Lecture A/1: Introduction to Languages

Dr Janka Chlebikova

School of Computing
University of Portsmouth

"We can only see a short distance ahead, but we can see plenty there that needs to be done." [Alan Turing]

Oct 3, 2024

Spoken and computer languages

The languages we use are built for communicating and passing along instructions to other humans or to computers.

- These are defined in terms of the set of symbols – alphabet – which are combined into acceptable strings (words/sentences or keywords/computer programs).
- The rules which tell us how to combine strings in sensible ways are called the grammar.
- Using alphabet, strings and grammar we define the language.

Example (spoken language): English

Alphabet: The alphabet for the English language is Latin $A = \{a, b, c, d, e, \ldots, x, y, z\}$

Strings/words: strings are formed from A, e.g. fun, mathematics – the English vocabulary defines which are really strings, e.g. which appear in the Oxford English dictionary

Grammar: from this collection of words we can build sentences using the English rules of grammar

Language: the set of possible sentences makes up the English language

The elements of formal language are exactly the same, but must be defined without any ambiguity! E.g. programming languages, . . . the precise description of syntax must be given

Alphabets and strings

Definition (Alphabet)

A finite, nonempty set of symbols is called an alphabet.

Example: $\Sigma = \{a, b, c\}$ is an example of the alphabet.

Definition (String)

A string is a finite sequence of symbols from the alphabet (place next to each other in juxtaposition).

Example: abc, aaa, bb are examples of strings on Σ.

A string with no symbols is called an empty string and is denoted Λ. (It has zero length.)

Languages

Definition (Language)

If Σ is an alphabet, then a language over Σ is a set of strings (including empty string Λ) whose symbols come from Σ.

Example: If $\Sigma = \{a, b\}$, then $L = \{ab, aaaab, abbb, a\}$ is an example of a language over Σ.

If Σ is an alphabet, then Σ^* denotes the infinite set of all strings made up from Σ (including an empty string Λ).

Example: If $\Sigma = \{a, b\}$, then $\Sigma^* = \{\Lambda, a, b, ab, aab, aaab, bba, \ldots\}$.

Looking at Σ^*, a language over Σ is any subset of Σ^*.

Examples of languages

Four simple examples of languages over an alphabet Σ are the sets:

- $\emptyset$, $\{\Lambda\}$, Σ, and Σ^*.

Example: If $\Sigma = \{a\}$ then these four simple languages over Σ are:

- $\emptyset$, $\{\Lambda\}$, $\{a\}$, and the infinite set $\Sigma^* = \{\Lambda, a, aa, aaa, \ldots\}$.

Recall $\{\Lambda\}$ is the set containing the empty string while $\emptyset$ is the empty set.

Combining Languages: union and intersection

There are three common ways of creating a new language from two languages: union, intersection, and product.

Languages are sets of strings, so they can be combined by the usual set operations of union and intersection.

Example: If $L = \{aa, bb, ab\}$ and $M = \{ab, aabb\}$ then

$$L \cap M = \{ab\}$$

and

$$L \cup M = \{aa, bb, ab, aabb\}.$$

Combining Languages: product

The operation of concatenation of strings places two strings in juxtaposition.

Example: The concatenation of the two strings *aab* and *ba* is the string *aabba*.

We use the name cat to denote this operation: cat(aab, ba) = aabba.

We can combine two languages L and M by forming the set of all concatenations of strings in L with strings in M, which is called the product of the two languages.

Combining Languages: product

Definition (Product of two languages)

If L and M are languages, then the new language called the product of L and M is defined as $L \cdot M$ (or only LM)

$$L \cdot M = \{cat(s, t) : s \in L \text{ and } t \in M\}$$

Example: If $L = \{ab, ac\}$ and $M = \{a, bc, abc\}$, then the product $L \cdot M$ is the language

$$L \cdot M = \{aba, abbc, ababc, aca, acbc, acabc\}.$$

Contrast this with the union of the sets:

$$L \cup M = \{a, ab, ac, bc, abc\}$$

Basic properties of products

It is easy to see that for any language L the following simple properties hold:

$$L \cdot \{\Lambda\} = \{\Lambda\} \cdot L = L$$
$$L \cdot \emptyset = \emptyset \cdot L = \emptyset$$

And how about the commutativity and associativity of the operation of concatenation?

