

Computational Logic

Davide Martinenghi

Free University of Bozen-Bolzano

Spring 2010

Objectives

- ▶ To offer an introduction to the methods and techniques in Computational Logic
- ▶ To explore logic-based programming paradigms
- ▶ To present the area of databases as a special case where logical principles and methods are widely applied
- ▶ To provide a solid background in logic from a computational perspective
- ▶ To train fundamental mathematical skills such as giving formal definitions, formulating theorems, and proving or disproving formal statements

Syllabus in two parts

The course is divided in two parts:

- ▶ Computational Logic

- ▶ Taught by Davide Martinenghi

- ▶ <http://home.dei.polimi.it/martinen/courses/cl2010/CompLog2010.html>

- ▶ Textbook:

- ▶ [Logic for Computer Science: Foundations of Automatic Theorem Proving](#), J.H. Gallier, Harper & Row, 1986

- ▶ Available at <http://www.cis.upenn.edu/~jean/gbooks/logic.html>

- ▶ Foundations of Databases

- ▶ Taught by Werner Nutt

- ▶ Textbook:

- ▶ [Foundations of Databases](#): S. Abiteboul, R. Hull, and V. Vianu, Addison Wesley, 1995

Syllabus - Computational Logic part

- ▶ A history of logic
- ▶ Preliminaries: propositional logic and first-order logic
- ▶ Calculi:
 - ▶ natural deduction
 - ▶ sequent calculus
 - ▶ semantic tableaux
 - ▶ resolution
 - ▶ DPLL
 - ▶ binary decision diagrams
- ▶ Logic programming
- ▶ Negation and non-monotonicity:
 - ▶ closed-world assumption, negation-as-failure, completed database
 - ▶ circumscription
 - ▶ default logic
 - ▶ auto-epistemic logic

Exam and assignments

- ▶ The final mark will be based on assignments and on a written exam.
- ▶ Students will submit solutions for the assignments and present their solutions in class.
- ▶ Students who do not submit assignments will be assessed on the exam alone.
- ▶ For students who do submit assignments, the final mark will be based on both the exam mark and the exercise mark
 - ▶ Assessment: $\max\{0.7 \cdot EM + 0.3 \cdot AM, EM\}$
 - ▶ **EM** = exam mark, **AM** = assignment mark
- ▶ Assignments can be done in groups of one or two persons
- ▶ The exercises will be taken into account, **independently of when** the student takes the exam.
- ▶ Possible kinds of project assignments:
 - ▶ **written**: solve some theoretical problem
 - ▶ **programming**: develop some program (Prolog?)
 - ▶ **report**: analyze recent articles or compare different implementations or formalize in logic a challenging problem.

History of Logic

- ▶ First Age of Logic: Symbolic Logic (500 B.C. - 19th Century)
- ▶ Second Age of Logic: Algebraic Logic (Mid to Late 19th Century)
- ▶ Third Age of Logic: Mathematical Logic (Late 19th to Mid 20th Century)
- ▶ Fourth Age of Logic: Logic in Computer Science

from Moshe Vardi, "A Brief History of Logic", and

Alan Robinson, "Computational Logic: Memories of the Past and Challenges for the Future"

History of Logic

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

- ▶ Logic was originally studied by the Sophists, who engaged in **formal debates**
- ▶ Eventually, they tried to devise an objective **system of rules** to determine beyond any doubt who had won and argument
- ▶ Originally, logic dealt with arguments in natural language used by humans
- ▶ But natural language is **very ambiguous**:
 - ▶ “Eric does **not** believe that Mary can pass **any** test”
 - ▶ “I **only** borrowed your car”
- ▶ and prone to **paradoxes** (like many other formalisms...):
 - ▶ “This sentence is a lie”

History of Logic

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

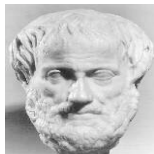
- ▶ Digression: more natural language **paradoxes**

The Sophist paradox: A Sophist is sued for his tuition by the school that educated him. He claims 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 teacher 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 students deduce that the test can't be given on Friday: no surprise. But then it can't be given on Thursday either: 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!

