Recitation - Finite Automata

1 Regular Announcements

- Homework solution sessions: Saturdays 15:30 - 16:30, Sundays 12:30 - 13:30. If you cannot make it to the solution sessions, you can ask us about the solutions in office hours. You can also ask your mentor TA for help. Just let us know.
- Homework resubmission deadline is Wednesday 19:00 (a week after the corresponding homework-writing session).
- Homework regrade request deadline is Sunday 17:00 (4 days after the corresponding homework-writing session).
- Come talk to us if you had difficulties on Homework 1!

2 Definitions for All

- **Deterministic Finite Automaton (DFA)**: A DFA M is a machine that reads a finite-length string one symbol at a time in one pass, transitions from state to state, and ultimately accepts or rejects. Formally, M is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where
  - Q is a **finite**, **non-empty** set of states;
  - Σ is the **finite**, **non-empty** alphabet;
  - $\delta : Q \times \Sigma \to Q$ is the transition function;
  - $q_0 \in Q$ is the starting state;
  - $F \subseteq Q$ is the set of accepting states;
- Given a DFA M, $L(M)$ denotes the set of strings that the DFA accepts.

- **Regular language**: A language L is regular if $L = L(M)$ for some DFA M (M solves L).
- If L_1 and L_2 are both regular languages over Σ^*, for some fixed Σ, then the following are all regular:
  - $\overline{L_1} = \Sigma^* \setminus L_1$,
  - $L_1 \cup L_2$,
  - $L_1 \cap L_2$,
  - $L_1 L_2$ (the concatenation of two regular languages),
  - L_1^* (this is an exercise in the textbook; not an easy one though).

3 Odd Ones Out

Draw a DFA that decides the language

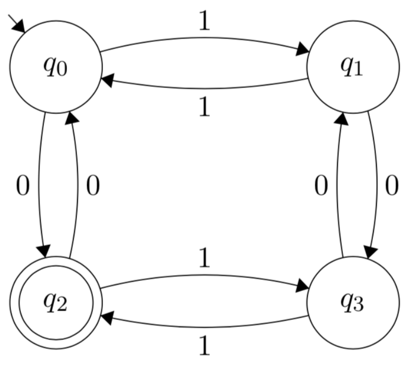
$$L = \{x : x \text{ has an even number of 1s and an odd number of 0s}\}$$

over the alphabet $\Sigma = \{0, 1\}$.

Solution. https://www.youtube.com/embed/askcdLyrSGU

We have 4 states with the following meanings. The string we have seen so far has:

- even number of 0's and even number of 1's (state q_0);
- even number of 0's and odd number of 1's (state q_1);
- odd number of 0's and even number of 1's (state q_2);
- odd number of 0's and odd number of 1's (state q_3).



■

4 Adam, I'm Ada!

Show that, if $|\Sigma| > 1$, then

$$L = \{x \mid x \in \Sigma^* \text{ and } x = x^R\}$$

is a non-regular language.

Solution. https://www.youtube.com/embed/GWsKqabEsGE

Any string x that satisfies $x = x^R$ is called a *palindrome*.

As usual, we will prove that the language is non-regular by a proof by contradiction and apply the pigeonhole principle (PHP). So assume for the sake of contradiction that there exists a DFA with k states that solves L. Consider two distinct symbols $a, b \in \Sigma$ (since we assumed $|\Sigma| > 1$). Consider the $k+1$ strings $b^n a$ for $n \in \{0, \dots, k\}$. Since there are only k states, by PHP, there must exist some i, j, $0 \leq i < j \leq k$, such that $b^i a$ and $b^j a$ end up in the same state. Thus, $b^i ab^i$ and $b^j ab^i$ must end up in the same state. However, $b^i ab^i$ is a palindrome, so it should end up in an accepting state, and $b^j ab^i$ is not a palindrome, so it should end up in a rejecting state. This is the desired contradiction since a state cannot be both accepting and rejecting. ■

5 Suffering with Suffixes

Given a word w, we say that u is a *proper suffix* of w if there is $v \neq \epsilon$ such that $w = vu$. For a language L, define

$$\text{SUFF}(L) = \{w \in L : \text{no proper suffix of } w \text{ is in } L\}.$$

Show that if L is regular, then so is SUFF(L), as follows. Give an exact description of a DFA solving SUFF(L), explicitly stating how Q, δ, q_0 and F are defined. Furthermore, briefly explain the reasoning behind your construction. A detailed proof of correctness is not needed.

