Formal Language Definitions

Symbol

A symbol is a character, glyph, mark. It is an abstract entity that has no meaning by itself. Letters from various alphabets, digits and special characters are the most commonly used symbols.

Alphabet

A finite set of symbols. An alphabet is often denoted by sigma, yet can be given any name.

B = {0, 1} says B is an alphabet of two symbols, 0 and 1.

C = {a, b, c} says C is an alphabet of three symbols, a, b and c.

Sometimes space and comma are in an alphabet while other times they are meta symbols used for descriptions.

Strings (also called Words)

A string is a finite sequence of symbols from an alphabet.

01110 and 111 are strings from the alphabet B above.

aaabccc and b are strings from the alphabet C above.

A null string is a string with no symbols, usually denoted by epsilon. The null string has length zero.

Length of String

Vertical bars around a string indicate the length of a string expressed as a natural number. For example |00100| = 5, |aab| = 3, | epsilon | = 0

Formal Language (also called a Language):

A set of strings from an alphabet. The set may be empty, finite or infinite.
There are many ways to define a language. There are many classifications for languages.
Because a language is a set of strings, the words language and set are often used interchangeably in talking about formal languages.

L(M) is the notation for a language defined by a machine M.
The machine M accepts a certain set of strings, thus a language.

L(G) is the notation for a language defined by a grammar G.
The grammar G recognizes a certain set of strings, thus a language.

M(L) is the notation for a machine that accepts a language.
The language L is a certain set of strings.

G(L) is the notation for a grammar that recognizes a language.
The language L is a certain set of strings.

Operations on Languages:

The union of two languages is a language.
$L = L1 \cup L2$
The intersection of two languages is a language.
$L = L1 \cap L2$
The complement of a language is a language.
$L = \sum^* - L1$
The difference of two languages is a language.
$L = L1 - L2$

Regular Language, Regular Expression:

A set of strings from an alphabet. The set may be empty, finite or infinite.

Definition: A language is called a regular language if some finite automaton recognizes it.

The regular operations:

Let A and B be languages. We define the regular operation union, concatenation and star as follows:

- **Union**: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.
- **Concatenation:** $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$
- **Star** : $A^* = \{x_1 x_2 \ldots x_k \mid k>0 \text{ and each } x_i \in A\}$

Theorem

The class of regular languages is closed under the union operation. That is if A1 and A2 are regular languages then $A1 \cup A2$ is a regular language.

Theorem:

The class of regular languages is closed under the concatenation operation.

The building blocks of regular languages are symbols:

- concatenation of symbols to make strings (words),
- set union of strings and
- Kleene closure (denoted as *, also called the Kleene star

Informally, we use a syntax for regular expressions.

Using sigma as the set {0, 1} (an alphabet of two symbols):

-01110 is a string starting with the symbol 0 and then concatenating 1, then 1, then 1, and finally concatenating 0.

No punctuation is used between symbols or strings that are concatenated.

-(01+10) is a union of the two strings 01 and 10. The set {01, 10}

-(00+11)* is the Kleene closure of the union of 0 concatenated with 0 and 1 concatenated with 1.

The Kleene closure (star) is defined as the concatenation of none, one, two, or any countable number strings it applies to.
Examples of Kleene star:
1* is the set of strings {epsilon, 1, 11, 111, 1111, 11111, etc. }. This set is infinite.
(1100)* is the set of strings {epsilon, 1100, 11001100, 110011001100, etc. }
(00+11)* is the set of strings {epsilon, 00, 11, 0000, 0011, 1100, 1111, 000000, 000011, 001100, etc. }

Note how the union (+ symbol) allows all possible choices of ordering when used with the Kleene star.

(0+1)* is all possible strings of zeros and ones, often written as $\sum^*$ where $\sum = \{0, 1\}$
(0+1)* (00+11) is all strings of zeros and ones that end with either 00 or 11.
Note that concatenation does not have an operator symbol.

$(w)^+$ is a shorthand for (w)(w)*, w is any string or expression and the superscript plus, + , means one or more copies of w are in the set defined by this expression.

Grammar, a formal grammar

A grammar is defined as G = (V, T, P, S) where:

- V is a set of symbols called variables, typically S, A, B, ...
- T is a set of symbols called terminal, typically 0, 1, a, b, ...
- P is a set of productions
- S is the starting or goal symbol from V

The productions P are of the form:

$A \rightarrow w$

Where A is a variable, w is any concatenation of variables and terminals.

