Regular Epressions

Regular expressions are a convenient notation for regular languages.

A regular expression over alphabet Σ is defined by the following syntax:

$$
\begin{array}{rll}
r ::= & \mathtt{1} & \\
& \mathtt{0} & \\
& a & \text{for every } a \in \Sigma \\
& (r_1 \mathtt{+} r_2) & \\
& (r_1 r_2) & \\
& (r_1^*) &
\end{array}
$$

We usually drop parentheses, under the assumption that r_1^* binds tighter than concatenation $r_1 r_2$ which binds tighter than $r_1 \mathtt{+} r_2$. For example, $\mathtt{ab+ac}$ is a regular expression, as is $\mathtt{a(b+c)}$ and $\mathtt{a}^*\mathtt{(b+c)}^*$.

A regular expression r denotes a language $[\![r]\!]$ over Σ in the following way:

$$
\begin{aligned}
[\![\mathtt{1}]\!] &= \{\epsilon\} \\
[\![\mathtt{0}]\!] &= \varnothing \\
[\![a]\!] &= \{a\} \\
[\![r_1 \mathtt{+} r_2]\!] &= [\![r_1]\!] \cup [\![r_2]\!] \\
[\![r_1 r_2]\!] &= [\![r_1]\!] \cdot [\![r_2]\!] \\
[\![r_1^*]\!] &= [\![r_1]\!]^*
\end{aligned}
$$

For example:

$$
\begin{aligned}
[\![\mathtt{ab+ac}]\!] &= [\![\mathtt{ab}]\!] \cup [\![\mathtt{ac}]\!] \\
&= ([\![\mathtt{a}]\!] \cdot [\![\mathtt{b}]\!]) \cup ([\![\mathtt{a}]\!] \cdot [\![\mathtt{c}]\!]) \\
&= (\{\mathtt{a}\} \cdot \{\mathtt{b}\}) \cup (\{\mathtt{a}\} \cdot \{\mathtt{c}\}) \\
&= \{\mathtt{ab}\} \cup \{\mathtt{ac}\} \\
&= \{\mathtt{ab}, \mathtt{ac}\}
\end{aligned}
$$

$$
\begin{aligned}
[\![\mathtt{a(b+c)}]\!] &= [\![\mathtt{a}]\!] \cdot [\![\mathtt{b+c}]\!] \\
&= [\![\mathtt{a}]\!] \cdot ([\![\mathtt{b}]\!] \cup [\![\mathtt{c}]\!])
\end{aligned}
$$

$$
\begin{aligned}
&= \{\mathtt{a}\} \cdot (\{\mathtt{b}\} \cup \{\mathtt{c}\}) \\
&= \{\mathtt{a}\} \cdot (\{\mathtt{b}, \mathtt{c}\}) \\
&= \{\mathtt{ab}, \mathtt{ac}\}
\end{aligned}
$$

$$
\begin{aligned}
[\![\mathtt{a}^*(\mathtt{b}+\mathtt{c})^*]\!] &= [\![\mathtt{a}^*]\!] \cdot [\![(\mathtt{b}+\mathtt{c})^*]\!] \\
&= [\![\mathtt{a}]\!]^* \cdot [\![\mathtt{b}+\mathtt{c}]\!]^* \\
&= \{\mathtt{a}\}^* \cdot ([\![\mathtt{b}]\!] \cup [\![\mathtt{c}]\!])^* \\
&= \{\mathtt{a}\}^* \cdot (\{\mathtt{b}\} \cup \{\mathtt{c}\})^* \\
&= \{\mathtt{a}\}^* \cdot \{\mathtt{b}, \mathtt{c}\}^*
\end{aligned}
$$

And thinking about this last set (which is difficult to write down), it is basically the set of all strings obtained by concatenating a sequence of as (including none) to a sequence of bs and cs (in any order, including none). So aaaaaa is in this set, as is aaaab, aaaabbbb, aaaabbbcbcbc, etc.

Theorem: A language A is regular exactly if there is a regular expression r such that $[\![r]\!] = A$.

To show that the language of a regular expression is regular, we need to construct for every regular expression a corresponding DFA that accepts the language of that regular expression. We'll do that next. To show that every regular language is the language of a regular expression, we need to construct a regular expression from a DFA witnessing the regularity of the language. That's harder, and I'll point you to Chen for that.

Nondeterministic Finite Automata

A nondeterministic finite automata is just a finite automata like we saw last time but **without** the deterministic restriction on the transition relation: there is no requirement that there be exactly one transition out of every state labeled with each of the symbols of the alphabet.

