3 Regular expressions

3.1 Definition

Reading: Sipser 1.3 (pp. 63-66)
In addition to DFAs and NFAs, regular expressions (REs) also represent regular languages. Let $L(R)$ be the language that regular expression R represents. A recursive definition for R (and $L(R)$) is given below:

- **Basis**: ε and $\emptyset$ are regular expressions, and $L(\varepsilon) = \{\varepsilon\}$ and $L(\emptyset) = \emptyset$. For any $a \in \Sigma$, a is a regular expression and $L(a) = \{a\}$.
- **Induction**: If R_1 and R_2 are regular expressions, then $R_1 \cup R_2$ is a regular expression, with $L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$, and R_1R_2 is a regular expression, with $L(R_1R_2) = L(R_1)L(R_2)$. If R is a regular expression, then R^* is a regular expression, with $L(R^*) = (L(R))^*$, and (R) is a regular expression, with $L((R)) = L(R)$.

Remark:

- Precedence order for regular-expression operators: Star, concatenation, and finally union.
- Use of R^+ and R^k.
- Algebraic laws:
  - $R_1 \cup R_2 = R_2 \cup R_1$, $(R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3)$, and $(R_1R_2)R_3 = R_1(R_2R_3)$.
  - $\emptyset \cup R = R \cup \emptyset = R$, $\varepsilon R = R\varepsilon = R$, $\emptyset R = R\emptyset = \emptyset$, and $R \cup R = R$.
  - $R_1(R_2 \cup R_3) = R_1R_2 \cup R_1R_3$ and $(R_1 \cup R_2)R_3 = R_1R_3 \cup R_2R_3$.
  - $(R^*)^* = R^*$, $\emptyset^* = \varepsilon$, $R^+ = RR^* = R^*R$, and $R^* = R^+ \cup \varepsilon$.

Examples:
1. A regular expression for the language of strings that consist of alternating 0s and 1s: $(\mathbf{01})^* \cup (\mathbf{10})^* \cup \mathbf{0}(\mathbf{10})^* \cup \mathbf{1}(\mathbf{01})^*$.
2. A regular expression for the language of strings with a 1 in the third position from the end: $(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})(\mathbf{0}+\mathbf{1})$.

3.2 Converting regular expressions to finite automata

Reading: Sipser 1.3 (pp. 67-69)
Since regular expressions are defined recursively, it is suitable to construct the equivalent finite automata recursively.

- **Basis**: The finite automata for regular expressions ε, $\emptyset$, and a for $a \in \Sigma$.
- **Induction**: Given the finite automata for regular expressions R_1 and R_2, what are the finite automata for $R_1 \cup R_2$, R_1R_2, and R_1^*?

Example: Convert regular expression $(\mathbf{0} \cup \mathbf{1})^*\mathbf{1}(\mathbf{0} \cup \mathbf{1})$ to a finite automaton.

3.3 Converting finite automata to regular expressions

Reading: Sipser 1.3 (pp. 69-76)
A generalized NFA (GNFA) is an NFA with regular expressions (not symbols) on its transition arcs.
Assume that the given finite automaton is a DFA $M = (Q, \Sigma, \delta, q_0, F)$. We first convert the DFA to a GNFA by (1) adding a new start state s that goes to the old start state q_0 via an ε-transition, (2) adding a new accept state a to which there is an ε-transition from each old accept state in F, and (3) converting symbols to regular expressions on all arcs. Then this GNFA with $|Q|+2$ states will be converted to an equivalent GNFA with $|Q|+1$ states by eliminating a state that is neither s nor a. This state elimination step will be applied a total of $|Q|$ times until there are only states s and a left in the resulting GNFA. The regular expression on the arc from s to a is the regular expression for the original DFA.
Given a GNFA with k states, how can one convert it to an equivalent GNFA with $k-1$ states by eliminating a state that neither s nor a? (Sipser Figure 1.63 on p.72)
Example (Siper p. 76): A three-state DFA to be converted to a regular expression.
Example: A language of all strings over $\{0,1\}$ with one 1 either two or three positions from the end.
Theorem: The equivalence of RLs, REs, and FAs.