6 Regular Expressions

This section introduces regular expressions. A regular expression describes a set of strings. The class of languages that can be decribed by regular expressions is exactly the class of regular languages, which Section 5defines to be the class of languages that can be solved by finite-state machines.

6.1 Regular operations

The regular operations are operations on languages. 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 6.1. 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, $\{\texttt{"}aa\texttt{"}, \texttt{"}ccb\texttt{"}\} \cdot \{\texttt{"}abc\texttt{"}, \texttt{"}bb\texttt{"}\} = \{\texttt{"}aaabc\texttt{"}, \texttt{"}aabb\texttt{"}, \texttt{"}ccbabc\texttt{"}, \texttt{"}ccbbb\texttt{"}\}$.

Definition 6.2. 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 $A = \{\texttt{"}a\texttt{"}, \texttt{"}bcb\texttt{"}\}$ then $A^* = \{\varepsilon, \texttt{"}a\texttt{"}, \texttt{"}bcb\texttt{"}, \texttt{"}aa\texttt{"}, \texttt{"}abcb\texttt{"}, \texttt{"}bcba\texttt{"}, \texttt{"}bcbbcb\texttt{"}, \ldots\}$. 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 and all members of A.

6.2 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. A symbol $a \in \Sigma$ is a regular expression. $L(a) = \{"a"\}$.
2. 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.

6.3 Regular expressions and regular languages

We do not have time to prove the following two theorems.

Theorem 6.1. If e is a regular expression then $L(e)$ is a regular language.

Theorem 6.1. If A is a regular language then there exists a regular expression e so that $L(e) = A$.

We have two very different ways to describe the class of regular languages: as languages that are decidable by FSMs and as languages that can be described by regular expressions.

Just because two things are defined differently does not necessarily make them different things.

6.4 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 alphabed $\{a, b\}$ that consists of zero or more as followed by zero or more bs. $\{\varepsilon$, "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 character 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 FSM in Figure 5-5. Starting in state 0, what can the FSM 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 FSM from state 0 back to state 0, can be repeated any number of times.

Exercises

Why can't you write a regular expression e so that $L(e)$ is the set of all strings over $\{a, b\}$ that have the same number of a's as bs?

prev **next**