4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\varepsilon$,

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \text{ and } a = \varepsilon \\ \{q_1\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon. \end{cases}$$

1.3 REGULAR EXPRESSIONS

In arithmetic, we can use the operations $+$ and $\times$ to build up expressions such as

$$(5+3) \times 4\,.$$

Similarly, we can use the regular operations to build up expressions describing languages, which are called ***regular expressions***. An example is:

$$(\texttt{0} \cup \texttt{1})\texttt{0}^*.$$

The value of the arithmetic expression is the number 32. The value of a regular expression is a language. In this case, the value is the language consisting of all strings starting with a 0 or a 1 followed by any number of 0s. We get this result by dissecting the expression into its parts. First, the symbols 0 and 1 are shorthand for the sets $\{\texttt{0}\}$ and $\{\texttt{1}\}$. So $(\texttt{0} \cup \texttt{1})$ means $(\{\texttt{0}\} \cup \{\texttt{1}\})$. The value of this part is the language $\{\texttt{0,1}\}$. The part $\texttt{0}^*$ means $\{\texttt{0}\}^*$, and its value is the language consisting of all strings containing any number of 0s. Second, like the $\times$ symbol in algebra, the concatenation symbol $\circ$ often is implicit in regular expressions. Thus $(\texttt{0} \cup \texttt{1})\texttt{0}^*$ actually is shorthand for $(\texttt{0} \cup \texttt{1}) \circ \texttt{0}^*$. The concatenation attaches the strings from the two parts to obtain the value of the entire expression.

Regular expressions have an important role in computer science applications. In applications involving text, users may want to search for strings that satisfy certain patterns. Regular expressions provide a powerful method for describing such patterns. Utilities such as awk and grep in UNIX, modern programming languages such as Perl, and text editors all provide mechanisms for the description of patterns by using regular expressions.

EXAMPLE 1.51

Another example of a regular expression is

$$(\mathtt{0} \cup \mathtt{1})^*.$$

It starts with the language $(\mathtt{0} \cup \mathtt{1})$ and applies the $*$ operation. The value of this expression is the language consisting of all possible strings of 0s and 1s. If $\Sigma = \{\mathtt{0},\mathtt{1}\}$, we can write Σ as shorthand for the regular expression $(\mathtt{0} \cup \mathtt{1})$. More generally, if Σ is any alphabet, the regular expression Σ describes the language consisting of all strings of length 1 over this alphabet, and Σ^* describes the language consisting of all strings over that alphabet. Similarly, $\Sigma^*\mathtt{1}$ is the language that contains all strings that end in a 1. The language $(\mathtt{0}\Sigma^*) \cup (\Sigma^*\mathtt{1})$ consists of all strings that start with a 0 or end with a 1. ■

In arithmetic, we say that $\times$ has precedence over $+$ to mean that when there is a choice, we do the $\times$ operation first. Thus in $2+3\times 4$, the 3×4 is done before the addition. To have the addition done first, we must add parentheses to obtain $(2+3)\times 4$. In regular expressions, the star operation is done first, followed by concatenation, and finally union, unless parentheses change the usual order.

FORMAL DEFINITION OF A REGULAR EXPRESSION

DEFINITION 1.52

Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,
5. $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, or
6. (R_1^*), where R_1 is a regular expression.

In items 1 and 2, the regular expressions a and ε represent the languages $\{a\}$ and $\{\varepsilon\}$, respectively. In item 3, the regular expression $\emptyset$ represents the empty language. In items 4, 5, and 6, the expressions represent the languages obtained by taking the union or concatenation of the languages R_1 and R_2, or the star of the language R_1, respectively.

Don't confuse the regular expressions ε and $\emptyset$. The expression ε represents the language containing a single string—namely, the empty string—whereas $\emptyset$ represents the language that doesn't contain any strings.

Seemingly, we are in danger of defining the notion of a regular expression in terms of itself. If true, we would have a ***circular definition***, which would be invalid. However, R_1 and R_2 always are smaller than R. Thus we actually are defining regular expressions in terms of smaller regular expressions and thereby avoiding circularity. A definition of this type is called an ***inductive definition***.

Parentheses in an expression may be omitted. If they are, evaluation is done in the precedence order: star, then concatenation, then union.

