Mod 2

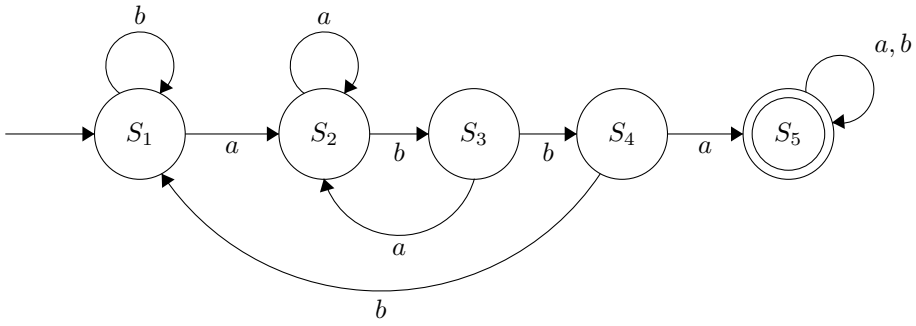
Due on February 20, 2025
Tuesday/Thursday 11:00-12:15, Warner 209

Mark Floryan - Section 001

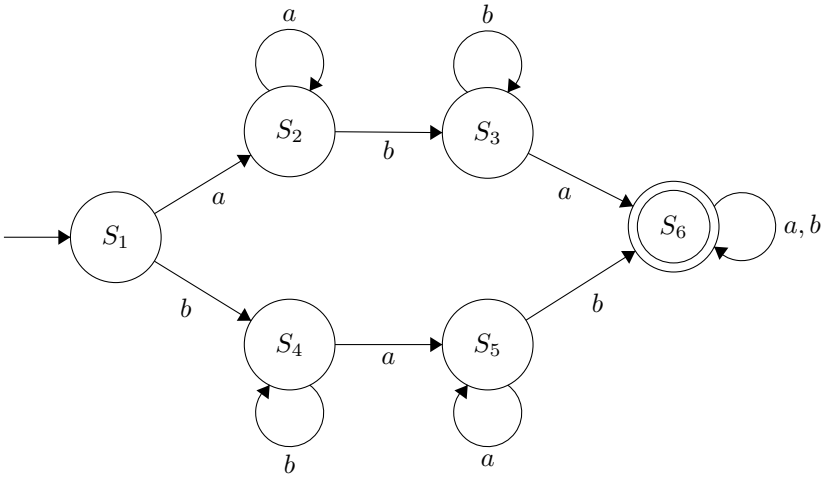
Rushil Umaretiya
frj2ka@virginia.edu

#1 Draw out *DFA*s (not *NFA*s) for each of the following languages. For some of these, a small hint is provided. Your goal is to construct a *DFA* with as few states as possible (just like how we prefer to write succinct code when possible). For all of these, let $\Sigma = \{a, b\}$

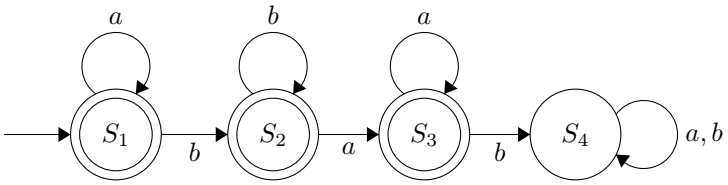
- $\{w \mid w$ does not contain the substring *abba*$\}$ (**Hint: Draw out the DFA for a simpler language that DOES contain abba and then try to change that machine slightly.*)



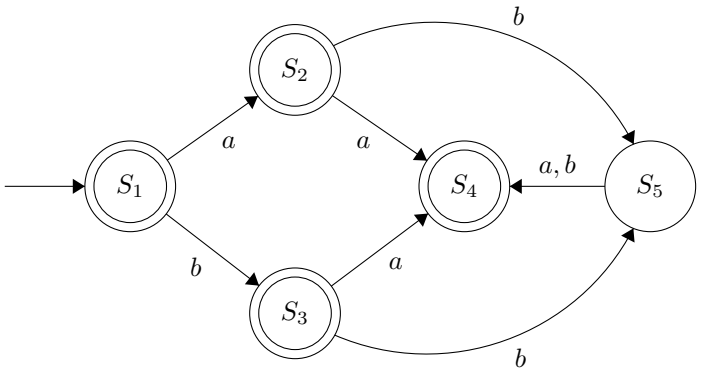
- $\{w \mid w$ contains BOTH the substrings ab and ba$\}$



- $\{w \mid w \in a^*b^*a^*\}$

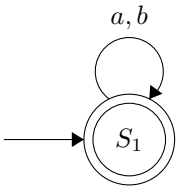


- $\{w \mid w \neq ab \wedge w \neq bb\}$



- $\{w \mid w \in a^i w' \mid i \in \mathbb{N}, w' \in \{a, b\}^*,$ w' contains at least i a's$\}$ (**Hint: This one LOOKS not regular but it actually is. Can you figure out why?*)

  Note: This will accept everything due to the w'.



#2 Prove that regular languages are closed under *intersection*. Do this by starting with *DFA*s for two regular languages A and B, and describe how to construct a new *DFA* for $A \cap B$

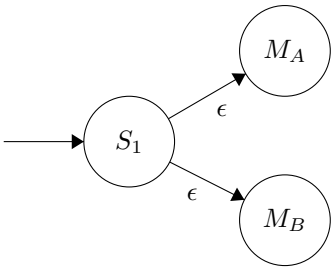
Proof. Let A and B be regular languages. Let $M_A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$ and $M_B = (Q_B, \Sigma, \delta_B, q_{0B}, F_B)$ be the *DFA*s that recognize A and B respectively. We will construct a new *DFA* $M = (Q, \Sigma, \delta, q_0, F)$ that recognizes $A \cap B$.

