CpSc 421 Homework 2 **Due: Oct. 2**

1. **(25 points)** Sipser, problem 1.31.
   For any string $w = w_1 w_2 \ldots w_n$, the **reverse** of w, written $w^{\mathcal{R}}$, is the string w in reverse order, $w^{\mathcal{R}} = w_n \ldots w_2 w_1$. For any language A, let
$$A^{\mathcal{R}} \quad = \quad \{w^{\mathcal{R}} \mid w \in A\}.$$
   Prove that if A is regular, then $A^{\mathcal{R}}$ is regular as well.

Solution: Construct an NFA for $A^{\mathcal{R}}$.

Because A is regular, we can represent it with an DFA. If we reverse the arcs between states and swap the start and accepting states, we get an NFA that recognizes $A^{\mathcal{R}}$. In the stuff that follows, I'll formalize this description, take care of a few technical details, and then prove that it works as advertised.

A is an regular language. Let $M = (Q, \Sigma, \delta, q_0, F)$ be an DFA such that $L(M) = A$. Choose q_x such that $q_x \notin Q$ (i.e. q_x is a new state), and let $N = (Q \cup \{q_x\}, \Sigma, \delta^{\mathcal{R}}, q_x, \{q_0\})$, where

$$\begin{array}{rcll} \delta^{\mathcal{R}}(q,c) & = & \{p \mid \delta(p,c) = q\}, & \text{reverse the arcs, } q \neq q_x \\ \delta^{\mathcal{R}}(q_x,\epsilon) & = & F, & \text{start with an } \epsilon \text{ move to a final state of } M \\ \delta^{\mathcal{R}}(q_x,c) & = & \emptyset, & \text{force that initial } \epsilon \text{ move} \end{array}$$

I'll now prove that $L(N) = A^{\mathcal{R}}$. Because N is an NFA, $L(N)$ is regular. Thus, this will show that $A^{\mathcal{R}}$ is regular.

The key to the proof is that after reading some string, $w^{\mathcal{R}}$, the set of possible states of N are exactly those states from which M could read w and reach an accepting state. The proof is by induction on w.

Induction Hypothesis: $p \in (\delta^{\mathcal{R}}(\{q_x\}, w^{\mathcal{R}}) \cap Q) \;\Leftrightarrow\; \delta(p,w) \in F$.

Base case, $w = \epsilon$:

$$\begin{array}{lll} & p \in \delta^{\mathcal{R}}(\{q_x\}, w^{\mathcal{R}}) \cap Q & \\ \Leftrightarrow & p \in \delta^{\mathcal{R}}(\{q_x\}, \epsilon^{\mathcal{R}}) \cap Q, & w = \epsilon \\ \Leftrightarrow & p \in \delta^{\mathcal{R}}(\{q_x\}, \epsilon) \cap Q, & \epsilon = \epsilon^{\mathcal{R}} \\ \Leftrightarrow & p \in (\{q_x\} \cup F) \cap Q, & \text{For any set, } B,\ \delta^{\mathcal{R}}(B,\epsilon) = B \\ \Leftrightarrow & (p \in F) & (F \subseteq Q) \wedge (q_x \notin Q) \\ \Leftrightarrow & \delta(p,\epsilon) = \in F, & \text{For any state, } q,\ \delta(q,\epsilon) = q \\ \Box & & \end{array}$$

I showed all of the steps for completeness. It would be sufficient to write:

$$\begin{array}{ll} & p \in \delta^{\mathcal{R}}(\{q_x\}, \epsilon) \cap Q \\ \Leftrightarrow & p \in F \\ \Leftrightarrow & \delta(p,\epsilon) \in F \end{array}$$