To be complete: a *(nondeterministic) finite automaton* is a structure

$$M = (Q, \Sigma, \Delta, s, F)$$

where

- Q is a finite set of states
- Σ is a finite alphabet
- $\Delta \subseteq Q \times \Sigma \times Q$ is a transition relation: $\langle p, a, q \rangle \in \Delta$ when there is a transition from state p to state q labeled by the symbol a
- $s \in Q$ is the start state
- $F \subseteq Q$ is a set of final states.

Finite automaton M *accepts* string $u = a_1 \dots a_k$ if there is a path in M starting with s labeled by $a_1, a_2, \dots, a_k$, and ending up in a final state.

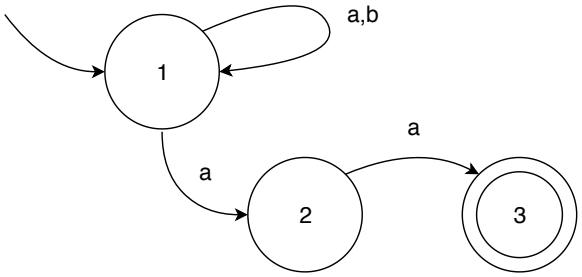
Formally: $M = (Q, \Sigma, \Delta, s, F)$ accepts $u = a_1 \dots a_k$ if there exists $q_0, q_1, \dots, q_k \in Q$ such that $q_0 = s$, $q_k \in F$, and $\langle q_{i-1}, a_i, q_i \rangle \in \Delta$ for all $1 \leq i \leq k$.

The language accepted by M is

$$L(M) = \{u \mid M \text{ accepts } u\}$$

In order to check if a finite automaton accepts a string, now we basically have to "search" through all paths in the automata, and see if one of them gets to a final state. If one does, we accept the string; if no path does, then we reject.

Example: The following nondeterministic finite automaton accepts exactly the strings over $\{\mathtt{a}, \mathtt{b}\}$ whose last two symbols are $\mathtt{a}$:



$$M_{last} = (\{1, 2, 3\}, \{\mathtt{a}, \mathtt{b}\}, \Delta_{last}, 1, \{3\})$$

where

$$\Delta_{last} = \{\langle 1, \mathtt{a}, 1\rangle, \langle 1, \mathtt{b}, 1\rangle, \langle 1, \mathtt{a}, 2\rangle, \langle 2, \mathtt{a}, 3\rangle\}$$

Subset Construction

Even though DFAs seem more restrictive than nondeterministic finite automata, it turns out that DFAs accept exactly the regular languages. Clearly, since every DFA is a finite automaton, if a DFA accepts a language A then A is regular. But if A is regular, while we know it can be accepted by a finite automaton, it's not clear it can be accepted by a *deterministic* finite automaton.

Consider the following way to "execute" a nondeterministic finite automaton M. The idea is to look at the possible transitions that can be taken for a given symbol of the input all at once. Intuitively, we begin in the start state, and for every symbol of the input string in order, we keep track of the possible states we can reach by following a transition labeled with that symbol from any of the possible states we have already reached. Thus, at every step in this process, we maintain a set of possible states that M can be in. If at the end of the string one of the possible states is final, then M accepts the string.

This process, where we consider *sets* of states of M, in fact describes the execution of a deterministic finite automaton, in which states are sets of states of M, and there is a transtion from a set X of states of M to a set Y of states of M labeled a exactly when the transitions of M labeled a from any state in X lead to the states in Y.

Formally, for every finite automaton M, there exists a deterministic finite automaton $\widehat{M}$ that accepts the same language as M, constructed as follows. If $M = (Q, \Sigma, \Delta, s, F)$, construct

$$\widehat{M} = (\widehat{Q}, \Sigma, \widehat{\Delta}, \widehat{s}, \widehat{F})$$

where:

- $\widehat{Q} = \{X \mid X \subseteq Q\}$
- $\widehat{\Delta} = \{\langle X, a, Y\rangle \mid X \subseteq Q, Y = \{q \in Q \mid \langle p, a, q\rangle \in \Delta \text{ for some } p \in X\}\}$
- $\widehat{s} = \{s\}$
- $\widehat{F} = \{X \subseteq Q \mid X \cap F \neq \varnothing\}$

This construction is called the *subset construction*.

We can see that $\widehat{M}$ is deterministic because any subset $X \subseteq Q$ uniquely determines a subset Y to transition to in the definition of $\widehat{\Delta}$.