

Computational Logic

Pablo R. Fillottrani
Faculty of Computer Science
Free University of Bozen/Bolzano

Course Overview

- objectives
- syllabus
- projects
- Computational Logic: the science

Objectives

- to offer a comprehensive introduction of the **methods and techniques** in Computational Logic.
- to provide a solid background to logic from a computational perspective.

Syllabus

- Computational Logic: introduction (1 lecture)
- Propositional Logic (2 lectures)
- Applying propositional logic: SATPLAN (1 lecture)
- First Order Logic (3 lectures)
- Equational Logic (1 lecture)
- Nonmonotonic logics (3 lectures)
- Applying non-monotonic logics: Action Languages (1 lectures)
- Constraint (Logic) Programming (1 lecture)
- Abduction and Induction (1 lecture)

Assessment: projects (assignments) (50%) + oral exam (50%)

Projects

- assignments are **individual**
but you may work in groups
- one project per week of course
- each Wednesday (except of the first) you should handle the solutions for the exercises given in the previous week
- Monday labs will be used for reviewing the exercises and their solutions
- there are three kinds of exercises:
 - **written**: you must solve some theoretical problem
 - **programming**: you must develop some program in Prolog
 - **experience**: you must report your experience in using some existing software
- projects will be preliminary evaluated a week after their deadline
- the final global evaluation of assignments will be given at the end of the course

A Note on Slides

- slides, and additional material, will be available on the course web page
www.inf.unibz.it/fillottrani/CompLog2006.html
- but remind that by no means slides provide [a complete guide to the contents of the course](#)

Computational Logic: the science

- Description
- Applications
- History of Logic

Description

What is **Logic**?

- the ability to determine correct answers through a standardized process
- the study of formal inference
- a sequence of verified statements
- reasoning, as opposed to intuition
- the deduction of statements from a set of statements

Description

What is **Computational Logic**?

- it reflects all uses of logic to formalize computer science
- and all computer science problems handled by logic methods
- Logic is a very old science, and Computing a rather new one
- but they have both merged during the last century
- nowadays it is difficult to make a boundary between them

Applications

- Computational Logic can be seen in artificial intelligence, computational complexity, database systems, programming languages, automata and temporal logic, constraint programming, commonsense and knowledge representation, model checking, proof theory, modal logic, formal program specifications and development, etc.

History of Logic from Moshe Vardi, “A Brief History of Logic”, and Alan Robinson “Computational Logic: Memories of the Past and Challenges for the Future”

The First Age of Logic: Symbolic Logic (500 B.C. - 19th Century)

- logic was originally studied by the Sophist, who engaged in **formal debates**
- eventually, they sought to devise an objective **system of rules** to determine beyond any doubt who had won and argument
- so originally logic dealt with arguments in natural language used by humans
- natural language is **very ambiguous**
- natural language lead also to paradoxes
“This sentence is a lie”

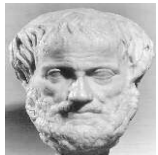
The First Age of Logic: Symbolic Logic (500 B.C. - 19th Century)

- other natural language **paradoxes**:

The Sophist paradox: A Sophist is sued for his tuition by the school that educated him. He says he should win, since if he loses then the school didn't educate him well enough, and doesn't deserve the money. The school argues that he must lose, since if he wins he was educated well enough, and therefore should pay for it.

The Surprise paradox: A logic professor announces to his class that there will be a test next week, but he won't tell which day. He promises: "When you get it, you will be surprised". The student deduce that the test can't be given on Friday: no surprise. But then it can't be given also on Thursday: no surprise. Similarly, it couldn't be held on Wednesday, Tuesday, or Monday. But on Tuesday the professor does give the test, and the students were very surprised.

The First Age of Logic: Symbolic Logic (500 B.C. - 19th Century)



- [Aristotle](#) (384-322 BC) was born in northern Greece.
- his analysis of inference patterns known as [syllogisms](#) was the only logical paradigm for two millenia
- the aim of Aristotle's logical treatises was to develop a [universal method of reasoning](#) by means of which it would be possible to learn everything there is to know about reality.
- thus, in the Categories he proposed a scheme for the description of particular things in terms of their properties, states, and activities.
- On Interpretation, Prior Analytics, and Posterior Analytics examined the nature of deductive inference, outlining the system of syllogistic reasoning from true propositions that later came to be known as propositional logic.