Let

$$\begin{aligned} Q &= Q_A \times Q_B \\ \delta((q_A, q_B), a) &= (\delta_A(q_A, a), \delta_B(q_B, a)) \\ q_0 &= (q_{0A}, q_{0B}) \\ F &= F_A \times F_B \end{aligned}$$

Let $w \in A \cap B$. Then $w \in A$ and $w \in B$. Since M_A and M_B recognize A and B respectively, M_A will accept w and M_B will accept w. Thus, M will accept w as well. Thus, $A \cap B$ is regular. □

An alternative way to prove that regular languages are closed under intersection is to acknowledge that since regular languages can be represented by NFAs, we can create an NFA to represent the unison of the two languages.



In this diagram, S_1 is the start state, M_A is the NFA that recognizes A, and M_B is the NFA that recognizes B. The ϵ transitions are used to combine the two NFAs into one. This new NFA will recognize the intersection of A and B.

#3 Prove that regular languages are closed under *complement*. Do this by starting with a *DFA* for a regular language A, and describe how to construct a new *DFA* for $\bar{A}$.

Proof. Let A be a regular language. Let $M_A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$ be the *DFA* that recognizes A. We will construct a new *DFA* $M = (Q, \Sigma, \delta, q_0, F)$ that recognizes $\bar{A}$.

Let

$$\begin{aligned} Q &= Q_A \\ \delta(q, a) &= \delta_A(q, a) \\ q_0 &= q_{0A} \\ F &= Q_A - F_A \end{aligned}$$

Let $w \in \bar{A}$. Then $w \notin A$. Since M_A recognizes A, M_A will not accept w. Thus, M will accept w as well. Thus, $\bar{A}$ is regular. □

#4 For any string $w = w_1 w_2, ..., w_n$, let w^R be the reverse of string w (i.e., $w^R = w_n, ..., w_2, w_1$). Prove that if a language A is regular, then the language $A^R = \{w^R \mid w \in A\}$ is also regular.

Proof. Let A be a regular language. Let $M_A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$ be the *DFA* that recognizes A. We will construct a new *DFA* $M = (Q, \Sigma, \delta, q_0, F)$ that recognizes A^R.

First, we can reverse each of the transitions from our original *DFA* M_A. This will allow us to read the string in reverse. We can also reverse the start and accept states. However, since there are multiple accept states, we will need to create a new start state ($q_0^{'}$) that ϵ transitions to each of the original accept states. This will give us a formal definition of M as follows:

This will give us a formal definition of M as follows:

$$
\begin{aligned}
Q &= Q_A \cup \{q_0^{'}\} \\
\delta'(q,a) &= \begin{cases} \{\, p \in Q \mid q \in \delta(p,a) \,\} & \text{if } q \in Q \text{ and } a \in \Sigma, \\ F & \text{if } q = q_0' \text{ and } a = \epsilon, \\ \emptyset & \text{otherwise.} \end{cases} \\
q_0 &= q_0^{'} \\
F &= q_{0A}
\end{aligned}
$$

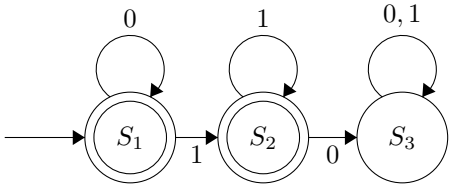
Since the NFA, M recognizes A^R, A^R is regular.

□

#5 Use the pumping lemma to show that the following languages are not regular OR argue that they are regular.

- $A = \{0^*0^n1^n1^* \mid n \geq 0\}$

  Proof. Notice that if we were to take $n = 0$, we would end up with a string of the form 0^*1^*. Every string in this language would be comprised of a possibly empty string of zeroes followed by a string of ones, any section of which can be pumped. This language can also be represented by the following DFA:

  Therefore the language, A, is regular. □

- $B = \{www \mid w \in \{0,1\}^*\}$

  Proof. Assume B is regular. Let p be the pumping length. Let $w = 0^p1^p0^p1^p0^p1^p$. If we were to remove any substring of w by setting $i = 0$ under the constraints of the pumping lemma ($xy^iz \in B$), we would end up with a string that is not in B as it would not ever be symmetrical as the language implies.

  The same follows for the opposite. If we attempted to pump any substring of our string, we would end up with a string that is not in B as it would not ever be symmetrical as the front half would be larger than the back half.

  We have found a contradiction according to the pumping lemma. Therefore, B is not regular. □

#6 Find and describe the error that exists in the following proof. The proof attempts to show that 0^*1^* is not regular, when in fact it is:

Assume, for sake of contradiction, that 0^*1^* *is regular. We select an element from this language that is greater than the pumping length p. We select* 0^p1^p*. In class, when proving that* 0^n1^n *was not regular, we showed that* 0^p1^p *cannot be pumped. Therefore,* 0^*1^* *is not regular.*

In class, when we showed that 0^p1^p could not be pumped, it was under the constraint that there had to be an equal number of zeroes and ones. In the case of 0^*1^*, there is no such constraint and we are free to pump any substring of the string.
In fact we can show this by the three cases of substrings that we can pump:

- If we pump the leading zeroes, we will end up with a string that is still in the language.
- If we pump the lagging ones, we will end up with a string that is still in the language.
- If we pump some combination of the zeroes and ones, we will end up with a string that is still in the language.

Therefore, 0^*1^* is regular.