Induction step, $w = \mathtt{c} \cdot x$: Noting that $(\mathtt{c} \cdot x)^{\mathcal{R}} = x^{\mathcal{R}} \cdot \mathtt{c}$, we need to prove

$$\begin{array}{lcl} & p \in \delta^{\mathcal{R}}(\{q_x\}, x^{\mathcal{R}} \cdot \mathtt{c}) & \\ \Leftrightarrow & \exists r \in \delta^{\mathcal{R}}(\{q_x\}, x^{\mathcal{R}}).\ p \in \delta^{\mathcal{R}}(r,c), & \text{def. } \delta^{\mathcal{R}} \text{ for strings} \\ \Leftrightarrow & \exists r \in \delta^{\mathcal{R}}(\{q_x\}, x^{\mathcal{R}}).\ \delta(p,c) = r, & \text{def. } \delta^{\mathcal{R}} \text{ for symbols} \\ \Leftrightarrow & \delta(p,c) \in \delta^{\mathcal{R}}(\{q_x\}, x^{\mathcal{R}}) & \\ \Leftrightarrow & \delta(\delta(p,c),x) \in F, & \text{induction hypothesis} \\ \Leftrightarrow & \delta(p, c \cdot x) \in F, & \text{def. } \delta \text{ for strings} \end{array}$$

Intuitively, what this argument says is that if N can reach some state, p, by reading $x^{\mathcal{R}} \cdot c$; then it did it by first reaching some state, r, by reading $x^{\mathcal{R}}$, and then got to state p by reading c. We then take advantage that $\delta^{\mathcal{R}}$ is the reversal of δ. Thus, M will go from p to r by reading c. Finally, we use the induction hypothesis with r and x to conclude that M will go from r to some state in F by reading x. I will accept an intuitive argument like this one, or the mathematical version that I stated first.

2. **(25 points)** Sipser, problem 1.32.

Let

$$\Sigma_3 = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \cdots \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}.$$

Σ_3 contains all size 3 columns of 0s and 1s. A string of symbols in Σ_3 gives three rows of 0s and 1s. Consider each row to be a binary number with the most significant bit first. For example, let

$$w = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

The first row of w is the binary representation of 7, the second row corresponds to 5, and the third row corresponds to 12.

Let

$$B = \{w \in \Sigma_3^* \mid \text{the bottom row of } w \text{ is the sum of the top two rows}\}.$$

Show that B is regular. (Hint: Working with $B^{\mathcal{R}}$ is easier. You can use the result that you were asked to prove for question 1).

Solution:

The language B is regular. I'll construct an DFA for $B^{\mathcal{R}}$. This proves that $B^{\mathcal{R}}$ is regular. Having shown in the previous problem that the regular languages are closed under reversal, this shows that B is regular as well.

To avoid lots of vertical blank space, I'll write the the triple (a, b, c) to mean $\begin{bmatrix} a \\ b \\ c \end{bmatrix}$.

First, I'll define the DFA. Note that $B^{\mathcal{R}}$ inputs the bits starting at the least-significant-bit. It basically computes the sum of the carry from the previous bit and the two input bits and checks to make sure that the third input bit agrees. It then moves to state q_0 or q_1 according to the carry. If a sum bit of the input is wrong, then the DFA moves to state q_x, a garbage state.

$$\begin{aligned} M &= (Q, \Sigma_3, \delta, q_0, F) \\ Q &= \{q_0, q_1, q_x\} \\ \delta\{q_i, (a, b, i \oplus a \oplus b) &= q_{\mathsf{maj}(i,a,b)} \\ \delta\{q_i, (a, b, \neg i \oplus a \oplus b) &= q_x \\ F &= \{q_0\} \end{aligned}$$

where $\oplus$ represent exclusive-OR and $\mathsf{maj}(x, y, z)$ is 1 if at least two of x, y, and z are 1, and zero otherwise (the "majority" function).