The First Age of Logic: Symbolic Logic (500 B.C. - 19th Century)



- [Gottfried Leibniz](#) (1646-1716), born in Leipzig, Germany, and



- [Blais Pascal](#) (1623-1662), born in Clermont, France
- both realized the syllogist analysis could be done [mechanically](#), by machines even feasible in the limited technology of their time
- but even now we have not quite completed their dreams...

The Second Age of Logic: Algebraic Logic (Mid to Late 19th Century)

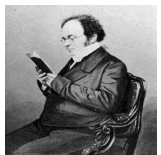


- **George Boole** (1815-1864), born in Lincoln, England, attempted in “The Mathematical Analysis of Logic” to formulate logic in terms of a mathematical language
- rules of inference were modeled after various laws for manipulating **algebraic expressions**
- the basis of this was the similarity between set union and intersection with addition and multiplication

The Second Age of Logic: Algebraic Logic (Mid to Late 19th Century)



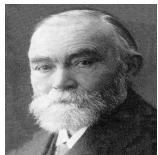
- Ernst Schröder (1841-1902), born in Mannheim, Germany, anticipated the importance of developing fast algorithms to decide logical problems



- Augustus De Morgan (1806-1871), born in Madura, India recognized the purely symbolic nature of algebra, and he was aware of the existence of algebras other than ordinary algebra
- once symbolic logic was matured, it became extremely useful in resolving many serious problems in mathematics

The Third Age of Logic: Mathematical Logic (Late 19th to Mid 20th Century)

- as mathematical proofs became more sophisticated, paradoxes began to show up as they did in natural language



- [Gottlob Frege](#) (1848-1925), born in Wismar, Germany,
- was one of the founders of modern symbolic logic putting forward the view that [mathematics is reducible to logic](#)
- Frege presented for the first time what we would recognize today as a logical system with negation, implication, universal quantification, essentially the idea of truth tables, etc (modern [first order logic](#))
- he wanted to have a precise way of stating results and of proving them, and indeed, he showed that all attempts to define "number" before him contained logical errors

The Third Age of Logic: Mathematical Logic (Late 19th to Mid 20th Century)

- Frege attempted to axiomatize arithmetic with an intuitive set of logical axioms, and intended to extend the theory to real numbers, but



- [Bertrand Russel](#) (1872-1970), born in Ravenscroft, Wales, showed that his system gave a contradiction, [Russell's paradox](#)
- Frege tried to modify his work, but none of his corrections worked out

The Third Age of Logic: Mathematical Logic (Late 19th to Mid 20th Century)

- Russell inherited Frege's idea of **logicism**, the theory that mathematics was in some important sense reducible to logic
- he wrote, with the help of Alfred Whitehead, "Principia Mathematica" with detailed derivations of many major theorems in set theory, finite and transfinite arithmetic, and elementary measure theory.
- but their axiomatization of set theory was not satisfactory. This has to wait for the work of Ernst Zermelo, Adolf Fraenkel and Thoralf Skolem in the early 1900's
- Russell's contributions also include the introduction of the theory of types, and his refining and popularizing of the first-order predicate calculus

The Third Age of Logic: Mathematical Logic (Late 19th to Mid 20th Century)



- **Georg Cantor** (1845-1918), born in St. Petersburg, Russia, used Frege's system to formally analyze the notion of **infinity**



- **Giuseppe Peano** (1858-1932), born in Cuneo, Italy, published in 1889 his famous axioms, called **Peano axioms**, which defined the natural numbers in terms of sets
- these and other logicist successes made the general believe that mathematics could be complete translated into logical axioms

The Third Age of Logic: Mathematical Logic (Late 19th to Mid 20th Century)



- **David Hilbert** (1862-1943), born in Kaliningrad, Russia, propose at the turn of the 20th Century a grand program to devise a **single formal procedure** that would derive all mathematical truth
- where he also presented his famous 23 Paris problems challenged (and still today challenge) mathematicians to solve them
- Hilbert's famous speech was delivered to the Second International Congress of Mathematicians in Paris. It was a speech full of optimism for mathematics in the coming century
- Hilbert's problems included the continuum hypothesis, the well ordering of the reals, Goldbach's conjecture, the transcendence of powers of algebraic numbers, the Riemann hypothesis, the extension of Dirichlet's principle and many more. Many of the problems were solved during this century, and each time one of the problems was solved it was a major event for mathematics.

