

Theory of Computation

MIEIC, 2nd Year

João M. P. Cardoso



Dep. de Engenharia Informática
Faculdade de Engenharia (FEUP)
Universidade do Porto
Porto
Portugal

Email: jmpc@acm.org

Outline

- ▶ Introduction to the topics of the course
- ▶ Concepts about Automata
- ▶ Proof method by induction

History

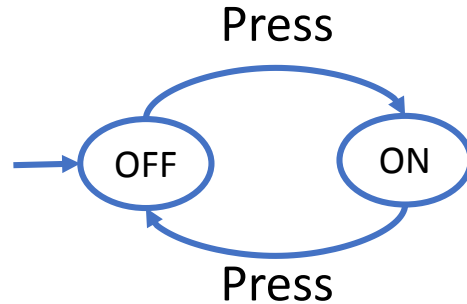
- ▶ Automata theory: study of abstract computing devices [*machines*]
- ▶ Alan Turing (1930's)
 - ▶ Studied the limits of an abstract machine equivalent to the current ones
 - ▶ Before the existence of Computers!
- ▶ 1940's, 1950's
 - ▶ Study of finite automata to model the human brain
- ▶ Noam Chomsky (1950's)
 - ▶ Formal grammars – related to abstract automata and very useful in compilers
- ▶ Stephen Cook (1969)
 - ▶ Complexity theory – what is feasible or not to compute



Relevance of the Automata Theory

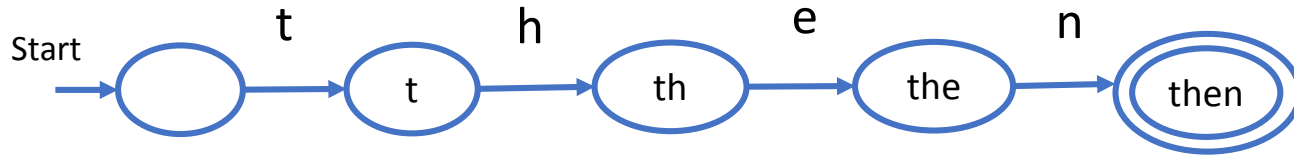
- ▶ Useful to model hardware and software
 - ▶ Design and test of digital circuits
 - ▶ Lexical analysis in compilers
 - ▶ Text processing, web search
 - ▶ State machines, communication protocols, security, cryptography, analytics
- ▶ Finite Automata
 - ▶ System that in each instant is in one of a finite number of states
 - ▶ State memorizes the part relevant of the history of the system
 - ▶ Being finite it needs to forget what is not relevant
 - ▶ It can be implemented with finite resources

Example of an Automaton: on/off switch



- ▶ Simple finite automaton – models switch
 - ▶ Two **states** [circles]: on and off
 - ▶ Only one **input** [labels in edges]: Press
 - ▶ Represents the external influence on the system [state transition]
 - ▶ Push button has an effect dependent of the state
 - ▶ **Initial state** represented by an arrow with label Start
 - ▶ There can exist one or more **final (acceptance) states**, represented by double circles

Example of an Automaton: recognizer



- ▶ If the input is the string “*then*” the automaton goes from the initial to the final state
 - ▶ Accumulate the history of the input
 - ▶ The goal is to recognize the string “*then*”

Structural Representations

► Grammars

- Process data with recursive structure [expressions]
- Example of a grammar rule: $E \Rightarrow E + E$
 - One expression can consist of two expressions connected by “+”
- Used in static analyzers [parsers] of compilers

► Regular Expressions

- Describe the structure of strings
- Example: `[1-9][0-9][0-9][0-9][-][0-9][0-9][0-9][][A-Z][a-z]*`
 - Describe “4200-465 Porto”, but not “5505-032 Vila Real”
 - Correction: `[1-9][0-9][0-9][0-9][-][0-9][0-9][0-9]([][A-Z][a-z]*)*`

Proof Methods

- ▶ Formal proofs are important for informatics engineers
 - ▶ There are people that think that the writing of a program should be accompanied by the respective demonstration of correctness (mathematical approach)
 - ▶ There are people that think that what counts is testing (experimental approach)
 - ▶ The reasonability is in the middle!
- ▶ Statements
 - ▶ **if ... then**
 - ▶ if A then B ($A \rightarrow B$)
 - ▶ **if and only if – iff**
 - ▶ A iff B ($A \leftrightarrow B$, prove: $A \rightarrow B$ and $B \rightarrow A$)