History of Logic

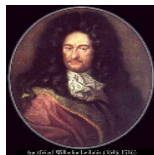
First Age of Logic: Symbolic Logic (500 B.C. - 19th Century)



- ▶ Aristotle (384-322 BC) was born in northern Greece.
- ▶ His analysis of inference patterns known as **sylogisms** was the only logical paradigm for two millennia
- ▶ 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, **Categories** proposes a scheme for the description of particular things in terms of their properties, states, and activities.
- ▶ **On Interpretation**, **Prior Analytics**, and **Posterior Analytics** examine the nature of deductive inference, outlining the system of syllogistic reasoning from true propositions that later came to be known as **propositional logic**.

History of Logic

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



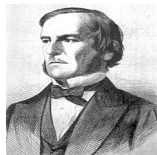
- ▶ Gottfried Leibniz (1646-1716), born in Leipzig, Germany,



- ▶ Blaise Pascal (1623-1662), born in Clermont-Ferrand, France
- ▶ Both realized that syllogist analysis could be done **mechanically**, by machines even feasible in the limited technology of their time
- ▶ Leibniz's **calculus ratiocinator** follows the idea that “when there are disputes among persons, we can simply say: **calculemus**, without further ado, to see who is right.”
- ▶ But even now we have not quite completed their dreams...

History of Logic

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

History of Logic

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

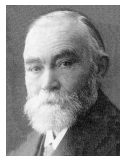


- ▶ **Augustus De Morgan** (1806-1871), born in Madura, India recognized the purely **symbolic nature** of algebra
- ▶ Once symbolic logic was mature, it became extremely useful in resolving many serious problems in mathematics

History of Logic

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
- ▶ He 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 proving them. Indeed, he showed that all attempts to define “number” before him contained logical errors

History of Logic

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 Russell (1872-1970), born in Trellech, UK, showed that his system gave a contradiction, [Russell's paradox](#) (aka [Russell's antinomy](#))
- ▶ Frege tried to modify his work, but none of his corrections worked out

History of Logic

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

- ▶ Russell inherited Frege's idea of **logicism**, the view 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 had to wait for the work of Ernst Zermelo, Adolf Fraenkel and Thoralf Skolem in the early 1900s
- ▶ Russell's contributions also include the introduction of the theory of types, and his refining and popularizing of first-order predicate calculus

History of Logic

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's successes made the general belief that mathematics could be completely translated into logical axioms

History of Logic

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



- ▶ **David Hilbert** (1862-1943), born in Kaliningrad, Russia, proposed at the turn of the 20th Century a grand program to devise a **single formal procedure** that would derive all mathematical truths
- ▶ He also presented his famous 23 Paris problems and challenged mathematicians to solve them (some are still unsolved)
- ▶ Hilbert's famous speech was delivered to the 2nd Int'l 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, and many more. Many of the problems were solved during the 20th century.

History of Logic

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

History of Logic

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



- ▶ 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
- ▶ In his **first incompleteness theorem** he showed that in any formal system powerful enough to form statements about what it can prove, there are statements that the system can express but cannot prove
- ▶ In his **second incompleteness theorem** he showed that any formal system powerful enough to form statements about arithmetics cannot prove its own consistency
- ▶ This ended years of attempts to establish axioms which would put the whole of mathematics on an axiomatic basis

History of Logic

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

- ▶ Gödel's results were a landmark in 20th-century mathematics
- ▶ Despite these results, logic continued to flourish, not as the universally accepted ultimate foundation of all mathematics, but simply as another branch of it

History of Logic

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

- ▶ Gödel, before his incompleteness results, and independently also



Jacques Herbrand (1908-1931), born in Paris, France, proved (both in their doctoral thesis) the **completeness** of first-order logic

