Computation theory

Computations are designed to solve problems. Computations are designed for processing information. They can be as simple as an estimation for driving time between cities, and as complex as a weather prediction.

Alphabets and Strings

Alphabets and Strings

The ability to represent information is crucial to communicate and process information. Human societies created spoken languages to communicate on a basic level, and developed writing to reach a more sophisticated level.

The English language, for instance, in its spoken form relies on some finite set of basic sounds as a set of primitives. The words are defined in term of finite sequences of such sounds. Sentences are derived from finite sequences of words. Conversations are achieved from finite sequences of sentences, and so forth.

Written English uses some finite set of symbols as a set of primitives. The words are defined by finite sequences of symbols. Sentences are derived from finite sequences of words. Paragraphs are obtained from finite sequences of sentences, and so forth.

Similar approaches have been developed also for representing elements of other sets. For instance, the natural number can be represented by finite sequences of decimal digits.

Computations, like natural languages, are expected to deal with information in its most general form. Consequently, computations function as manipulators of integers, graphs, programs, and many other kinds of entities. However, in reality computations only manipulate strings of symbols that represent the objects. The previous discussion necessitates the following definitions.

Alphabets and Strings

A finite, nonempty ordered set will be called an *alphabet* Σ *(sigma)* if its elements are *symbols* , or *characters*. A finite sequence of symbols from a given alphabet will be called a *string* over the alphabet. Some books refer to strings as words only if they talk about strings contained in a specific language. Some people (including me) do not necessarily make this distinction.

A string that consists of a sequence $a_1, a_2, \ldots, a_n$ of symbols will be denoted by the juxtaposition $a_1a_2 a_n$. Strings that have zero symbols, called *empty strings*, will be denoted by λ .

Example $\Sigma_1 = \{a, \ldots, z\}$ and $\Sigma_2 = \{0, \ldots, 9\}$ are alphabets. abb is a string over Σ_1, and 123 is a string over Σ_2. ba12 is not a string over Σ_1, because it contains symbols that are not in Σ_1. Similarly, 314 .. . is not a string over Σ_2, because it is not a finite sequence. On the other hand, λ is a string over any alphabet.
The empty set Ø is not an alphabet because it contains no element. The set of natural numbers is not an alphabet, because it is not finite. The union $\Sigma_1 \cup \Sigma_2$ is an alphabet only if an ordering is placed on its symbols.
An alphabet of cardinality 2 is called a *binary alphabet*, and strings over a binary alphabet are called *binary strings*. Similarly, an alphabet of cardinality 1 is called a *unary alphabet*, and strings over a unary alphabet are called *unary strings*.
The *length* of a string α is denoted $|\alpha|$ and assumed to equal the number of symbols in the string.

Example {0, 1} is a binary alphabet, and {1} is a unary alphabet. 11 is a binary string over the alphabet {0, 1}, and a unary string over the alphabet {1}.
11 is a string of length 2, $|\lambda| = 0$, and $|01| + |1| = 3$.
The string that consists of a sequence α followed by a sequence β is denoted $\alpha\beta$. The string is called the *concatenation* of α and β . The notation α^i is used for the string obtained by concatenating i copies of the string .

Example The concatenation of the string 01 with the string 100 gives the string 01100.
If $\alpha = 01$, then $\alpha^0 = \lambda$, $\alpha^1 = 01$, $\alpha^2 = 0101$, and $\alpha^3 = 010101$.

Formal Languages

The universe of strings is a useful medium for the representation of information as long as there is a function that provides the interpretation for the information carried by the strings. An interpretation is just the inverse of the mapping that a representation provides, that is, an *interpretation* is a function g from Σ^* to D for some alphabet Σ and some set D. The string 111, for instance, can be interpreted as the number one hundred and eleven

represented by a decimal string, as the number seven represented by a binary string, and as the number three represented by a unary string. The parties communicating a piece of information do the representing and interpreting. The representation is provided by the sender, and the interpretation is provided by the receiver. The process is the same no matter whether the parties are human or programs. Consequently, from the point of view of the parties involved, a language can be just a collection of strings because the parties embed the representation and interpretation functions in themselves.

Languages

In general, if is an alphabet Σ and L is a subset of Σ*, then L is said to be a *language* over Σ, or simply a language if Σ is understood. Each element of L is said to be a *sentence* or a *word* or a *string* of the language.

Example {0, 11, 001}, {λ , 10}, and {0, 1}* are subsets of {0, 1}*, and so they are languages over the alphabet {0, 1}. The empty set Ø and the set {λ } are languages over every alphabet. Ø is a language that contains no string. { λ} is a language that contains just the empty string.

The *union* of two languages L_1 and L_2, denoted $L_1 \cup L_2$, refers to the language that consists of all the strings that are either in L_1 or in L_2, that is, to { x | x is in L_1 or x is in L_2 }.

The *intersection* of L_1 and L_2, denoted $L_1 \cap L_2$, refers to the language that consists of all the strings that are both in L_1 and L_2, that is, to { x | x is in L_1 and in L_2 }.

The *complementation* of a language L over Σ, or just the complementation of L when is understood, denoted L^-, refers to the language that consists of all the strings over Σ that are not in L, that is, to { x | x is in Σ* but not in L }.

Example Consider the languages L_1 = {λ , 0, 1} and L_2 = { λ, 01, 11}. The union of these languages is $L_1 \cup L_2$ = {λ , 0, 1, 01, 11}, their intersection is $L_1 \cap L_2$ = {λ }, and the complementation of L_1 is L^-_1 = {00, 01, 10, 11, 000, 001, . . . }.

DONE

Instructor: Mohamed U. Mahdi