7 Regular Expressions

This section introduces regular expressions. A regular expression describes a set of strings. We will see that the class of languages that can be decribed by regular expressions is the same as the class of regular languages (those languages that can be solved by a finite-state machine).

Regular expressions are of practical value; some programming language libraries provide tools for doing searches based on regular expressions. Some text editors and console command languages similarly provide tools for searching for something that is a member of the set described by a given regular expression.

7.1 Regular Operations

The *regular operations* are operations on languages; they take one or more languages and yield another language, in the same sense that operator + takes two numbers and yields another number. The first regular operation is union ($A \cup B$), which we have already seen. The remaining two regular operations are concatenation and Kleene closure.

Definition 7.1. The *concatenation* $A \cdot B$ of languages A and B is defined by

$$A \cdot B = \{xy \mid x \in A \text{ and } y \in B\}.$$

That is, $A \cdot B$ is the set of all strings that can be formed by writing a member of A followed by a member of B. For example, $\{"aa", "ccb"\} \cdot \{"abc", "bb"\} = \{"aaabc", "aabb", "ccbabc", "ccbbb"\}$.

Definition 7.2. The *Kleene closure* A^* of language A is defined by

$$A^* = \{x_1 x_2 \cdots x_n \mid n \geq 0 \text{ and } x_i \in A \text{ for } i = 1, \ldots, n\}.$$

If $A = \{"a", "bcb"\}$ then $A^* = \{\varepsilon, "a", "bcb", "aa", "abcb", "bcba", "bcbbcb", \ldots\}$. A^* contains the empty string and all strings that can be formed by concatenating members of A together. Notice that $\{\}^* = \{\varepsilon\}$.

Language L is *closed under concatenation* if, whenever x and y are both in L, xy is also in L. Another way to define the Kleene closure of A is as the smallest set of strings that is closed under concatenation and that contains ε all members of A.

7.2 Regular Expressions

A regular expression e over alphabet Σ is an expression whose value is a language $L(e)$ over Σ. Regular expressions have the following forms.

1. Symbol $\emptyset$ is a regular expression. $L(\emptyset) = \{\}$.
2. A symbol $a \in \Sigma$ is a regular expression. $L(a) = \{"a"\}$.
3. If A and B are regular expressions, then:
   (a) $A \cup B$ is a regular expression. $L(A \cup B) = L(A) \cup L(B)$.
   (b) AB is a regular expression. $L(AB) = L(A) \cdot L(B)$.
   (c) A^* is a regular expression. $L(A^*) = L(A)^*$.

Conventionally, $*$ has highest precedence, followed by concatenation, with $\cup$ having lowest precedence. You can use parentheses to override precedence rules.

We put spaces in some regular expressions to make them more readable. They don't affect the meaning.

7.3 Examples of Regular Expressions

$(ab)^*$	Any string over alphabet $\{a, b\}$ that consists of ab repeated zero or more times. $\{\varepsilon$, "ab", "$abab$", "$ababab$", $\ldots\}$
a^*b^*	Any string over alphabet $\{a, b\}$ that consists of zero or more a's followed by zero or more b's: $\{\varepsilon$, "a", "b", "ab", "aab", "$aabb$", $\ldots\}$.
$(a \cup b)^*$	All strings over alphabet $\{a, b\}$.
$(a \cup b)^*a(a \cup b)$	All strings over alphabet $\{a, b\}$ whose next-to-last symbol is a.
$(a \cup b)^*aabb(a \cup b)^*$	All strings over alphabet $\{a, b\}$ that have $aabb$ as a contiguous substring.
$b^*(ab^*a)^*b^*$	All strings over alphabet $\{a, b\}$ that have an even number of as.
$(0 \cup 1(01^*0)^*1)^*$	All binary numbers that are divisible by 3. (This one is difficult and is not obvious. Look at the DFA in Figure 5.1.1. Starting in state 0, what can the DFA read to get it back to state 0? Certainly, it can read a 0. It can also read a 1, taking it to state 1, then 01^*0 repeated any number of times, then a 1 to get it back to state 0. Those two, getting the DFA from state 0 back to state 0, can be repeated any number of times.

prev **next**