Solution. https://www.youtube.com/embed/URld2_WcKyk

The construction is very similar to the construction we have seen for showing regular languages are closed under the concatenation operation. In that proof, we opened up a thread every time we encountered an accepting state of the first machine. In this case, we will open up a new thread after reading each symbol. We now give the details.

Since L is regular, we know that there is some DFA $M = (Q, \Sigma, \delta, q_0, F)$ that solves L. Recall that for $S \subseteq Q$ and $\sigma \in \Sigma$, we let $\delta_{\wp}(S, \sigma) = \{\delta(s, \sigma) : s \in S\}$. Now, we define $M' = (Q', \Sigma, \delta', q_0', F')$ that solves SUFF(L) as follows:

- $Q' = Q \times \wp(Q)$
- For any $q \in Q$, $S \subseteq Q$, and $\sigma \in \Sigma$, define $\delta'((q, S), \sigma) = (\delta(q, \sigma), \delta_{\wp}(S, \sigma) \cup \{q_0\})$
- $q_0' = (q_0, \varnothing)$
- $F' = \{(q, S) : q \in F, S \cap F = \varnothing\}$

Intuitively, the first component of the state tuple (in M') keeps track of 'the main thread' (i.e. the computation of the DFA on the whole string), while the second component keeps track of all the 'suffix threads' (i.e. the computation of the DFA on proper suffixes of the whole string). The transition function is defined with this design in mind. At each character, the main thread and each of the suffix threads takes one step; additionally, a new suffix thread is started. In the end, we accept if and only if the main thread has accepted and all the suffix threads have rejected. ■

6 States for Days (Extra Problem)

The *state complexity* of a language is the minimum number of states in any DFA solving the language. If a language is non-regular, then we define the state complexity to be infinity.

To show that a language is non-regular, we have seen a proof strategy that can be viewed as showing that the state complexity of the non-regular language is infinity. It turns out, the same kind of argument can be applied to regular languages to show a lower bound on the state complexity of the language. In this problem, our goal will be to illustrate this.

For any $k \geq 1$, let

$$R_k = \{x \mid x \in \{\texttt{0}, \texttt{1}\}^* \text{ and the } k\text{-th symbol from the right is a } \texttt{1}\}.$$

Show that any DFA that solves R_k must have at least 2^k states.

Solution. https://www.youtube.com/embed/q3N1RxmXjlU

Fix an arbitrary $k \geq 1$. We will show that any DFA solving R_k must have at least 2^k states. To do this, we will identify 2^k strings such that in any DFA solving the language, these 2^k strings, when fed into the DFA, must end up in different states.

Consider the set B_k of all binary strings of length k. Note that $|B_k| = 2^k$. We claim that in any DFA solving R_k, the strings in B_k must end up in different states. To prove this, assume for the sake of contradiction that there are two strings x and y that end up in the same state. Since $x \neq y$, there is some i such that $x_i \neq y_i$. Let's now consider the strings $x' = x\texttt{0}^i$ and $y' = y\texttt{0}^i$. Observe that in x' and y', the bits x_i and y_i are the k'th characters from the right. Since x and y end up in the same state, and we append the same string $\texttt{0}^i$ to both to obtain x' and y', we know x' and y' must end up in the same state. However, since $x_i \neq y_i$, one of x' and y' ends up in an accepting state, and the other in a rejecting state. This is the desired contradiction. ■

7 Regular Expressions (Extra Problem)

As mentioned in class, regular languages can be defined recursively/inductively as follows:

- $\varnothing$ is regular.
- $\{a\}$ for each $a \in \Sigma$ is regular.
- If L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular.
- If L_1 and L_2 are regular, then $L_1 L_2$ is regular.
- If L is regular, then L^* is regular.