Proof Methods

- ▶ There are several proof methods (e.g., by deduction)
- ▶ if H then C ($H \rightarrow C$)
 - ▶ By contradiction: H and not C implies falsehood
 - ▶ By counter-example: show an example that proves the proposition is false
 - ▶ By counter-positive : if not C then not H (proving one is proving the other)
 - ▶ By induction (see the following slides)

Proof by Induction

- ▶ Proving a statement $S(n)$ over an integer n (or a structure defined inductively, such as a tree)
 - ▶ Basis: prove $S(i)$ for some small i 's, typically $i=0$ or $i=1$
 - ▶ Inductive step: assuming by **hypothesis** that $S(n)$ is true, show that $S(n+1)$ verifies
 - ▶ Being n general, the property verifies for all n
- ▶ Elements of an inductive proof
 - ▶ Structure over which we apply induction
 - ▶ Integers, trees, graphs, sets, strings
 - ▶ Statement $S(n)$ which we intend to prove (n is de step)
 - ▶ Base case (basis)
 - ▶ Induction/inductive step

Induction proofs

- ▶ The principle of induction

- ▶ If we prove $S(i)$ and prove that for $n \geq i$, $S(n)$ implies $S(n + 1)$, then we can conclude that $S(n)$ is true for any $n \geq i$

Example

- ▶ Prove that for any natural number n , the sum of the first n naturals is $n(n+1)/2$

Example (cont.)

► Proof:

Structure: \mathbb{N} is the set of natural numbers

Statement $S(n)$: the sum of the first n natural numbers is $n(n+1)/2$

Basis: the sum of the first 0 natural numbers is 0

Induction step: Let k a natural number for which $S(k)$ is true

Sum of the first k natural numbers is $k(k+1)/2$, by **hypothesis**

Sum of the first $k+1$ natural numbers:

$$k(k+1)/2 + k+1 = (k+1)(k/2 + 1) = (k+1)(k+2)/2 \quad (\text{applying the hypothesis})$$

which is exactly the expression of the sum of the first natural numbers given by the expression in the statement $S(k+1) = (k+1)(k+2)/2$

Q.E.D. (*quod erat demonstrandum*)

Widening the scope of the concept

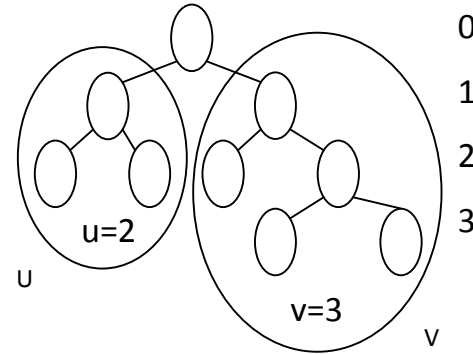
- ▶ To prove statements of the form:
 - ▶ $\forall n [P(n) \rightarrow S(n)]$
- ▶ Induction necessary when $P(n)$ has an inductive definition

Example

- ▶ Prove that a quasi complete binary tree with k leaves has $2k-1$ nodes
- ▶ Inductive definition of a quasi complete binary tree (ABqc):
 - ▶ An isolated node is a ABqc
 - ▶ If U and V are ABqc, then a node with U and V as children is a ABqc
- ▶ Proof based on the structure of the tree
- ▶ Structure: set of binary trees
 - ▶ n step: trees with height n
 - ▶ We could have selected the number of nodes but we preferred the structure of the tree (height)

Example (cont.)

- ▶ Statement $S(T)$: if T is a binary tree with k leaves then T has $2k-1$ nodes
- ▶ Basis: a tree with height 0, only root, has 1 leaf and $2 \times 1 - 1 = 1$ node
- ▶ Induction step: Assume $S(U)$ for the trees of order until n and in particular for the subtrees of T
 - ▶ T is a tree with order $n+1$ with root and two subtrees U and V (at least one of order n)
 - ▶ If U and V have u and v leaves, respectively, then T has $t = u + v$ leaves
 - ▶ By hypothesis U and V have $2u-1$ and $2v-1$ nodes, respectively
 - ▶ By the definition of the tree, T has $1 + (2u-1) + (2v-1) = 2(u+v) - 1 = 2t - 1$ nodes
 - ▶ So, $S(T)$ is true
- ▶ **Important**: we consider that the hypothesis is true for all the cases $\leq n$



Exercise 1 (TPC)

- ▶ Prove that for any natural number n , the sum of the first n squares is $n(n+1)(2n+1)/6$

Exercise 2 (TPC)

- Prove that for any natural number x greater or equal than 4, $2^x \geq x^2$

Exercise 3 (TPC)

► Prove that the sum of the first n perfect cubes is a perfect square.