The Third Age of Logic: Mathematical Logic (Late 19th to Mid 20th Century)



- **Alfred Tarski** (1902-1983), born in Warsaw, Poland introduced the concept of **semantics** that complemented Frege's syntactic system
- he introduced in 1929 the rigorous concept of interpretation, and of the denotation of an interpretation



- together with **Gerhard Gentzen** (1909-1945), he introduced the notion of logical consequence, and the semantic properties of sentences



- but then it was possible for **Kurt Gödel** (1906-1978), born in Brno, Czech Republic, to yield two results that proved devastating to Hilbert's program

The Third Age of Logic: Mathematical Logic (Late 19th to Mid 20th Century)

- in his [first incompleteness theorem](#) he showed that in any axiomatic mathematical system there are propositions that cannot be proved or disproved with the axioms of the system
- and in the [second incompleteness theorem](#) he showed that any formal system powerful enough to form statements about arithmetic cannot prove its own consistency
- this ended years of attempts to establish axioms which would put the whole of mathematics on an axiomatic basis
- Gödel's results were a landmark in 20th-century mathematics, showing that mathematics is not a finished object, as had been believed
- despite this results, logic continued to flourish, not as the universally accepted ultimate foundation of all mathematics, but simply as another branch of it

The Fourth Age of Logic: Logic in Computer Science



Jacques Herbrand

- Gödel, before his incompleteness results, and independently also Jacques Herbrand (1908-1931), born in Paris, France, proved the completeness of first order logic: a formula is derivable iff it is logical consequence
- Herbrand also established a link between quantification theory and sentential logic, which is important in that it gives a method to test a formula in quantification theory by successively testing formulas for sentential validity
- since testing for sentential validity is a mechanical process, Herbrand's theorem is today of major importance in software developed for theorem proving by computer
- in this way logic gave us characterization of computability, or solvability

The Fourth Age of Logic: Logic in Computer Science



- [Alan Turing](#) (1912-1954), born in London, England, and [Alonzo Church](#) (1903-1995), born in Washington D.C., USA, independently characterized the notion of computation
- and proved, roughly at the same time (1930), that there are some problems that no algorithm could ever solve
- thus computation has the same fate as logic...
- Turing also discussed the issue of [artificial intelligence](#), and proposed the famous Turing test
- Turing's work influenced John von Neumann in the design of the first electronic digital computers

The Fourth Age of Logic: Logic in Computer Science

- in 1954, Martin Davies carried out one of the earliest computational logic experiments by programming and running an algorithm for first order theory of integer addition
- it ran very slowly, but he could eventually prove that the sum of two even numbers is an even number
- Simon and Newell devised in 1956 the first heuristic theorem prover.
- in 1934 Gentzen gave the method of [sequent calculus](#), which were particularly useful for deriving metalogical decidability results
- in 1955 Evert Beth and Jaakko Hintikka described the [tableau method](#), an attractive and elegant version of a proof procedure
- in 1957, Davis and Hillary Putnam wrote their Herbrand-based proof procedure programs, based on the idea of systematically enumerating the Herbrand Universe of a proposed theorem

The Fourth Age of Logic: Logic in Computer Science

- in 1960, the Swedish logician Dag Prawitz rediscovered **unification** (originally presented by Herbrand)
- a work that led Alan Robinson to introduce in 1965 a new machine-oriented inference rule: **resolution**
- but again a negative result, in 1971, Cook demonstrated that the satisfiability problem in predicate calculus is **NP-complete**
- this was the first problem shown to belong to this class, a result that started the study of **complexity theory**
- so not only there are a lot of problems that computers can't solve, but also there are problems that computers can't solve with a reasonable amount of resources

