String Notations

For a set C of characters, the notation C^* denotes the set of all finite strings over C. Each $x\square C^*$ has a **length**, $len(x) \geq 0$, the number of characters in string x. The **null string** which has length 0 is included in C^* and is written as $\square$.

For strings $u,v\square C^*$, their **concatenation** is $uv\square C^*$ (u followed by v) and len(uv) = len(u) + len(v). Also for $w\square C^*$ and $n \geq 0$ an integer, $w^n = ww \ldots w$ (n copies) is the n-fold concatenation of w with itself, and $w^0 = \square$. Note that $w^n w^m = w^{n+m}$ for all $m,n \geq 0$.

A **language** is just a subset of C^*. Since a language L is a set, we may speak of its cardinality (number of elements), card(L). For sets of strings $S,T\square C^*$ we perform the usual set-theoretic operations of union, intersection and complementation. We also perform **set concatenation** $S \bullet T$ to get a new set $S \bullet T = \{ st \mid s\square S \text{ and } t\square T\}$. We can observe that $card(S \bullet T) \leq card(S) * card(T)$. We also use the notation S^n to denote the set of strings $S \bullet S \bullet \ldots \bullet S$ (n copies), where $S^0 = \{\square\}$. And **set iteration** or **star** is defined as an arbitrary number of iterations, $S^* = S^0 \square S^1 \square \ldots \square S^n \square \ldots$ The laws of exponents are valid for the power notation for set concatenation as well as string concatenation.

Regular Expressions

The set operations $\square$, $\bullet$, and * are called the *regular expression* operations. A regular expression is a prototypical description of a language. A **regular expression** over a character set C is a formula (or pattern) involving characters from C plus several auxiliary symbols, constructed according to the following rules:

(1) each $\square\square C$ is a regular expression, and auxiliary symbols $\square$ and $\varnothing$ are regular expressions;

(2) using additional auxiliary symbols | (or), • (concatenation), * (star), and parenthesis, if A and B are regular expressions, then so are

- (a) $(A \mid B)$,
- (b) $(A \bullet B)$, and
- (c) (A^*);

(3) only formulas constructed by repeated application of rules (1) and (2) are regular expressions.

The formal rules for writing regular expressions as given above require a fully parenthesized form. To provide a more practical format, the regular expression operations are given precedence so that parenthesis can often be omitted: * is highest, • is intermediate, and | is lowest; also, in place of $A \bullet B$ we normally write AB. Each regular expression A denotes a language $L(A)\square C^*$, referred to as a *regular language*, as defined by:

$L(\square) = \{\square\}$ for $\square\square C$,
$L(\square) = \{\square\}$,
$L(\varnothing) = \varnothing$,
if $A = B \mid C$, then $L(A) = L(B) \square L(C)$,
if $A = B \bullet C$, then $L(A) = L(B) \bullet L(C)$,
if $A = B^*$, then $L(A) = (L(B))^*$.

Examples

For all these examples we take the character set C = {0,1}. Note that a regular expression written in precedence-oriented shorthand such as (00)* 1* in the fully parenthesized form of the formal definition would be written as (((0•0)*)•(1*)). We use precedence conventions in these examples:

- 001 | 010 | 100 denotes the language with three strings {001, 010, 100}
- (0 | 1)* denotes the (infinite) language consisting of *all* strings, {□, 0, 1, 00, 01, … }
- 0(0 | 1)* denotes the (infinite) language consisting of *all* strings beginning with '0'
- (0 | 1)*1 denotes the (infinite) language consisting of *all* strings ending with '1'
- 0*(10*10*)* denotes the (infinite) language consisting of *all* strings having an even number of '1's
- 0 | 1 | 0(0 | 1)*0 | 1(0 | 1)*1 denotes the (infinite) language consisting of *all* strings with the same first and last character