An example grammar is G = (V, T, P , S)

where V = { S, A } T = { 0, 1 } and the productions, P , are:

$S \rightarrow 0A \mid 0$

$A \rightarrow 10A$

This grammar corresponds to the regular expression 0(10)* which in turn corresponds to a DFA

Regular Grammar

A grammar is defined as G = (V, T, P, S) where:

V is a set of symbols called variables, v1, v2, ... ,vn
T is a set of symbols called terminal, t1, t2, ,,, ,tm
P is a set of productions
S is the starting or goal variable from V

The productions P are of the form:

$A \rightarrow w$

$A \rightarrow wB$

Where:

- A and B are variables
- w is any combination terminals, may be empty string.

Any regular grammar can be converted to an equivalent DFA, NFA, regular language or regular expression.

Context Free Language, CFL

A grammar G = (V, T, P, S) with the productions:

S→ Aw

Where:

- S, A are non terminal symbols.
- w is a string of 0 or more terminals and non terminal symbols.

Context Free Languages are related to push down automata.

Chomsky Normal Form, CNF:

A grammar G = (V, T, P, S) with the productions restricted to the forms:

variable → variable variable

variable → terminal

A → B C A, B and C are variables in V and exactly two variables are on the right

A → a A is a variable in V and a is exactly one terminal symbol in T.

Recursive Languages, recursive sets :

The languages, sets, accepted by Turing machines and unrestricted grammars.

Recursively enumerable sets, r.e. languages

The sets, languages, that can be generated (enumerated) where all strings in the set (language) of a given length can be generated.

Usually the enumeration is strings of length 1, then strings of length 2, and so forth. Of course there may be no strings for some lengths.

In some cases, generate $\sum^n$ and pass each string to a machine or grammar. The accepted strings are the strings of length n.
There is no requirement that the strings be generated in lexical or any other order.
If both a set and its complement are recursively enumerable, then the set is recursive.

Chomsky Hierarchy of Grammars/Languages

Type 0, Grammars that generate r.e. sets and characterize the r.e. languages

Unrestricted grammars of the form G = (V, T, P ,S)
The restriction is removed from the form of the productions.
$(T \cup V)(V \cup T)$
Type 0 grammars can have infinite loops in the parser for the grammar, when a string not in the grammar, is input to the parser (not decidable)

Type 1, Grammars that characterize context sensitive languages.

Type 2, Grammars that characterize context free languages.

Type 3, Grammars that characterize regular languages.

P and NP classes of languages

A class of languages is a set of languages that share some characteristic. Since a language is a set of strings from a finite alphabet, a class of languages is a set of sets.

The language class P is the set of languages for which there exists a deterministic Turing machine that accepts each language in a number of transitions bounded by a fixed polynomial in the length of the input string.
Start with a "standard" Turing machine with a finite state control that

is deterministic, the TM has a transition table with one entry for each (state, input-symbol) pair.
Consider a specific language that has strings of various lengths. Let the length of any string in the language be denoted n.

If there exists a fixed polynomial in n, e.g. cn^r for some fixed constant c and some fixed constant r, such that the Turing machine accepts or rejects all strings in the specific language in cn^r moves, transitions, then that specific language is in the set P.

The language class NP is different from the language class P in two ways:

1) NP languages have Turing machine with a nondeterministic finite state control.
2) NP languages have a Turing machine that does not have to reject a string in any prescribed number of moves.

Note: Without the time restriction, bounded number of moves, any nondeterministic Turing machine has an equivalent deterministic Turing machine. It is believed that the language class P is not equivalent to the language class NP but this belief is not yet proven.

Reducibility and Unsolvability:

Definition of Reducibility:

The language L1 is reducible to the language L2 if there is an algorithm computing a total function f: C* → D* that translates each string w over the alphabet C of L1 into a string z= f(w) over the alphabet D of L2 such that $w \in L1$ if and only if $z \in L2$.

In this definition, testing for membership of a string w in L1 is reduced to testing for membership of a string z in L2, where the latter problem is presumably a previously solved problem.

Reducibility establish a link between two problems with the expectation that the properties of one can be used to deduce properties of the other.

Lemma: Let L1 be reducible to L2. If L2 is decidable, then L1 is decidable. If L1 is unsolvable and L2 is recursively enumerable, L2 is insolvable.

Unsolvable Problems:

We examine some representative unsolvable problems:
The halting problem for Turing Machines:
Determine for an arbitrary TM M and an arbitrary input string x whether M with input x halts or not.

Let $L_h = \{\rho(M), w \mid M \text{ halts on input } w\}$

Theorem: The language L_h is recursively enumerable but not decidable.

Rice Theorem:
The Rice Theorem says that no algorithm exists to determine from the description of TM M whether or not the language it accepts falls into any proper subset of recursively enumerable languages.

Let RE be the set of recursively enumerable languages over B. For each set C that is a proper subset of RE, define the following language: $Lc = \{ \rho(M) \mid L(M) \in C\}$

Rice theorem says that, for all C such that $C \neq \varnothing$ and $C \subset RE$, the language Lc defined above is undecidable.