BSc (Hons) Computer Science
University of Portsmouth
Third Year

Theoretical Computer Science

M21276
Semester 1

Hugh Baldwin
hugh.baldwin@myport.ac.uk

Contents

Lecture - Induction Lecture

13:00	25/09/24	Janka Chlebikova

Contact Time

- Lectures
  - Delivered by Janka
  - Slides and videos from Covid times available weekly on Moodle
- Tutorials
  - Delivered by either Janka or Dr Paolo Serafino
  - Worksheets available with solutions weekly on Moodle
  - 5 Groups
  - Work through the worksheet solutions
  - Kahoot quiz for revision
  - ‘Tutorial 0’ available as revision of concepts from DMAFP needed for this module

Assessments

- 50%, 90 minute In-Class Test covering Part A - November 20th, 2pm
- 50%, 90 minute Exam covering Part B (Only) - TB1 Assessment Period (January)
- Deferred assessment covers both Part A & B, so will be harder than the two separate exams

There are two past papers (paper-based, but still relevant to the computer-based test) available for each of the assessments. As with DMAFP, the solutions are supplied in the form of a crowd-sourced document that Janka will check over before the exam.

Resources

Lecture slides, videos and tutorial papers are released every week. All of the books on the reading list are available online through the links on Moodle.

Lecture - A1: Introduction to Languages

13:00 03/10/24 Janka Chlebikova

Languages

In this context, a **language** is a set of symbols which can be combined to create a list of acceptable **strings**. There are then also rules which tell us how to combine these strings together, known as **grammars**. The combination of an alphabet, list of valid strings and grammars is known as a language. There will be a formal definition later on.

Definition

An **Alphabet** is a finite, non-empty set of symbols.

For example, in the English language we would define the alphabet, A, as $\mathrm{A} = \{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \ldots, \mathrm{x}, \mathrm{y}, \mathrm{z}\}$. These symbols can then be combined to create a **string**.

Definition

A **String** is a finite sequence of symbols from the alphabet of a language.

With the alphabet A, we could have strings such as 'cat', 'dog', '$\mathrm{antidisestablishmentarianism}$', etc *over* the alphabet. A string with no symbols, and therefore a length of zero, is known as the **empty string**, and is denoted by Λ (capital lambda).

Definition

A **Language** over an alphabet (e.g. English over A) is a set of strings – including Λ – made up of symbols from A which are considered 'valid'. They could be valid as per a set of rules, or could be arbitrary as in most spoken languages.

If we have an alphabet, Σ, then Σ^* denotes the infinite set of all strings made up of symbols in Σ – including Λ. Therefore, a language over Σ is any subset of Σ^*.

Example

If $\Sigma = \{\mathrm{a}, \mathrm{b}\}$, then $\Sigma^* = \{\Lambda, \mathrm{a}, \mathrm{b}, \mathrm{aa}, \mathrm{ab}, \mathrm{ba}, \mathrm{bb}, \ldots\}$
There are many languages which could be defined over this alphabet, but a few simple one are

- $\emptyset$ (The empty set)
- $\{\Lambda\}$ (The set containing an empty string)
- $\{\mathrm{a}\}$ (The set containing a)
- The infinite set $\Sigma^* = \{\Lambda, \mathrm{a}, \mathrm{aa}, \mathrm{aaa}, \ldots\}$

Combining Languages

There are three common ways of combining two languages to create a new language – union, intersection and product.

Union and Intersection

Since languages are sets of strings, they can be combined as you usually would for any other set with the usual operations, union and intersection.

Example

If $L = \{aa, bb, cc\}$ and $M = \{cc, dd, ee\}$, then
$L \cap M = \{cc\}$, and
$L \cup M = \{aa, bb, cc, dd, ee\}$

Product

To combine two languages L and M, we form the set of all **concatenations** of strings in L with strings In M. This is known as the **product** of the two languages.

