1$ and $r'_2$ are regular expressions for the languages $\mathit{insert1}(L(r_1))$ and $\mathit{insert1}(L(r_2))$ respectively. Describe a regular expression for the language $\mathit{insert1}(L(r_1 + r_2))$ using $r_1, r_2, r'_1, r'_2$.

iii. Same as the previous part but now consider $L(r_1 r_2)$.

iv. Same as the previous part but now consider $L((r_1)^*)$.

**6.** Recall that for any language $L$, $\overline{L} = \Sigma^* - L$ is the complement of $L$. In particular, for any NFA $N$, $\overline{L(N)}$ is the complement of $L(N)$.
Let $N = (Q, \Sigma, \delta, s, A)$ be an NFA, and define the NFA $N_{\text{comp}} = (Q, \Sigma, \delta, s, Q \setminus A)$. In other words we simply complemented the accepting states of $N$ to obtain $N_{\text{comp}}$. Note that if $M$ is DFA then $M_{\text{comp}}$ accepts $\Sigma^* - L(M)$. However things are trickier with NFAs.

(a) Describe a concrete example of a machine $N$ to show that $L(N_{\text{comp}}) \neq \overline{L(N)}$. You need to explain for your machine $N$ what $\overline{L(N)}$ and $L(N_{\text{comp}})$ are.

(b) Define an NFA that accepts $\overline{L(N)} - L(N_{\text{comp}})$, and explain how it works.

(c) Define an NFA that accepts $L(N_{\text{comp}}) - \overline{L(N)}$, and explain how it works.

*Hint:* For all three parts it is useful to classify strings in $\Sigma^*$ based on whether $N$ takes them to accepting and non-accepting states from $s$.

**7.** Let $L$ be an arbitrary regular language. Prove that the language $\mathit{half}(L) := \{w \mid ww \in L\}$ is also regular.

*Solution:* Let $M = (\Sigma, Q, s, A, \delta)$ be an arbitrary DFA that accepts $L$. We define a new NFA $M' = (\Sigma, Q', s', A', \delta')$ with $\varepsilon$-transitions that accepts $\mathit{half}(L)$, as follows:

$$\begin{aligned} Q' &= (Q \times Q \times Q) \cup \{s'\} \\ s' &\text{ is an explicit state in } Q' \\ A' &= \{(h, h, q) \mid h \in Q \text{ and } q \in A\} \\ \delta'(s', \varepsilon) &= \{(s, h, h) \mid h \in Q\} \\ \delta'((p, h, q), a) &= \big\{\big(\delta(p, a), h, \delta(q, a)\big)\big\} \end{aligned}$$

$M'$ reads its input string $w$ and simulates $M$ reading the input string $ww$. Specifically, $M'$ simultaneously simulates two copies of $M$, one reading the left half of $ww$ starting at the usual start state $s$, and the other reading the right half of $ww$ starting at some intermediate state $h$.

- The new start state $s'$ non-deterministically guesses the "halfway" state $h = \delta^*(s, w)$ without reading any input; this is the only non-determinism in $M'$.
- State $(p, h, q)$ means the following:
  - The left copy of $M$ (which started at state $s$) is now in state $p$.
  - The initial guess for the halfway state is $h$.
  - The right copy of $M$ (which started at state $h$) is now in state $q$.
- $M'$ accepts if and only if the left copy of $M$ ends at state $h$ (so the initial non-deterministic guess $h = \delta^*(s, w)$ was correct) and the right copy of $M$ ends in an accepting state.

*Rubric:* 5 points =

+ 1 for a formal, complete, and unambiguous description of a DFA or NFA
  - No points for the rest of the problem if this is missing.
+ 3 for a correct NFA
  - −1 for a single mistake in the description (for example a typo)
+ 1 for a *brief* English justification. We explicitly do *not* want a formal proof of correctness, but we do want one or two sentences explaining how the NFA works.