The Fourth Age of Logic: Logic in Computer Science

- but resolution in a fragment of predicate logic was shown to run very efficiently
- the possibilities of resolution for computational linguistics were immediately seen by Alain Colmerauer, who was working in Grenoble
- he contacted Bob Kowalski, in Edinburgh, who developed a refinement of resolution ([SLD resolution](#))
- and this encounter was the birth of Prolog and [logic programming](#) by 1970
- Prolog shown in practice an ideal instrument for Artificial Intelligence research
- but pure declarative programming is not a complete panacea (Japan's Fifth Generation Project)

The Fourth Age of Logic: Logic in Computer Science

- by 1990 several groups continued to investigate unification-based inference engines, term-rewriting systems, proof finding strategies for [automated theorem proving](#)
- in 1996 McCune's OTTER theorem prover proves Robbins conjecture
- today: [applications](#) like functional equivalence of two chips, verification of hardware and software, eliminating redundancies and proving properties of group communication systems, designing layout of yellow pages, natural language processing, model generation, semantic web, ...

Propositional Logic

- syntax
- semantics
- normal forms
- propositional Horn logic
- calculi
- algorithms

- \mathcal{L} is a **language** of propositional logic
- the **alphabet** of \mathcal{L} is composed of
 - a finite or countably infinite set of **propositions** (nullary relation symbols): A, B, C, \dots (often called the **signature**)
 - the following **connectives**: $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
 - the **punctuation symbols**: $(,)$
- no meaning is attached to the symbols of the alphabet (this is the role of the "semantics")

- the **set of formulas** of \mathcal{L} is the minimal set satisfying:
 - each proposition is a formula
 - if F and G are formulas, then

$$\neg F, (F \wedge G), (F \vee G), (F \Rightarrow G), (F \Leftrightarrow G)$$

are formulas

- parenthesis are omitted whenever possible using the precedence:

$$\neg > \wedge > \vee > \Rightarrow > \Leftrightarrow$$

- if A is a proposition, then A and $\neg A$ are called **literals**
- A is called a **positive literal**; $\neg A$ is called a **negative literal**
- if L is a literal, \bar{L} is the **complementary literal** defined as $\neg A$ if $L = A$, or A if $L = \neg A$

- sometimes we want to prove that the set of formulas in \mathcal{L} has some **property**, or we want to define a function over it. Then the following theorems are useful
- **Theorem 1. [Structural Induction]** *Every formula of \mathcal{L} has a property \mathbb{E} provided that:*
 - *basis step: every proposition has the property \mathbb{E}*
 - *inductive steps: if F and G are formulas that exhibit property \mathbb{E} , then the formulas $\neg F$, $(F \wedge G)$, $(F \vee G)$, $(F \Rightarrow G)$, and $(F \Leftrightarrow G)$ also have property \mathbb{E}*
- **Exercise** prove that the number of opening parenthesis in a formula is equal to the number of closing parenthesis in it.

- **Theorem 2. [Structural Recursion]** *There is one, and only one function $h(\cdot)$ defined on the set of propositional formulas such that*
 - *basis step: $h(A)$ is specified for every proposition A*
 - *recursion steps: the values of $h(\neg F)$, $h(F \wedge G)$, $h(F \vee G)$, $h(F \Rightarrow G)$ and $h(F \Leftrightarrow G)$ are defined in terms of the values of $h(F)$ and $h(G)$*
- The following relation $n(\cdot)$ that counts the number of connectives in a formula is indeed a function over the propositional formulas

$$n(F) = \begin{cases} 0 & \text{if } F \text{ is a proposition} \\ n(G) + 1 & \text{if } F = \neg G \\ n(G) + n(H) + 1 & \text{if } F \text{ is of the form } G \wedge H, G \vee H, G \Rightarrow H \text{ or } G \Leftrightarrow H \end{cases}$$