► Examples:

► $1^3 + 2^3 + 3^3 = 36 = 6^2$

► $1^3 + 2^3 + 3^3 + 4^3 + 5^3 = 225 = 15^2$

► Solution:

► Induction using integers

► Statement: $\sum_{i=1}^n i^3 = a^2$

► Basis: $n=1$

► $a=1, 1^3 = a^2 = 1^2$

► Induction step

► $\sum_{i=1}^{n+1} i^3 = \sum_{i=1}^n i^3 + (n+1)^3 = a^2 + (n+1)^3 = b^2$ Which b ?

Exercise 3: Inventor's paradox

- ▶ Solution: reformulate the statement to prove in order to make it stronger
 - ▶ Instead of “one” perfect square, say which is “the” square: the sum of the numbers
 - ▶ Prove that exists one and we identify it, “invent” an extra restriction which serves to proceed with the proof → Inventor's paradox
 - ▶ New statement: $\sum_{i=1}^n i^3 = (\sum_{i=1}^n i)^2$
 - ▶ Induction step (basis: the same as before)
 - ▶ $\sum_{i=1}^{n+1} i^3 = (\sum_{i=1}^{n+1} i)^2$ objective
 - ▶ $\sum_{i=1}^n i^3 + (n+1)^3 = (\sum_{i=1}^n i + (n+1))^2$ algebra
 - ▶ $\sum_{i=1}^n i^3 + (n+1)^3 = (\sum_{i=1}^n i)^2 + 2(\sum_{i=1}^n i)(n+1) + (n+1)^2$ algebra
 - ▶ $(n+1)^3 = 2(\sum_{i=1}^n i)(n+1) + (n+1)^2$ hypothesis
 - ▶ $(n+1)^3 = 2(n/2)(n+1)(n+1) + (n+1)^2$ sum of the arithmetic series
 - ▶ $(n+1)^3 = n^3 + 3n^2 + 3n + 1$ Q.E.D.

Example – Balanced parenthesis

- ▶ Two definitions of balanced parenthesis:
 - ▶ Grammatically (EG)
 - ▶ The empty string ε is balanced
 - ▶ If w is balanced then “(w)” is balanced
 - ▶ If w and x are balanced then wx is balanced
 - ▶ By scanning (EV)
 - ▶ w is balanced if and only if (*iff*)
 - ▶ Has an equal number of (and)
 - ▶ Each prefix of w has at least as many (as)
- ▶ Theorem: a string of parenthesis is EG iff is EV
 - ▶ Bidirectional proof

Example – balanced parenthesis (proof)

► $EG \leftarrow EV$

► Proof by induction base on the length of the string w (+ conditional proof)

► Basis: $w = \varepsilon$, $|w| = 0$

► $w = \varepsilon \in EG$, by the first rule

► Induction step

► For $|w|=n+1$ there are two cases

► I) w does not have a non-empty prefix with the same number of (and)

► Then w must begin with (and finish with), i.e., $w = (x)$

► x must be EV $\rightarrow |x|$ even

► $|x| \leq n$, so, by hypothesis x is EG

► By the second rule, $w = (x)$ is also EG

► II) w has a non-empty prefix with the same number of (and)

► Then $w = xy$, in which x is the shorter of those prefixes and $y \neq \varepsilon$

► x and y are EV; by hypothesis, x and y are EG

► By the third rule w is EG

Example – balanced parenthesis (proof)

► EG \rightarrow EV

- Prove by induction based on the structure EG of the string w , i.e., in the number of applications of the rules of the EG definition (+ conditional proof)
- Basis: $w = \varepsilon$, $n = 1$, first rule of EG
 - $w = \varepsilon$ is EV (trivial)
- Induction step
 - For $n+1$ applications of EG rules there are two cases
 - I) w is EG because of the second rule, i.e., $w = (x)$ and x is EG
 - Then, by hypothesis, x is EV
 - As x has the same number of (and), (x) also has
 - As x does not have prefix with more) than (, (x) also does not
 - II) w is EG because of the third rule, i.e., $w = xy$ and x and y are EG
 - By hypothesis, x and y are EV (rigorously, the hypothesis is $EG \rightarrow EV$ for a number of rules $\leq n$)
 - As x and y have equal number of (and), w also has
 - If w had a prefix with more) than (, then or x would have such a prefix (in contradiction for being EV) or would have it x followed by a prefix of y (in contradiction to y being EV) (*proof by contradiction*)
- Q.E.D.

Summary

- ▶ Introduction to the Theory of Computations
- ▶ Introduction to finite automata
- ▶ Proof methods with emphasis on the proofs by the induction method (revision)