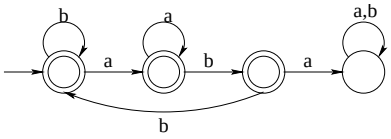
# Solutions to Problem Set 1

1. Prove that the following languages are regular, either by exhibiting a regular expression representing the language, or a DFA/NFA that recognizes the language:
**[10 x 3 = 30 points]**

   (a) all strings that do not contain the substring *aba*, for $\Sigma = \{a, b\}$ (for instance, *aabaa* contains the substring *aba*, whereas *abba* does not)

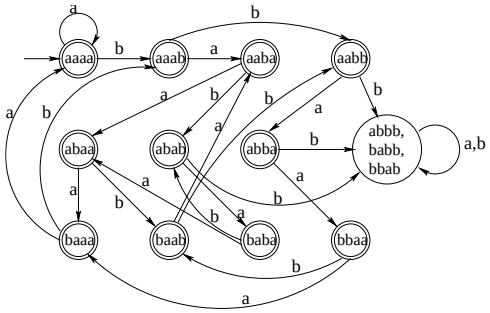
   SOLUTION: The following machine recognizes the given language by maintaining a state for "how much" of the string *aba* it has seen. On seeing *aba* it goes into a non-accepting state and stays there.




   (7 points for the DFA and 3 for the explanation.)

   (b) set of strings such that each block of 4 consecutive symbols contains at least two $a$'s, for $\Sigma = \{a, b\}$

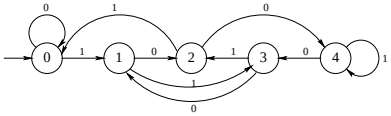
   SOLUTION: The following machine remembers the last four characters it has read from the string. The names of the states indicate the (length four) blocks they represent.




   (7 points for the DFA and 3 for the explanation.)

   (c) set of binary strings ($\Sigma = \{0, 1\}$) which when interpreted as a number (with the most significant bit on the left), are divisible by 5.

   SOLUTION: We maintain the remainder of the number read so far, when divided by 5. To update the remainder, note that if $x$ is the number read so far, and $b$ is the new bit that is read then the new number is $y = 2x + b$ and $y \mod 5 = ((2x \mod 5) + b) \mod 5$. (6 points for the DFA and 4 for the explanation.)



2. (Sipser, problem 1.31) For any string $w = w_1 w_2 \cdots w_n$, the reverse of $w$, written as $w^R$ is the string $w$ in reverse order, $w_n \cdots w_2 w_1$. For any language $A$, let $A^R = \{w^R \mid w \in A\}$. Show that if $A$ is regular, so is $A^R$.
**[20 points]**

SOLUTION: One solution is recursively (or inductively) define a reversing operation on regular expressions, and apply that operation on the regular expression for $A$. In particular, given a regular expression $R$, $\mathsf{reverse}(R)$ is:

- $a$ for some $a \in \Sigma$,
- $\epsilon$ if $R = \epsilon$,
- $\emptyset$ if $R = \emptyset$,
- $(\mathsf{reverse}(R_1) \cup \mathsf{reverse}(R_2))$, if $R = R_1 \cup R_2$,
- $(\mathsf{reverse}(R_2) \circ \mathsf{reverse}(R_1))$ if $R = R_1 \circ R_2$, or
- $(\mathsf{reverse}(R_1)^*)$, if $R = (R_1^*)$.

(8 points for saying reversing the regular expression, and 12 points for explaining how it's done. It's important to point out that the operation is performed recursively.)

Another solution is to start with a DFA $M$ for $A$, and build a NFA $M'$ for $A^R$ as follows: reverse all the arrows of $M$, and designate the start state for $M$ as the only accept state $q'_{acc}$ for $M$'. Add a new start state $q'_0$ for $M'$, and from $q'_0$, add $\epsilon$-transitions to each state of $M'$ corresponding to accept states of $M$.

It is easy to verify that for any $w \in \Sigma^*$, there is a path following $w$ from the state start to an accept state in $M$ iff there is a path following $w^R$ from $q'_0$ to $q'_{acc}$ in $M'$. It follows that $w \in A$ iff $w^R \in A^R$.

(7 points for saying reversing the arrows; 3 points for explaining the new accept state, and 5 points for explaining the new start state and the $\epsilon$-transitions. 5 points for explaining, or at least making the final observation about the paths/connectivity.)

3. We say a string $w = w_1 w_2 \ldots w_n$ is a *shuffle* of strings $u$ and $v$ if there exists $J \subseteq \{1, \ldots, n\}$ such that $(w_j)_{j \in J} = u$ and $(w_j)_{j \notin J} = v$. For example CSS17PR2ING07 is a shuffle of the strings CS172 and SPRING07 and in fact, there are two sets $J = \{1, 2, 4, 5, 8\}$ and $J = \{1, 3, 4, 5, 8\}$ which work here.

We then define the shuffle of two languages $A$ and $B$ as

$$S(A, B) = \{w | \exists\ u \in A, v \in B \text{ s.t. } w \text{ is a shuffle of } u \text{ and } v\}$$

Show that if $A$ and $B$ are regular languages over a common alphabet $\Sigma$, then so is $S(A, B)$.
**[20 points]**

SOLUTION: Let $M_A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$ and $M_B = (Q_B, \Sigma, \delta_B, q_{0B}, F_B)$ be two DFAs accepting the languages $A$ and $B$ respectively. Then we define an NFA $M = (Q, \Sigma, \delta, q_0, F)$ for $S(A, B)$ as follows.

Let $Q = Q_A \times Q_B$, $q_0 = (q_{0A}, q_{0B})$ and $F = F_A \times F_B$. Define $\delta((q_A, q_B), s) = \{(\delta_A(q_A, s), q_B)\} \cup \{(q_A, \delta_B(q_B, s))\}$, i.e., at each step, the machine changes $q_A$ according to $\delta_A$ or $q_B$ according to $\delta_B$. It reaches a state in $F_A \times F_B$ if and only if the moves according to $\delta_A$ take it from $q_{0A}$ to a state in $F_A$, and the ones according to $\delta_B$ take it from $q_{0B}$ to a state in $F_B$. Hence $M$ accepts exactly the language $S(A, B)$.
(12 points for designing the machine and 8 for the argument.)