Properties of products – commutativity

The operation of concatenation is not commutative. In other words, the order matters! Given two languages L and M, it's usually true that:

$$L \cdot M \neq M \cdot L$$

Example. If $L = \{ab, ac\}$ and $M = \{a, bc, abc\}$, then the product $L \cdot M$ is the language

$$L \cdot M = \{aba, abbc, ababc, aca, acbc, acabc\},$$

but the product $M \cdot L$ is the language

$$M \cdot L = \{aab, aac, bcab, bcac, abcab, abcac\}$$

These have no strings in common!

Properties of products – associativity

The operation of concatenation is associative. In other words, if L, M, and N are languages, then

$$L \cdot (M \cdot N) = (L \cdot M) \cdot N$$

Example: If $L = \{a, b\}$, $M = \{a, aa\}$ and $N = \{c, cd\}$, then

$$\begin{aligned} L \cdot (M \cdot N) &= L \cdot \{ac, acd, aac, aacd\} \\ &= \{aac, aacd, aaac, aaacd, bac, bacd, baac, baacd\}. \end{aligned}$$

Which is the same as,

$$\begin{aligned} (L \cdot M) \cdot N &= \{aa, aaa, ba, baa\} \cdot N \\ &= \{aac, aacd, aaac, aaacd, bac, bacd, baac, baacd\}. \end{aligned}$$

Powers of languages

If L is a language, then the product $L \cdot L$ is denoted by L^2.
The language product L^n for every $n \in \{0, 1, 2, \dots\}$ is defined as follows:

$$L^0 = \{\Lambda\},$$
$$L^n = L \cdot L^{n-1}, \text{ if } n > 0$$

Example. If $L = \{a, bb\}$ then the first few powers of L are

$$L^0 = \{\Lambda\},$$
$$L^1 = L = \{a, bb\},$$
$$L^2 = L \cdot L = \{aa, abb, bba, bbbb\}$$
$$L^3 = L \cdot L^2 = \{aaa, aabb, abba, abbbb, bbaa, bbabb, bbbba, bbbbbb\}$$

Closure of a language

If L is a language over Σ (i.e. $L \subset \Sigma^*$) then the closure of L is the language denoted by L^* and is defined as follows:

$$L^* = L^0 \cup L^1 \cup L^2 \cup \cdots$$

The positive closure of L is the language denoted by L^+ and defined as follows:

$$L^+ = L^1 \cup L^2 \cup L^3 \cup \cdots$$

L^* versus L^+

It follows that $L^* = L^+ \cup \{\Lambda\}$.
But it's not necessarily true that

$$L^+ = L^* - \{\Lambda\}.$$

Example: If we let our alphabet be $\Sigma = \{a\}$ and our language be $L = \{\Lambda, a\}$, then

$$L^+ = L^*.$$

Can you find a condition on a language L such that $L^+ = L^* - \{\Lambda\}$?

Properties of Closure

Based on these definitions, we can derive some interesting properties of the closure operation. Let L and M be languages over the alphabet Σ. Then:

- $\{\Lambda\}^* = \emptyset^* = \{\Lambda\}$
- $L^* = L^* \cdot L^* = (L^*)^*$
- $\Lambda \in L$ if and only if $L^+ = L^*$
- $(L^* \cdot M^*)^* = (L^* \cup M^*)^* = (L \cup M)^*$
- $L \cdot (M \cdot L)^* = (L \cdot M)^* \cdot L$

We'll derive a few of these in the tutorials ...

Closure of an alphabet

Do you remember the definition of Σ^* for the alphabet Σ?

The closure of Σ coincides with our definition of Σ^* as the set of all strings over Σ. In other words, we have a nice representation of Σ^* as follows:

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots$$

where Σ^k denote the set of strings of length k, each of whose symbols is in Σ.

Do you remember at least something?

Question 1. Within computer science concepts, what do we mean by a language?

1. A language is a set of strings.
2. A language is a sequence of symbols.
3. A language is a set of symbols.
4. A language is a sequence of strings.

Question 2. If a language A is $\{a, b\}$ and another language B is $\{0, 1\}$, what is $A \cdot B$?

1. $A \cdot B = \{a0, b1\}$
2. $A \cdot B = \{a0, a1, b0, b1\}$
3. $A \cdot B = \{a \cdot 0, a \cdot 1, b \cdot 0, b \cdot 1\}$
4. $A \cdot B = \{a \cdot 0, a \cdot 1, b \cdot 0, b \cdot 1, 0 \cdot a, 1 \cdot a, 0 \cdot b, 1 \cdot b\}$