For convenience, we let R^+ be shorthand for RR^*. In other words, whereas R^* has all strings that are 0 or more concatenations of strings from R, the language R^+ has all strings that are 1 or more concatenations of strings from R. So $R^+ \cup \varepsilon = R^*$. In addition, we let R^k be shorthand for the concatenation of k R's with each other.

When we want to distinguish between a regular expression R and the language that it describes, we write $L(R)$ to be the language of R.

EXAMPLE 1.53

In the following instances, we assume that the alphabet Σ is $\{0,1\}$.

1. $0^*10^* = \{w|\ w \text{ contains a single } 1\}$.
2. $\Sigma^*1\Sigma^* = \{w|\ w \text{ has at least one } 1\}$.
3. $\Sigma^*001\Sigma^* = \{w|\ w \text{ contains the string } 001 \text{ as a substring}\}$.
4. $1^*(01^+)^* = \{w|\text{ every } 0 \text{ in } w \text{ is followed by at least one } 1\}$.
5. $(\Sigma\Sigma)^* = \{w|\ w \text{ is a string of even length}\}$.[5]
6. $(\Sigma\Sigma\Sigma)^* = \{w|\text{ the length of } w \text{ is a multiple of } 3\}$.
7. $01 \cup 10 = \{01, 10\}$.
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w|\ w \text{ starts and ends with the same symbol}\}$.
9. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$.
   The expression $0 \cup \varepsilon$ describes the language $\{0, \varepsilon\}$, so the concatenation operation adds either 0 or ε before every string in 1^*.
10. $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$.
11. $1^*\emptyset = \emptyset$.
    Concatenating the empty set to any set yields the empty set.
12. $\emptyset^* = \{\varepsilon\}$.
    The star operation puts together any number of strings from the language to get a string in the result. If the language is empty, the star operation can put together 0 strings, giving only the empty string.

■

[5]The ***length*** of a string is the number of symbols that it contains.

If we let R be any regular expression, we have the following identities. They are good tests of whether you understand the definition.

$R \cup \emptyset = R$.
Adding the empty language to any other language will not change it.

$R \circ \varepsilon = R$.
Joining the empty string to any string will not change it.

However, exchanging $\emptyset$ and ε in the preceding identities may cause the equalities to fail.

$R \cup \varepsilon$ may not equal R.
For example, if $R = \texttt{0}$, then $L(R) = \{\texttt{0}\}$ but $L(R \cup \varepsilon) = \{\texttt{0}, \varepsilon\}$.

$R \circ \emptyset$ may not equal R.
For example, if $R = \texttt{0}$, then $L(R) = \{\texttt{0}\}$ but $L(R \circ \emptyset) = \emptyset$.

Regular expressions are useful tools in the design of compilers for programming languages. Elemental objects in a programming language, called ***tokens***, such as the variable names and constants, may be described with regular expressions. For example, a numerical constant that may include a fractional part and/or a sign may be described as a member of the language

$$\big(\texttt{+} \cup \texttt{-} \cup \varepsilon\big)\,\big(D^{\texttt{+}} \cup D^{\texttt{+}}\,\texttt{.}\,D^{*} \cup D^{*}\,\texttt{.}\,D^{\texttt{+}}\big)$$

where $D = \{\texttt{0},\texttt{1},\texttt{2},\texttt{3},\texttt{4},\texttt{5},\texttt{6},\texttt{7},\texttt{8},\texttt{9}\}$ is the alphabet of decimal digits. Examples of generated strings are: `72`, `3.14159`, `+7.`, and `-.01`.

Once the syntax of a programming language has been described with a regular expression in terms of its tokens, automatic systems can generate the ***lexical analyzer***, the part of a compiler that initially processes the input program.

EQUIVALENCE WITH FINITE AUTOMATA

Regular expressions and finite automata are equivalent in their descriptive power. This fact is surprising because finite automata and regular expressions superficially appear to be rather different. However, any regular expression can be converted into a finite automaton that recognizes the language it describes, and vice versa. Recall that a regular language is one that is recognized by some finite automaton.

THEOREM 1.54

A language is regular if and only if some regular expression describes it.

This theorem has two directions. We state and prove each direction as a separate lemma.