Automata, Languages and Computation

videos of lectures of 2 years ago available on the course page

Introduction

One of the main goals of theoretical computer science is the mathematical study of **computation**

- computability : what can be computed ?
- tractability : what can be **efficiently** computed ?

The mathematical study of computation requires

- abstract models of machine : **automata theory**
- abstract representations of data : **formal language theory**

Introduction

Most well-known models of computation :

- Turing machines, introduced for the study of computability
- finite automata, introduced as models of neuronal calculus
- formal grammars, introduced by Noam Chomsky as linguistic models

Structural Representation

give input to the model and it gives you an output

An FA is a **recognition** model : it takes as input a sequence (string) and either accepts or rejects

Alternatively, we can use a **generative** model : such model generates all of the **desired** sequences (no input)

Recognition models are operational, generative models are declarative

don't get input but gives some sort of output (or all possible solutions)

Deductive proof

Typical form of the statement to be proved (H, C properties) :

If H, then C

also written as $H \Rightarrow C$, where H = **hypothesis**, C = **conclusion**

This means

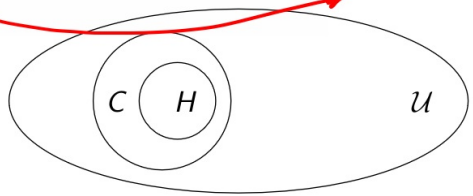
- H is a **sufficient** condition for C
- C is a **necessary** condition for H

See insiemistic interpretation in next slide

Deductive proof

In an **insiemistic** interpretation, H and C are associated with all the elements of the universe $\mathcal{U}$ that have that property

$H \Rightarrow C$ is equivalent to $H \subseteq C$: if H is true, C can't be false



Many students at the final exam use $H \Rightarrow C$ and $C \Rightarrow H$ interchangeably:

don't do that! (duhhh ...)

Additional techniques

Another technique used a lot

↑ If I ask to show that a theorem is false
Find an example that breaks it

Counterexample : to prove that a theorem is false it is enough to show a case in which the statement is false

Example :

Is it true that if x is a prime number, then x is odd ?

No, in fact 2 is a prime number but it is not odd

Inductive proof

↳ We use it when we have infinite long list

Main technique when working on **recursively** defined objects (expressions, trees, etc.)

- base case to prove base elements
- induction to reach every element

Induction on integers : we need to prove statement $S(n)$, for non-negative integer numbers n

- in the **base** case we show $S(i)$ for some specific integer i (usually $i = 0$ or $i = 1$)
- in the **inductive** step, for $n \geqslant i$ prove statement "if $S(n)$ then $S(n+1)$"

We can then conclude that $S(n)$ is true for every $n \geqslant i$, where i is the base case

Think: why is induction so powerful?

Alphabet & strings

(Formal Language Theory) is based on 3 steps:

1) **Alphabet**: finite and nonempty set of atomic symbols → Basic bricks to build more complex objects
   ↳ can't decompose the symbols.

Example:
- $\Sigma = \{0, 1\}$, the binary alphabet ← symbol "1"
- $\Sigma = \{a, b, c, \ldots, z\}$, the set of all lowercase letters
- the set of all printable ASCII characters

Alphabet

We can have an infinite amount of copies of a symbol.

2) **String**: finite sequence of symbols from some alphabet ← ordered
   - 0011001 string over $\Sigma = \{0, 1\}$

↳ usually from the same alphabet
(we can merge alphabets if needed)

Alphabet & strings

Empty string : The string with zero symbols (taken from any alphabet) is denoted ϵ

Length of a string : Number of **occurrences** (standpoints) for the symbols in the string

- $|w|$ denotes the length of the string w
- $|0110| = 4$, $|\epsilon| = 0$

Alphabet & strings

stella di klenee ma limitata a una lunghezza finita

Powers of an alphabet : Σ^k is the set of all k-length strings with symbols from Σ

- $\Sigma = \{0, 1\}$
- $\Sigma^1 = \{0, 1\}$; **ambiguity** between Σ and Σ^1

  Elements of Σ are alphabet symbols, elements of Σ^1 are strings
- $\Sigma^2 = \{00, 01, 10, 11\}$
- $\Sigma^0 = \{\epsilon\}$

{'0','1'} → symbols
{"0","1"} → strings

Question : How many strings are there in Σ^3 ?

↳ $|\Sigma|^3 = 2^3 = 8$

In general size $\Sigma^n = |\Sigma|^n$ (ϵ is not to be counted cause it has length 0)

Alphabet & strings

The set of all strings from Σ is denoted Σ^*

We have

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots$$
$$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \cdots$$
$$\Sigma^* = \Sigma^+ \cup \{\epsilon\}$$

It is a mistake to write $\Sigma^+ \cup \epsilon$: why?

Alphabet & strings

Concatenation : If x and y are strings, then xy is the string obtained by putting a copy of y immediately after a copy of x

Example :

$$x = 01101$$
$$y = 110$$
$$xy = 01101110$$

Sometimes we also use the the '$\cdot$' operator to represent concatenation and write $x \cdot y$

Some textbooks use the notation $x.y$

Alphabet & strings

For each string x :

$$x\epsilon = \epsilon x = x$$

ϵ is the **neutral** element of the concatenation

You can always think of ϵ occurring any number of times within a string :

$$\begin{aligned} x \cdot y &= x \cdot \epsilon \cdot y \\ &= x \cdot \epsilon \cdot \epsilon \cdot y \\ &= x \cdot \epsilon \cdot \epsilon \cdot \epsilon \cdot y \\ &= \cdots \end{aligned}$$

Compare with $2 + 3 = 2 + 0 + 3 = 2 + 0 + 0 + 3 = \cdots$

Alphabet & strings

Notational conventions :

- a, b, c, ..., a_1, a_2, ..., a_i, ... alphabet symbols
- u, w, x, y, z strings
- for $n \geqslant 0$, $a^n = aa \cdots a$ (a repeated n times)
- $a^0 = \epsilon$, $a^1 = a$

symbol

string

symbol

strings