Often, regular languages are represented compactly using what is known as a *regular expression*. We define regular expressions inductively as follows:

- Any single string is a regular expression (representing the singleton set containing that string).
- If A and B are regular expressions, then $(A + B)$ is a regular expression (representing the union of A and B).
- If A and B are regular expressions, then (AB) is a regular expression (representing the concatenation of A and B).
- If A is a regular expression, then A^* is the regular expression.

Noting the correspondence between the two inductive definitions above, a language is regular if and only if it can be represented by a regular expression.

Here are a few shortcuts related to regular expressions. We use Σ to denote the regular expression corresponding to taking the union of all strings of length 1. So if $\Sigma = \{\texttt{0}, \texttt{1}\}$ then Σ would denote $(\texttt{0} + \texttt{1})$. We also use A^k to denote the regular expression corresponding to taking the concatenation of A with itself k times. Parentheses in regular expressions can be omitted to reduce clutter, provided the removal does not introduce any ambiguity.

For the problems below, you do not need to provide justifications for your answers.

1. Let $\Sigma = \{\texttt{0}, \texttt{1}\}$. For each regular expression below, describe in English the language it represents.
   - $\texttt{1}\Sigma^*$
   - $\Sigma^*\texttt{101}\Sigma^*$
   - $\texttt{1}(\texttt{00})^*\texttt{1}$
   - $(\Sigma^{251})^*$
2. Let $\Sigma = \{\texttt{a}, \texttt{b}\}$. Give a regular expression for the languages below.
   - The set of all strings with a single a.
   - The set of all strings that start and end with the same symbol.
   - The set of all strings that contain at least two a's and at most one b.
   - The set of all strings containing an even number of a's, an odd number of b's, and not containing the substring ab.

Solution.

1.
   - The set of all strings that start with a 1.
   - The set of all strings that contain 101 as a substring.
   - The set of all even-length strings that have exactly two 1's, one at the start of the string and one at the end.
   - The set of all strings whose length is a multiple of 251.
2.
   - $\texttt{b}^*\texttt{ab}^*$
   - $\texttt{a}\Sigma^*\texttt{a} + \texttt{b}\Sigma^*\texttt{b} + \texttt{a} + \texttt{b}$
   - $\texttt{aa}^*\texttt{a} + \texttt{aaa}^*\texttt{ba}^* + \texttt{aa}^*\texttt{ba}^*\texttt{a} + \texttt{a}^*\texttt{ba}^*\texttt{aa}$
   - $\texttt{b}(\texttt{bb})^*(\texttt{aa})^*$

■

8 Multiple Multiples (Bonus)

Let $\Sigma = \{\texttt{0}, \texttt{1}\}$. Define

$$C_3 = \{x \in \Sigma^* : x \text{ is a binary number that is a multiple of } 3\}.$$

Show that C_3 is regular.

Solution. https://www.youtube.com/embed/qZsZdqRm-Wc

Our set of states will be $Q = \{q_0, q_1, q_2\}$ corresponding to the possible remainders modulo 3. We want the following property to hold: a string corresponding to the binary representation of a number w should end on state q_i if $w \equiv i \pmod 3$. If we can establish this, then we can set $F = \{q_0\}$.

As we read a binary string s, we can determine what number/value it corresponds to as follows: Start with value 0. Scan s from left to right. If we see a 0, multiply the value by 2. If we see a 1, multiply the value by 2 and add 1. (Take a moment to verify that this is true.)

With this observation in mind, we'll define our transition function as follows. $\delta(q_i, \texttt{0}) = q_{2i}$, and $\delta(q_i, \texttt{1}) = q_{2i+1}$, where indices of the q's are taken modulo 3.

We leave the following as exercises for you: (i) Why is the above transition function correct? (ii) Does the argument generalize to mod values other than 3? ■