Strings and Languages[1]

1. The term *alphabet* or *character class* denotes any finite set of symbols. Typical examples of symbols are letters and characters. The set **{0,1}** is the *binary* alphabet. ASCII and EBCDIC are two examples of computer alphabets.
2. **String**: A *string* over some alphabet is a finite sequence of symbols drawn from that alphabet.
3. **Length**: The *length* of a string s, usually written $|s|$, is the number of occurrences of symbols in s. For example, `banana` is a string of length six. The empty string, denoted Î, is a special string of length zero.
4. **Language**: The term *language* denotes any set of strings over some fixed alphabet. This definition is very broad. Abstract languages like {}, the *empty* set, or { Î }, the set containing only the empty string, are languages under this definition. So too are the set of all syntactically well-formed Pascal programs and the set of all grammatically correct English sentences, although the latter two sets are much more difficult to specify. Also note that this definition does not ascribe any meaning to the strings in a language.
5. **Concatenation**: If *x* and *y* are strings, then the *concatenation* of *x* and *y*, written *xy*, is the string formed by appending *y* to *x*. For example, if *x* = `dog` and *y* = `house`, then *xy* = `doghouse`. The empty string is the identity element under concatenation. That is, *s* Î = Î *s* = *s*.
6. **Exponentiation**: If we think of concatenation as a "product", we can define string "exponentiation" as follows. Define s^0 to Î, and for $i > 0$ define s^i to be $s^{i-1}s$. Since Î *s* is s itself, $s^1 = s$. The, $s^2 = ss$, $s^3 = sss$, and so on.

Term	Definition
prefix of *s*	A string obtained by removing zero or more trailing symbols of string *s*; e.g., `ban` is a prefix of `banana`.
suffix of s	A string obtained by removing zero or more leading symbols of string *s*; e.g., `nana` is a suffix of `banana`.
substring of s	A string obtained by deleting a prefix and a suffix from *s*; e.g., `nan` is a substring of `banana`. Every prefix and every suffix of *s* is a substring of *s*, but not every substring of *s* is a prefix or suffix of *s*. For every string *s*, both *s* and Î are prefixes, suffixes, and substrings of *s*.
proper prefix, suffix or substring of *s*	Any nonempty string *x* that is, respectively, a prefix, suffix, or substring of *s* such that s $s \neq x$.
subsequence of s	Any string formed by deleting zero or more not necessarily contiguous symbols from *s*; e.g., `baaa` is subsequence of `banana`.

Example: Let *L* the set {A, B, …, Z, a, b, …, z} and D the set {0, 1, …, 9}. We can think of *L* as the set of letters and *D* as the set of digits. The list below contains examples of new languages created for *L* and *D*.

1. $L \cup D$ is the set of letters and digits.
2. *LD* is the set of strings consisting of a letter followed by a digit.
3. L^4 is the set of all four-letter strings.
4. L* is the set of all strings of letters, including Î, the empty string.
5. $L(L \cup D)^*$ is the set of all strings of letters and digits beginning with a letter.
6. D^+ is the set of all strings of one or more digits.

[1] Excepted from Aho, Sethi, and Ullman *Compilers, principles, techniques, and tools*. Addison-Wesley, 1986, ISBN 0-201-10088-6.

Operation	Definition
union of L and M written $L \cup M$	$L \cup M = \{s \mid s \in L \text{ or } s \in M\}$
concatenation of L and M written LM	$LM = \{st \mid s \in L, t \in M\}$
Kleene closure of L written L^*	$L^* = \bigcup_{i=0}^{\infty} L^i$ L^* denotes "zero or more concatenations of" L
positive closure of L written L^+	$L^* = \bigcup_{i=1}^{\infty} L^i$ L^* denotes "one or more concatenations of" L

Regular Expressions

1. Î is a regular expression that denotes { Î }, that is, the set containing the empty string.
2. If *a* is a symbol in Σ, then a is a regular expression that denotes {*a*}, i.e., the set containing the string *a*. Although we use the same notation for all three, technically, the regular expression *a* is different from the string *a* or the symbol *a*. It will be clear from the context whether we are talking about *a* as a regular expression, string, or symbol.
3. Suppose *r* and *s* are regular expressions denoting the languages $L(r)$ and $L(s)$. Then
   - 3.1. $(r) \mid (s)$ is a regular expression denoting $L(r) \cup L(s)$
   - 3.2. $(r)(s)$ is a regular expression denoting $L(r)L(s)$
   - 3.3. $(r)^*$ is a regular expression denoting $(L(r))^*$
   - 3.4. (r) is a regular expression denoting $L(r)$

Examples:

1. The regular expression *a*|*b* denotes the set {a,b}.
2. The regular expression (*a*|*b*)(*a*|*b*) denotes {*aa*, *ab*, *ba*, *bb*}, the set of all strings of *a*'s and *b*'s of length two. Another regular expression for this same set is *aa* | *ab* | *ba* | *bb*.
3. The regular expression *a** denotes the set of all strings of zero or more *a*'s, i.e.,{ Î , *a*, *aa*, *aaa*, …}
4. The regular expression (*a*|*b*)* denotes the set of all strings containing zero or more instances of an *a* or *b*, that is, the set of all strings of *a*'s and *b*'s. Another regular expression for this set is (*a***b***)*.