State q_0 represents the situation where the carry from the previous bit is 0, and q_1 indicates that the carry from the previous bit is 1. I wrote the transition function using $\oplus$ and maj. If you prefer, here it is as a table:

$$\begin{array}{l}
\text{Transitions for correct sums:} \\
\quad \delta(q_0,(0,0,0)) \;=\; \{q_0\}, \quad \delta(q_0,(0,1,1)) \;=\; \{q_0\}, \\
\quad \delta(q_0,(1,0,1)) \;=\; \{q_0\}, \quad \delta(q_0,(1,1,0)) \;=\; \{q_1\}, \\
\quad \delta(q_1,(0,0,1)) \;=\; \{q_0\}, \quad \delta(q_1,(0,1,0)) \;=\; \{q_1\}, \\
\quad \delta(q_1,(1,0,0)) \;=\; \{q_1\}, \quad \delta(q_1,(1,1,1)) \;=\; \{q_1\} \\
\hline
\text{Transitions for incorrect sums:} \\
\quad \delta(q_0,(0,0,1)) \;=\; \{q_x\}, \quad \delta(q_0,(0,1,0)) \;=\; \{q_x\}, \\
\quad \delta(q_0,(1,0,0)) \;=\; \{q_x\}, \quad \delta(q_0,(1,1,1)) \;=\; \{q_x\}, \\
\quad \delta(q_1,(0,0,0)) \;=\; \{q_x\}, \quad \delta(q_1,(0,1,1)) \;=\; \{q_x\}, \\
\quad \delta(q_1,(1,0,1)) \;=\; \{q_x\}, \quad \delta(q_1,(1,1,0)) \;=\; \{q_x\}, \\
\text{Stuck in the garbage state:} \\
\qquad \delta(q_1,c) \;=\; \{q_x\}.
\end{array}$$

Where c can be any symbol in Σ_3. For example $\delta(q_1,(0,1,0)) = \{q_1\}$ says that if the DFA is in state q_1 (a carry from the previous bit) and the first operand bit is a 0 and the second operand bit is a 1, then (in binary, $1+0+1 = 10$), the sum bit should be a 0 (the third bit of $(0,1,0)$), and the carry is 1 (the machine transitions to state q_1).

It is straightforward (albiet tedious) to verify that each transition does the right thing. Thus, $L(M) = B^{\mathcal{R}}$ which completes the proof.

An acceptable solution will just specify the DFA (or NFA) with a short explanation. It's enough to say that the machine has two states to keep track of $\mathsf{carry} = 0$ and $\mathsf{carry} = 1$, and that at each step, it checks to make sure that the sum is consistent with the input bits and carry, and the machine rejects if it sees an error.

3. **(25 points)** Let u and v be strings with $|u| = |v|$. We define $\mathit{weave}(u,v)$ as shown below:

$$\begin{array}{rcl}
\mathit{weave}(\epsilon,\epsilon) & = & \epsilon \\
\mathit{weave}(x\cdot a, y\cdot b) & = & \mathit{weave}(x,y)\cdot a\cdot b
\end{array}$$

For example, $\mathit{weave}(\texttt{cat},\texttt{dog}) = \texttt{cdaotg}$ and $\mathit{weave}(\texttt{srn},\texttt{tig}) = \texttt{string}$.

For any language A, let

$$\mathit{half}(A) \quad = \quad \{u \mid \exists v \in \Sigma^{|u|}.\ \mathit{weave}(u,v) \in A\}$$

Prove that if A is regular, then $\mathit{half}(A)$ is regular as well.

Solution: Let $M = (Q,\Sigma,\delta,q_0,F)$ be a DFA that recognizes A. Let $N = (Q,\Sigma,\delta',q_0,F)$ be an NFA with

$$\delta'(q,c) \quad = \quad \{p \mid \exists d \in \Sigma.\ p = \delta(\delta(q,c),d)\}$$

In English, p is in $\delta'(q,c)$ iff M can move from q to p first by reading c and then by reading *some* other symbol from Σ.

Claim: $L(N) = \mathit{half}(A)$.

Proof: As Sipser would say, the construction is obviously correct.

If you don't believe me, you can first prove that any string $w \in L(N)$ is also in $\mathit{half}(A)$. You do this by induction on w. The induction hypothesis is

$$q \in \delta'(q_0,u) \quad \Rightarrow \quad \exists v \in \Sigma^{|u|}.\ \delta(q_0, \mathit{weave}(u,v) = q$$

This is clearly true for $w = \epsilon$. The induction step uses the $\exists d$ from the definition of δ'. The symbol d is the one that you need to append to the v string. The proof that any $w \in \mathit{half}(A)$ is also in $L(N)$ is very similar.

