1 Regular Expressions

Regular expressions are a means of expressing regular languages using combinations of symbols and specialized operations, and serve as a shorthand for the verbose set notation given above:

- Concatenation ($ab$): a matching word must contain $a$ followed by $b$ in this position.
- Alternation ($a|b$): a matching word must contain one of $a$ or $b$ in this position.
- Repetition ($a^*$): a matching word may have 0 or more occurrences of $a$ in this position.

Expressions can be grouped into subexpressions using parenthesis. For example, $a(a|b)^*$ matches an $a$ followed by 0 or more $a$’s and $b$’s.

1.1 Exercises

Provide a regular expression corresponding to the following languages:

1. $\Sigma = \{a, b\}, L = \{aa, ab, ba, bb\}$.
2. Strings over the alphabet $\Sigma = \{a, b, +, -, \cdot, /\}$ representing valid arithmetic expressions with no parentheses. All operators should be binary (thus $a + b$ is not valid) and multiplication must be written explicitly (thus $a \cdot b$ is valid but $ab$ is not).
3. (Difficult) $\Sigma = \{0, 1, 2\}, L = \{x \in \Sigma^* | x \text{ contains an even number of 0’s and at least one 1.}\}$

2 NFAs: DFA conversion

Recall the definition of a DFA: $(Q, \Sigma, q_0, A, \delta)$ where:

- $Q$ is a finite set of states.
- $\Sigma$ is the input alphabet.
- $q_0 \in Q$ is the starting state.
- $A \subseteq Q$ (sometimes also denoted by $F$) is the set of accepting states.
- $\delta: Q \times \Sigma \rightarrow Q$ is the transition function.

Furthermore, recall the only difference for an NFA is that $\delta$ outputs a set of states (that is, $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$) instead of a single state. This means that we can reinterpret an NFA as a DFA where the states in the DFA are sets of states in the NFA. This is called the subset construction.
While in theory we could simply draw all $2^{|Q|}$ states in the DFA, this can be very time-consuming and most of the states will not be necessary in practice. Instead we use the following algorithm to convert the DFA $N = (Q', \Sigma, q_0', A', \delta')$ from the NFA $M = (Q, \Sigma, q_0, A, \delta)$:

1. Create the start state $q_0'$ of the DFA and label it $\{q_0\}$, where $q_0$ is the start state of the NFA.
2. Let $S = \{q_0\}$.
3. Add $S$ as a state to $Q'$.
4. For each letter $a \in \Sigma$, let $T$ be the set of states reachable by consuming $a$ when starting at some state in $S$. Add a transition from the state $S$ to the state $T$, adding $T$ to the set of states if necessary. Mathematically:
   $$\delta'(S, a) := \bigcup_{\sigma \in S} \delta(\sigma, a)$$
5. Repeat from step 3 with another state $S$ which has not yet had step 3 run on it, until no such states exist.
6. Label every state which contains an accepting state from $Q$ as accepting. Mathematically:
   $$A' := \{ S \in \mathcal{P}(Q) : S \cap A \neq \emptyset \}$$

### 2.1 Exercises

Convert the following NFA to an DFA using the subset construction algorithm:

![NFA Diagram]

### 3 $\epsilon$-NFAs

An $\epsilon$-Nondeterministic Finite Automaton ($\epsilon$-NFA) is a 5-tuple $(Q, \Sigma, q_0, A, \delta)$ where:

- $Q$ is a finite set of states.
- $\Sigma$ is the input alphabet.
- $q_0 \in Q$ is the starting state.
- $A \subseteq Q$ (sometimes also denoted by $F$) is the set of accepting states.
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$ is the transition function.
Notice that this definition is very similar to that of regular NFAs: the only difference is that the transition function now accepts $\epsilon$ as a special symbol. This represents a transition which can be taken without consuming any input.

$\epsilon$-NFAs don’t tend to be very natural to write directly, but are a very easy way to combine multiple DFAs or NFAs. For example, Thompson’s Construction is an algorithm that creates an $\epsilon$-NFA from a regular expression.

### 3.1 $\epsilon$-NFAs to NFAs: The $\epsilon$ closure

Similar to converting NFAs to DFAs, we can also convert $\epsilon$-NFAs to NFAs using the $\epsilon$ closure algorithm to convert the NFA $N = (Q', \Sigma, q_0', A', \delta')$ from the $\epsilon$-NFA $M = (Q, \Sigma, q_0, A, \delta)$:

1. Draw the start state of the NFA using the same start state as the $\epsilon$-NFA.
2. Let $Q'$ be the start state $q_0'$ as well as any other state $q \in Q$ which has at least one non-$\epsilon$ transition entering it.
3. For each state in $q \in Q'$ and every letter $a \in \Sigma$, add a transition to all states which can be reached from $q$ by consuming exactly one $a$.
4. Finally, mark each state $q \in Q'$ as accepting if it can reach an accepting state in the original $\epsilon$-NFA by only following $\epsilon$ transitions.

### 3.2 Exercises

Convert the following $\epsilon$-NFA to an NFA using the $\epsilon$ closure algorithm: