2.5 REGULAR EXPRESSIONS

The languages accepted by finite automata are easily described by simple expressions called regular expressions. In this section we introduce the operations of concatenation and closure on sets of strings, define regular expressions, and prove that the class of languages accepted by finite automata is precisely the class of languages describable by regular expressions.

Let Σ be a finite set of symbols and let L, L_1, and L_2 be sets of strings from Σ^*. The *concatenation* of L_1 and L_2, denoted L_1L_2, is the set $\{xy \mid x \text{ is in } L_1 \text{ and } y \text{ is in } L_2\}$. That is, the strings in L_1L_2 are formed by choosing a string L_1 and following it by a string in L_2, in all possible combinations. Define $L^0 = \{\epsilon\}$ and $L^i = LL^{i-1}$ for $i \geq 1$. The *Kleene closure* (or just *closure*) of L, denoted L^*, is the set

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

and the *positive closure* of L, denoted L^+, is the set

$$L^+ = \bigcup_{i=1}^{\infty} L^i.$$

That is, L^* denotes words constructed by concatenating any number of words from L. L^+ is the same, but the case of zero words, whose "concatenation" is defined to be ϵ, is excluded. Note that L^+ contains ϵ if and only if L does.

Example 2.10 Let $L_1 = \{10, 1\}$ and $L_2 = \{011, 11\}$. Then $L_1L_2 = \{10011, 1011, 111\}$. Also,

$$\{10, 11\}^* = \{\epsilon, 10, 11, 1010, 1011, 1110, 1111, \ldots\}.$$

If Σ is an alphabet, then Σ^* denotes all strings of symbols in Σ, as we have previously stated. Note that we are not distinguishing Σ as an alphabet from Σ as a language of strings of length 1.

Let Σ be an alphabet. The regular expressions over Σ and the sets that they denote are defined recursively as follows.

1) $\varnothing$ is a regular expression and denotes the empty set.
2) ϵ is a regular expression and denotes the set $\{\epsilon\}$.
3) For each a in Σ, **a**† is a regular expression and denotes the set $\{a\}$.
4) If r and s are regular expressions denoting the languages R and S, respectively, then $(r + s)$, (rs), and (r^*) are regular expressions that denote the sets $R \cup S$, RS, and R^*, respectively.

† To remind the reader when a symbol is part of a regular expression, we shall write it in boldface. However, we view **a** and a as the same symbol.

In writing regular expressions we can omit many parentheses if we assume that * has higher precedence than concatenation or +, and that concatenation has higher precedence than +. For example, $((\mathbf{0}(\mathbf{1}^*)) + \mathbf{0})$ may be written $\mathbf{0}\mathbf{1}^* + \mathbf{0}$. We may also abbreviate the expression rr^* by r^+. When necessary to distinguish between a regular expression r and the language denoted by r, we use $L(r)$ for the latter. When no confusion is possible we use r for both the regular expression and the language denoted by the regular expression.

Example 2.11 **00** is a regular expression representing {00}. The expression $(\mathbf{0} + \mathbf{1})^*$ denotes all strings of 0's and 1's. Thus, $(\mathbf{0} + \mathbf{1})^*\mathbf{00}(\mathbf{0} + \mathbf{1})^*$ denotes all strings of 0's and 1's with at least two consecutive 0's. The regular expression $(\mathbf{1} + \mathbf{10})^*$ denotes all strings of 0's and 1's beginning with 1 and not having two consecutive 0's. In proof, it is an easy induction on i that $(\mathbf{1} + \mathbf{10})^i$ does not have two consecutive 0's.† Furthermore, given any string beginning with 1 and not having consecutive 0's, one can partition the string into 1's, with a following 0 if there is one. For example, 1101011 is partitioned 1–10–10–1–1. This partition shows that any such string is in $(\mathbf{1} + \mathbf{10})^i$, where i is the number of 1's. The regular expression $(\mathbf{0} + \epsilon)(\mathbf{1} + \mathbf{10})^*$ denotes all strings of 0's and 1's whatsoever that do not have two consecutive 0's.

For some additional examples, $(\mathbf{0} + \mathbf{1})^*\mathbf{011}$ denotes all strings of 0's and 1's ending in 011. Also, $\mathbf{0}^*\mathbf{1}^*\mathbf{2}^*$ denotes any number of 0's followed by any number of 1's followed by any number of 2's. This is the language of the NFA of Fig. 2.8. $\mathbf{00}^*\mathbf{11}^*\mathbf{22}^*$ denotes those strings in $\mathbf{0}^*\mathbf{1}^*\mathbf{2}^*$ with at least one of each symbol. We may use the shorthand $\mathbf{0}^+\mathbf{1}^+\mathbf{2}^+$ for $\mathbf{00}^*\mathbf{11}^*\mathbf{22}^*$.

Equivalence of finite automata and regular expressions

We now turn to showing that the languages accepted by finite automata are precisely the languages denoted by regular expressions. This equivalence was the motivation for calling finite automaton languages regular sets. Our plan will be to show by induction on the *size* of (number of operators in) a regular expression that there is an NFA with ϵ-transitions denoting the same language. Finally, we show that for every DFA there is a regular expression denoting its language. These constructions, together with Theorems 2.1 and 2.2, show that all four language defining mechanisms discussed in this chapter define the same class of languages, the regular sets. Figure 2.12 shows the constructions we shall perform or have performed, where an arrow from A to B means that for any descriptor of type A a construction yields an equivalent descriptor of type B.

We proceed to prove that for every regular expression there is an equivalent NFA with ϵ-transitions.

† If r is a regular expression, r^i stands for $rr \cdots r$ (i times).