CS 361: Theory of Computation

Assignment 2 (due 09/25/2024)

Instructor: Shikha Singh

LaTeX Source for Solutions: https://www.overleaf.com/read/mphkhbwznhnp#c83197

Problem 1. For any string $w = w_1 w_2 \dots w_n$, the reverse of w, written w^R, is the string w in reverse order, that is, $w^R = w_n \dots w_2 w_1$. For any language L, let $L^R = \{w^R \mid w \in L\}$. Show that regular languages are closed under the reverse operation, that is, show that if L is regular, so is L^R. (*Hint. Use the fact that NFAs and DFAs are equivalent. Let M be the DFA recognizing L, construct an NFA N, using M, that recognizes L^R.)*

Regular Expressions (Sipser Chapter 1.3.) So, far we have used machines to *recognize* languages. We can also use grammars or expressions to *generate* the strings in a language. The first example of this in the course is regular expressions. As we will see, regular expressions are an alternate way to define and study regular languages. They are defined in Definition 1.52 of Sipser, reproduced below.

We say R is a **regular expression** if R is

1. (a symbol) a for some $a \in \Sigma$,
2. (empty string) ε,
3. (empty set) $\emptyset$,
4. (union) $R_1 \cup R_2$, where R_1 and R_2 are (smaller) regular expressions,
5. (concatenation) $R_1 \circ R_2$, where R_1 and R_2 are (smaller) regular expressions, or
6. (Kleene star) R_1^*, where R_1 is a regular expression.

Note that the definition is inductive (with three base cases). Example 1.53 in Sipser describes several regular expressions and the corresponding languages. Review these before answering the following questions.

Based on the definition, it is easy to see that we can create an NFA for any language that is generated by regular expression. This is described in Lemma 1.55 in Sipser. We can argue this inductively. It is possible to design a valid NFA for each of the base cases and since we can construct NFAs for union, concatenation and Kleene star from these base cases, it is possible to construct NFAs for any regular expression. For specific examples, see Example 1.56 and 1.58 in the textbook.

Problem 2. For each of the following regular expressions give

- a description of the language of the regular expression in English,

- two strings that are members of the language and two strings that are NOT members of the language, and
- the state diagram of an NFA that recognizes the same language.

Assume the alphabet $\Sigma = \{a, b\}$.

(a) $a(ba)^*b$

(b) $a^* \cup b^*$

(c) $\Sigma^* a \Sigma^* b \Sigma^* a \Sigma^*$

(d) $(\varepsilon \cup a)b$

Problem 3. Give regular expressions generating the following languages.

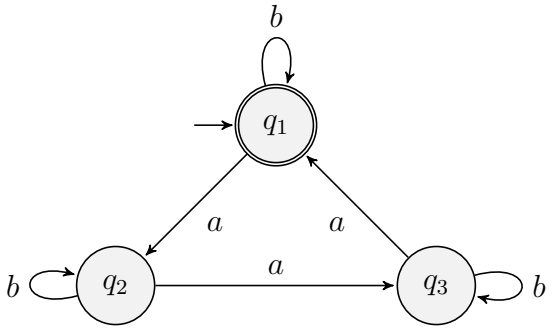
(a) $\{w \mid w$ begins with a 1 and ends with a 0 $\}$

(b) $\{w \mid w$ contains at least three 1s$\}$

(c) $\{w \mid w$ contains the substring 0101$\}$

(d) $\{w \mid w$ has length at least 3 and its third symbol is a 0$\}$

Problem 4. Use the state-elimination algorithm on a generalized non-deterministic finite automata (GNFA), (CONVERT(G), described on Page 73 in Sipser (Proof of Lemma 1.60) to convert the following finite automaton to a regular expression. Show your work as you eliminate each state. Refer to similar examples: Example 1.66 and 1.68 in the textbook (*You may attach a clear hand-drawn image of your work.*)



Problem 5. Let L be any language over the alphabet Σ. Let $\equiv_L$ be the *indistinguishability* relation over Σ^* defined as: for any $x, y \in \Sigma^*$, they are **indistinguishable** with respect to L, denoted $x \equiv_L y$ if for all $z \in \Sigma^*$, $xz \in L$ iff $yz \in L$. A relation if an equivalence relation if it is symmetric, reflexive and transitive. Moreover, an equivalence relation $\sim$ over a set S partitions it into *equivalence classes*, where two elements $x, y \in S$ are in the same equivalence class if only if they are related to each other that is, $x \sim y$. Denote the equivalence class of an element $x \in S$ as $[x]$.

(a) Show that $\equiv_L$ is an equivalence relation on Σ^*, that is, it is symmetric, transitive and reflexive.

(b) Consider the language L described by the regular expression $(0 \cup 1)^*01$. Draw a 3-state DFA that recognizes L.

(c) Identify the three equivalence classes of $\equiv_L$ over Σ^*. Note that $[p] \neq [q]$ if there exists a another string r that distinguishes them, that is, $pr \in L$ but $qr \notin L$ or vice versa. You may use the 3-state DFA for L to identify these classes.