- ▶ Herbrand's theorem put the basis for theorem proving by computers as a mechanical process

History of Logic

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



- ▶ Alan Turing, left, (1912-1954), born London, UK, and Alonzo Church (1903-1995), right, born in Washington D.C., USA, independently characterized the notion of computation
- ▶ They 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

History of Logic

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 from first-order theory of integer addition
- ▶ It ran very slowly, but it could eventually prove that the sum of two even numbers is an even number
- ▶ In 1934 Gentzen gave the method of [sequent calculus](#), which was particularly useful for deriving meta-logical decidability results
- ▶ In 1955 Evert Beth and Jaakko Hintikka described the [tableau method](#), an attractive and elegant version of a proof procedure
- ▶ In 1957, Martin Davis and Hilary Putnam wrote their Herbrand-based proof procedure programs, based on the idea of systematically enumerating the Herbrand Universe of a proposed theorem

History of Logic

Fourth Age of Logic: Logic in Computer Science

- ▶ In 1960, the Swedish logician Dag Prawitz rediscovered **unification** (originally presented by Herbrand)
- ▶ This work led Alan Robinson to introduce in 1965 a new machine-oriented inference rule: **resolution**
- ▶ 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** in 1972
- ▶ Prolog proved in practice an ideal instrument for Artificial Intelligence research

History of Logic

Fourth Age of Logic: Logic in Computer Science

- ▶ But pure declarative programming is not a complete panacea (Japan's Fifth Generation Project)
 - ▶ A bold 10-year effort by Japan to seize the lead in computer technology is fizzling to a close having failed to meet many of its ambitious goals or to produce technology that Japan's computer industry wanted. After spending \$400 million on its widely heralded Fifth Generation computer project, the Japanese Government said this week that it was willing to give away the software developed by the project to anyone who wanted it, even foreigners.
- ▶ In the 1990s several groups continued to investigate unification-based inference engines, term-rewriting systems, proof-finding strategies for **automated theorem proving**
- ▶ Now the field has moved from "logic programming" to "**computational logic**", to mean a narrower interest within a broader framework, characterized by some sort of a connection between computing and logic

Propositional Logic - syntax

- ▶ \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, ...**
 - ▶ 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”)

Propositional Logic - syntax

- ▶ The **set of formulas** of \mathcal{L} is the minimal set such that:
 - ▶ Each proposition is a formula
 - ▶ If **F** and **G** are formulas, then $(\neg F)$, $(F \wedge G)$, $(F \vee G)$, $(F \rightarrow G)$, and $(F \leftrightarrow G)$ are formulas
- ▶ Parentheses 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 $L = \neg A$

Propositional Logic - syntax

- ▶ 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 (Structural Induction)

Every formula of \mathcal{L} has a property \mathbb{E} provided that:

- ▶ *basis: every proposition has 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}*
- ▶ By structural induction you can, e.g., prove that, in a formula, no parenthesis symbol ((or)) is ever immediately followed by the other parenthesis symbol) or ().

Propositional Logic - syntax

Theorem (Structural Recursion)

Function $h(\cdot)$ on the set of propositional formulas is univocally determined if

- ▶ basis: $h(A)$ 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

$$n(F) = \begin{cases} 0 & \text{if } F \text{ is a proposition} \\ n(G) + 1 & \text{if } F = \neg G \\ n(G) + n(H) + 1 & \text{if } F \text{ of form } G \circ H, \circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\} \end{cases}$$

