String Notations

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

For strings u,v∈C*, their concatenation is uv∈C* (u followed by v) and len(uv) = len(u) + len(v). Also for w∈C* and n ≥ 0 an integer, wn = w...w (n copies) is the n-fold concatenation of w with itself, and w0 = □. Note that wnwm = wn+m for all m,n ≥ 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∈C* we perform the usual set-theoretic operations of union, intersection and complementation. We also perform set concatenation S•T to get a new set S•T = { s•t | s∈S and t∈T }. We can observe that card(S•T) ≤ card(S) card(T). We also use the notation Sn to denote the set of strings S•S•...•S (n copies), where S0 = {□}. And set iteration or star is defined as an arbitrary number of iterations, S* = S0 □ S1 □ ... □ Sn □ ... The laws of exponents are valid for the power notation for set concatenation as well as string concatenation.

Regular Expressions

The set operations [], •, 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 [□]C is a regular expression, and auxiliary symbols [] and ∅ 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 ∕ B),
(b) (A • 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 • B we normally write AB. Each regular expression A denotes a language L(A)∈C*, referred to as a regular language, as defined by:
L(□) = {□} for □∈C,
L([□]) = {□},
L(∅) = ∅,
if A = B ∕ C, then L(A) = L(B) ∕ L(C),
if A = B•C, then L(A) = L(B) • 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 \mid 010 \mid 100 denotes the language with three strings \{001, 010, 100\}
• (0 \mid 1)* denotes the (infinite) language consisting of all strings, \{\emptyset, 0, 1, 00, 01, \ldots\}
• 0(0 \mid 1)* denotes the (infinite) language consisting of all strings beginning with '0'
• (0 \mid 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 \mid 1 \mid (0 \mid 1)*0 \mid 1(0 \mid 1)*1 denotes the (infinite) language consisting of all strings with the same first and last character