A solution that says that the construction is obviously correct is acceptable.

4. **(10 points)** (from Sipser 1.20).
For each of the following languages, give two strings that are members and two strings that are *not* members – a total of four strings for each part. Assume that the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ in all parts.

   (a) $\mathtt{a}^*\mathtt{b}^*$.

   Solution:

   Two strings in the language: ϵ, aaabbbbb.

   Two strings not in the language: ba, aba.

   (b) $\mathtt{a}^*(ba)^*\mathtt{b}^*$.

   Solution:

   Two strings in the language: ϵ, aabababb.

   Two strings not in the language: baa, ba.

   (c) $\mathtt{a}^* \cup \mathtt{b}^*$.

   Solution:

   Two strings in the language: ϵ, aa.

   Two strings not in the language: ab, ba.

   (d) $(\mathtt{aaa})^*$.

   Solution:

   Two strings in the language: ϵ, aaa.

   Two strings not in the language: a, b.

   (h) $(\mathtt{a} \cup \mathtt{ba} \cup \mathtt{bb})\Sigma^*$.

   Solution:

   Two strings in the language: a, abbabab.

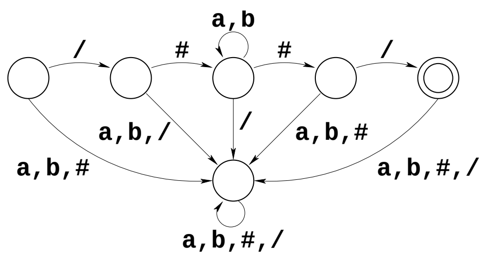
   Two strings not in the language: ϵ, b.

5. **(15 points)** (from Sipser 1.22).
In certain programming languages, comments appear between delimiters such as `/#` and `/#`. (We're using `/#` instead of `/*` as for comments in C to avoid confusion of the character `*` with the regular expression operator, *.) Let C be the language of all valid delimited comment strings. A member of C must begin with `/#` and end with `#/` but have no intervening `#/`. For simplicity, we'll say that comments themselves are written with only the symbols a, b; hence the alphabet of C is $\Sigma = \{\mathtt{a}, \mathtt{b}, /, \#\}$.

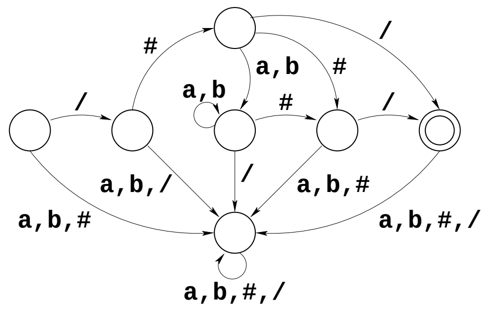
   Note: As described by Sipser, the text of the comment cannot contain the symbols `/` or `#`. Thus, `/#abbabaab#/` is a valid comment but `/#ab/babb#ba##/` is not a valid comment. You may make this assumption in your solution – it makes the solution easier.

   (a) Give a DFA that recognized C.

   Solution: Sipser's phrasing doesn't say whether or not the `#` in the `/#` has to be different than the one in the `#/`. In Java, they have to be different (i.e. `/*/` is not a valid comment in Java). I'll show one DFA for both versions. The version where `/#/` is not a comment is simpler, and I assume that it's the one Sipser had in mind. Either solution is acceptable.



Comments excluding /#/



Comments including /#/

(b) Give a regular expression that generates C.

Solution: Again, there are two ways to interpret the problem. I'll present solutions for both. Either is acceptable.

Comments excluding /#/: /#$(a \cup b)^*$#/

Comments including /#/: /#$(a \cup b)^*$#/ $\cup$ /#/