Definition

To **Concatenate** two strings is to juxtapose them such that a new string is made by appending the second string to the end of the first.

Example

If we concatenate the strings ab and ba, the result would be abba.
This can be represented using the function cat, such as $cat(ab, ba) = abba$

With this definition of concatenation, we can say that the product of two languages, L and M would be $L \cdot M$, where $L \cdot M = \{cat(s, t) : s \in L \text{ and } t \in M\}$

Example

If $L = \{a, b, c\}$ and $M = \{c, d, e\}$, then the product of the two languages, $L \cdot M$, is the language
$L \cdot M = \{ac, ad, ae, bc, bd, be, cc, cd, ce\}$

Further on Products

It is simple to see that for any language, L, the following simple properties are true:

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

Commutativity

Aside from the above properties, the product of any two languages is **not** commutative, and therefore

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

Example

If $L = \{a, b\}$ and $M = \{b, c\}$, then the product $L \cdot M$ is the language

$$L \cdot M = \{ab, ac, bb, bc\}$$

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

$$M \cdot L = \{ba, bb, ca, cb\}$$

which are clearly not the same language, as the only common string is bb

Associativity

However, the product of any two languages is associative. This means that if we had any three languages, L, M and N, then

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

Example

If we have the three languages $L = \{a, b\}$, $M = \{b, c\}$ and $N = \{c, d\}$, then

$$\begin{aligned} L \cdot (M \cdot N) &= L \cdot \{bc, bd, cc, cd\} \\ &= \{abc, abd, acc, acd, bbc, bbd, bcc, bcd\} \end{aligned}$$

which is the same as

$$\begin{aligned} (L \cdot M) \cdot N &= \{ab, ac, bb, bc\} \cdot N \\ &= \{abc, abd, acc, acd, bbc, bbd, bcc, bcd\} \end{aligned}$$

Powers of Languages

If we have the language L, then the product $L \cdot L$ of the language is denoted by L^2. The product L^n for every $n \in \mathbb{N}$ is defined as

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

Example

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

$$\begin{aligned} L^0 &= \{\Lambda\} \\ L^1 &= L = \{a, b\} \\ L^2 &= L \cdot L = \{aa, ab, ba, bb\} \\ L^3 &= L \cdot L^2 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\} \end{aligned}$$

The Closure of a Language

If L is a language over Σ, then the **closure** of L is the language denoted by L^*, and the **positive closure** is language denoted by L^+.

Definition

The **Closure** of the language L, L^* is defined as

$$L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \dots$$

and so, if $L = \{a\}$ then

$$\begin{aligned} L^* &= \{\Lambda\} \cup \{a\} \cup \{aa\} \cup \{aaa\} \cup \dots \\ &= \{\Lambda, a, aa, aaa, \dots\} \end{aligned}$$

Definition

The **Positive Closure** the language L, L^+ is defined as

$$L^+ = L^1 \cup L^2 \cup L^3 \cup L^4 \cup \dots$$

and so, if $L = \{a\}$ then

$$\begin{aligned} L^+ &= \{a\} \cup \{aa\} \cup \{aaa\} \cup \{aaaa\} \cup \dots \\ &= \{a, aa, aaa, aaaa, \dots\} \end{aligned}$$

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

Example

If our alphabet is $\Sigma = \{a\}$ and our language is $L = \{\Lambda, a\}$, then

$$L^+ = L^*$$

Properties of Closures

Based upon these definitions, you can derive some interesting properties of closures.

Example

If L and M are languages over the alphabet Σ, then

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

These will be explored more during the tutorial session for this week

The Closure of an Alphabet

Going back to the definition of Σ^* of the alphabet Σ, it lines up perfectly with the definition of a closure such that Σ^* is the set of all strings over Σ. This means that there is a nice way to represent Σ^* as follows

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots$$

From this, we can also see that Σ^k denotes the set of all strings over Σ whose length is k.