Automata

Review

A **string** is a sequence of symbols, each coming from a finite **alphabet** Σ. The **length** of a string is the number of symbols that it contains.

We use λ to denote the **empty string**, the unique string of length 0.

We omit the parentheses we normally use to denote sequences when we write sequences. We also often use superscripts to denote multiple, consecutive copies of the same symbol. For example a^2b^3 is shorthand for the string $aabbb$, which itself represents the symbol sequence (a, a, b, b, b).

A **language** is a set of strings over an alphabet. We define the following operations which allow us to build more complicated languages from simpler ones:

- **Union:** Given languages A and B, $A \cup B$ is the usual set union.
- **Concatenation:** Given languages A and B, AB is the language consisting of all concatenations of strings from A with strings from B (that is, all strings formed by writing the symbols in B's string after the symbols in A's string).
- **Repetition:** Given a string A and a constant $n \in \mathbb{N}$, A^n is the language of all strings formed by concatenating n strings from A. Additionally, A^* is defined as $\bigcup_{n \in \mathbb{N}} A^n$.

A **deterministic finite automaton** (DFA) is a simple computational model that receives a string as input and outputs whether to accept or reject that string. The language of a DFA, M (denoted $L(M)$) consists of all strings that M accepts.

Formally, a DFA M is a 5-tuple $M = (S, \Sigma, f, s_0, F)$, where:

- S is a finite set of states.
- $s_0 \in S$ is a designated initial state.
- $F \subseteq S$ are the final or accepting states.
- Σ is the finite input alphabet.
- $f : S \times \Sigma \to S$ is the transition function.

We often visualize a DFA as a directed graph with labeled edges, as shown later on this worksheet. The nodes of this graph represent the states and the arrows represent the state transitions.

To process a string, a DFA begins in its initial state. Then, for each character in the string (working from left to right) it jumps to its next state based upon its current state and the character it reads as ascribed by the transition function. Once it reaches the end of the string, the DFA accepts the string if it is sitting in a final state; otherwise, it rejects the string.

There are some languages that a DFA can recognize, called the **regular languages**, and other languages that it cannot. The **pumping lemma** can be used to argue that a language is not regular. The lemma states:

> Given a DFA M with k states, suppose that M accepts a string x with $|x| \geq k$. Then, there are strings u, v, w such that $x = uvw$, $|uv| \leq k$, $|v| \geq 1$, and $uv^i w \in L(M)$ for all $i \in \mathbb{N}$.

To use the pumping lemma, we assume that there is some machine M recognizing a language L, exhibit a string $x \in L$, and show that regardless of the breakdown of x into u, v, w, there is some i for which $uv^i w \notin L$.

A **non-deterministic finite automaton** (NFA) is a generalization of a DFA where our transition function is of the form $f : S \times \Sigma \to \mathcal{P}(S)$, so we may transition to any number of states from a state upon reading a character.

An NFA accepts a string if there is some path of arcs labeled with the characters of the string that leads from the initial state to a final state.

Although an NFA may initially seem more powerful than a DFA, the **subset construction** shows that they recognize the same set of languages. However, a DFA may require exponentially many more states to do this.

Regular expressions give an algebraic way to define a language. We define the set of regular expressions over Σ recursively, as follows:

- The symbols $\emptyset$ and λ are regular expressions.
- The symbol x is a regular expression if $x \in \Sigma$.
- Given regular expressions A and B, $A \cup B$ (alternation), AB (concatenation), and A^* (Kleene star) are also regular expressions.

Here, the Kleene star has precedence over concatenation, which has precedence over alternation. Parentheses can be used to adjust the order of operations.

We define the languages of regular expressions recursively as well. As our bases cases, $L(\emptyset) = \emptyset, L(\lambda) = \lambda$, and $L(x) = x \quad \forall x \in \Sigma$. Then, for any regular expressions A and B,

$$L(AB) = L(A)L(B), \qquad L(A \cup B) = L(A) \cup L(B), \qquad L(A^*) = L(A)^*.$$

Kleene's theorem states that L is the language of a regular expression if and only if it is the language of some DFA (so also of some NFA).

1. Give a description of each of the following languages, defined over the alphabet $\Sigma = \{a, b\}$.

(a) ab^*

(b) $\{ab\}^*$

(c) $\{a, b\}^*$

(d) a^*b^*

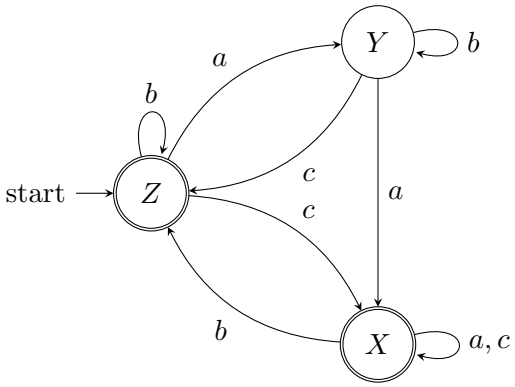
(e) a^*ba^*

(f) $\{aa, ab, ba, bb\}^*$

(g) $\{ab\}^n$

(h) $\{a, b\}^n$

2. Give a formal description of the DFA pictured below.



$S =$

$s_0 =$

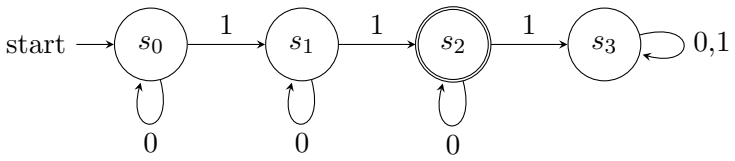
$F =$

$\Sigma =$

$f :$

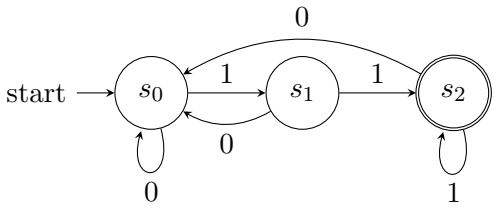
3. For each of the following DFAs, describe the language that they recognize.