- Informally, a **subformula** is the notion of a formula that is included in a formula
- Let $\mathcal{T}(F)$ be defined as the smallest set of formulas such that
 - $F \in \mathcal{T}(F)$
 - if $\neg G \in \mathcal{T}(F)$ then $G \in \mathcal{T}(F)$
 - if $G \wedge H, G \vee H, G \Rightarrow H$, or $G \Leftrightarrow H$ belongs to $\mathcal{T}(F)$, then $H, G \in \mathcal{T}(F)$
- With the help of theorem 2 we can be sure \mathcal{T} is a function in $\mathcal{L} \longrightarrow 2^{\mathcal{L}}$, that returns the set of all subformulas from a given formula

Propositional Logic - semantics

- semantics is introduced to assign **meaning** to formulas
- in propositional logic, each formula can be either **true** or **false**
- it is a **two-valued** logic; other logics introduce additional truth values
- an **interpretation** I is a total mapping from the set of propositions to the truth values
- any interpretation can be conveniently represented as the set of true propositions

Propositional Logic - semantics

- using theorem 2, an interpretation can be extended to the set of formulas
- $I \models F$ means I makes F true

$I \models A$ iff A is a proposition and $I(A) = \mathbf{true}$

$I \models \neg F$ iff $I \not\models F$

$I \models F \wedge G$ iff $I \models F$ and $I \models G$

$I \models F \vee G$ iff $I \models F$ or $I \models G$

$I \models F \Rightarrow G$ iff $I \not\models F$ or $I \models G$

$I \models F \Leftrightarrow G$ iff $I \models F \Rightarrow G$ and $I \models G \Rightarrow F$

- Example: $I_1 = \{A, C\}$, $I_2 = \{C, D\}$, $F = (A \vee B) \wedge (C \vee D)$ then $I_1 \models F$ but $I_2 \not\models F$
- the interpretation of connectives can be also done through **truth tables**

| F | G | $\neg F$ | $F \wedge G$ | $F \vee G$ | $F \Rightarrow G$ | $F \Leftrightarrow G$ |
|-------|-------|----------|--------------|------------|-------------------|-----------------------|
| true | true | false | true | true | true | true |
| true | false | false | false | true | false | false |
| false | true | true | false | true | true | false |
| false | false | true | false | false | true | true |

- if $I \models F$, then we say I is a **model** for F
- this notion can be extended to set of formulas
- F is **valid** or a **tautology** iff for all interpretations I it is true that $I \models F$
- F is **satisfiable** iff there exist an interpretation I such that $I \models F$
- F is **falsifiable** iff there exist an interpretation I such that $I \not\models F$
- F is **unsatisfiable** iff for all interpretation I it is true that $I \not\models F$
- examples: $A \vee \neg A$ is valid; $A \vee B$ is both satisfiable and falsifiable; $A \wedge \neg A$ is unsatisfiable
- any formula $F \wedge \neg F$ is called **contradiction** and often written \perp
- $F \vee \neg F$ is called **excluded middle** and often written \top

- a set of formulas \mathcal{F} **logically entails** a formula G (or G is a **logical consequence** of \mathcal{F}) if every model of \mathcal{F} is also a model of G
- it is written $\mathcal{F} \models G$
- example: $\{A, A \Rightarrow B\} \models B$, $\{A, A \Rightarrow B\} \models B \vee C$, $\{A, A \Rightarrow B\} \not\models C$
- in order to systematically determine if a formula follows from a set of formula, **truth tabling** can be used
- all possible combinations of truth values for every proposition should be considered :(
- example $A \Rightarrow B \wedge C \models A \Rightarrow C$

- the following theorems simplify the procedure
- **Theorem 3.** $\mathcal{F} \models A \Rightarrow B$ iff $\mathcal{F} \cup \{A\} \models B$
- example: in order to prove $\models A \wedge B \Rightarrow C$ it is sufficient to prove $A \wedge B \models C$
- **Theorem 4.** F is valid iff $\neg F$ is unsatisfiable
- example: to prove $\models A \wedge B \Rightarrow C$ (ie that the formula is valid) it is sufficient to prove that $\neg(A \wedge B \Rightarrow C)$ is unsatisfiable
- **Exercise:** prove the above theorems