(a) M_1:



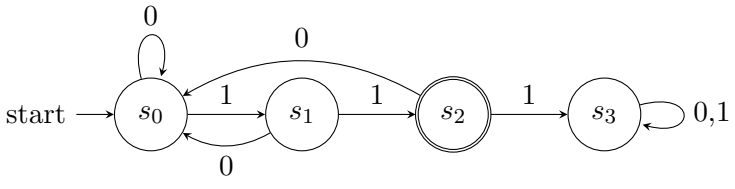
$L(M_1) =$

(b) M_2:



$L(M_2) =$

(c) M_3:



$L(M_3) =$

4. In this question, we'll construct a 3-state DFA M such that $L(M)$ is the binary strings that are multiples of 3 (here, we'll count the empty string as a multiple of 3.

(a) What information should we use each of our three states to encode? (**Hint:** Think back to our discussion of number theory.)

(b) Which state will be our initial state, and which state(s) will be accepting?

(c) We will receive the bits in the binary representation from left to right. Suppose we are given a binary representation for the number n. If we append a 0 to the (right) end of this representation, what number does it now represent? How should we transition between our states when we receive a 0.

(d) Repeat part (d) in the case that a 1 arrives.

(e) Use the information above to draw the DFA M.

5. In this question, we'll use the Pumping Lemma to prove that the following language is not regular:

$$L_1 = \{0^n 10^n : n \in \mathbb{N}\}$$

For sake of contradiction, we'll suppose that there is a DFA, M that recognizes the language L_1, and we'll let k be the number of states in M.

By assumption, $0^k 10^k$ belongs to $L(M)$.

(a) By the Pumping Lemma, there exist strings u, v, w, such that what four properties hold.

1.

2.

3.

4.

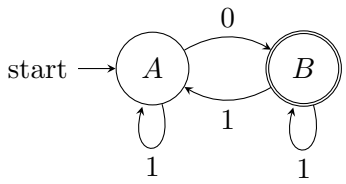
(b) Based upon these criteria, what can we conclude about the string v?

(c) Use your observation from part (b) and your criteria from part (a) to construct a string in $L(M)$ which is not in L_1.

Thus, our assumption that there is a DFA M that recognizes L_1 must be incorrect, so M must not be a regular language.

6. Use the Pumping Lemma to prove that the language $L_2 = \{0^m 1^n : m > n \in \mathbb{N}\}$ is not regular.

7. Here, we'll use the subset construction to convert the following NFA into a DFA.

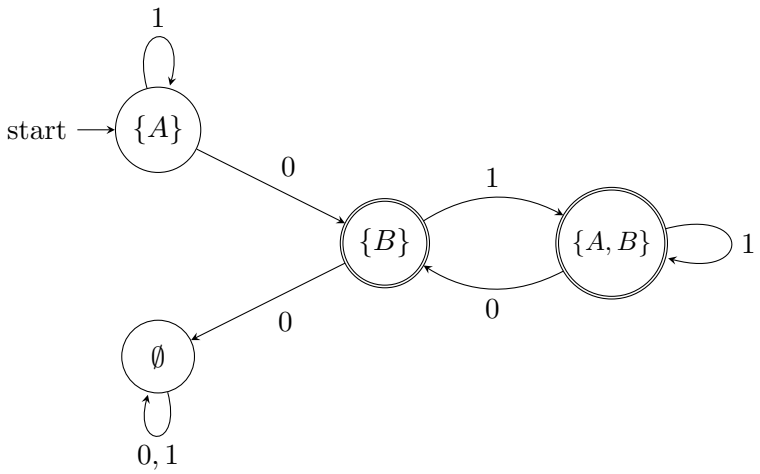


(a) Describe the following components of the DFA $M = (S, s_0, F, \Sigma, f)$.

$S =$ $s_0 =$

$F =$ $\Sigma =$

(b) Draw the DFA, M.



(c) Describe $L(M)$.

8. For each of the following pairs of regular expressions, determine whether they describe the same language, whether the language of one is contained in the other, or neither is contained in the other.

(a) $R_1 = 0(0^*1)^*$ $R_2 = (01^*)^*$

(b) $R_1 = (0 \cup 1)^*$ $R_2 = (0^*1^*)^*$

(c) $R_1 = 11(111 \cup 11)^*$ $R_2 = 111^*$

(d) $R_1 = 1(01^*01^*)^*$ $R_2 = 1 \cup (101^*01^*)^*$

9. Draw an NFA that recognizes the regular expression $(ab \cup bc)^*a((ac)^* \cup c)^*$.