Propositional Logic - syntax

- ▶ Informally, a **subformula** is the notion of a formula that is included in a formula
- ▶ Let $\mathcal{I}(\mathbf{F})$ be defined as the smallest set of formulas such that
 - ▶ $\mathbf{F} \in \mathcal{I}(\mathbf{F})$
 - ▶ if $\neg \mathbf{G} \in \mathcal{I}(\mathbf{F})$ then $\mathbf{G} \in \mathcal{I}(\mathbf{F})$
 - ▶ if $\mathbf{G} \wedge \mathbf{H}$, $\mathbf{G} \vee \mathbf{H}$, $\mathbf{G} \rightarrow \mathbf{H}$, or $\mathbf{G} \leftrightarrow \mathbf{H}$ belongs to $\mathcal{I}(\mathbf{F})$, then $\mathbf{H}, \mathbf{G} \in \mathcal{I}(\mathbf{F})$
- ▶ With the help of structural recursion we can be sure \mathcal{I} 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 structural recursion, 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 $I(A) = \text{true}$

$I \models \neg F$ 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 $I_2 \not\models F$

Propositional Logic - semantics

- ▶ The interpretation of connectives can be also 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 |

Propositional Logic - semantics

- ▶ If $I \models F$, then we say I is a **model** for F . This notion can be extended to sets of formulas
- ▶ F is **valid** or a **tautology** iff for all interpretations I it is true that $I \models F$
 - ▶ In this case one can also write $\models F$
- ▶ 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 $I \not\models F$
- ▶ F is **contingent** iff it is both satisfiable and falsifiable
- ▶ 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

Propositional Logic - semantics

- ▶ 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 whether a formula follows from a set of formulas, **truth tabling** can be used.
 - ▶ All possible combinations of truth values for every proposition should be considered

Propositional Logic - semantics

- ▶ The following theorems simplify the procedure

Theorem (Deduction theorem)

$$\mathcal{F} \models A \rightarrow B \text{ iff } \mathcal{F} \cup \{A\} \models B$$

- ▶ Example: in order to prove $A \wedge B \models C$ it is sufficient to prove $\models A \wedge B \rightarrow C$

Theorem

F is valid iff $\neg F$ is unsatisfiable

- ▶ Example: to prove $\models A \wedge B \rightarrow C$ it is sufficient to prove $\neg(A \wedge B \rightarrow C)$ is unsatisfiable.

Propositional Logic - semantics

- ▶ However, the exponential blowup in the computation procedure is unavoidable

Theorem (Cook 1971)

SAT (the problem of deciding if a formula in \mathcal{L} is satisfiable) is NP-complete.

- ▶ The calculi that will be introduced later show several possible ways to implement a procedure for SAT.
- ▶ They work on restricted sets of formulas, so **normal forms** should be introduced

Propositional Logic - normal forms

- ▶ Two formulas **F** and **G** are (semantically) equivalent iff **F** \models **G** and **G** \models **F**
- ▶ It is written **F** \equiv **G**

Propositional Logic - normal forms

► Some named equivalences:

| | |
|---|-------------------------|
| $(F \wedge F) \equiv F$ | \wedge -idempotency |
| $(F \vee F) \equiv F$ | \vee -idempotency |
| $(F \wedge G) \equiv (G \wedge F)$ | \wedge -commutativity |
| $(F \vee G) \equiv (G \vee F)$ | \vee -commutativity |
| $(F \wedge (G \wedge H)) \equiv ((F \wedge G) \wedge H)$ | \wedge -associativity |
| $(F \vee (G \vee H)) \equiv ((F \vee G) \vee H)$ | \vee -associativity |
| $((F \wedge G) \vee F) \equiv F$ | absorption |
| $((F \vee G) \wedge F) \equiv F$ | absorption |
| $(F \wedge (G \vee H)) \equiv ((F \wedge G) \vee (F \wedge H))$ | distributivity |
| $(F \vee (G \wedge H)) \equiv ((F \vee G) \wedge (F \vee H))$ | distributivity |
| $(\neg(\neg F)) \equiv F$ | double negation |
| $(\neg(F \wedge G)) \equiv (\neg F \vee \neg G)$ | de Morgan's law |
| $(\neg(F \vee G)) \equiv (\neg F \wedge \neg G)$ | de Morgan's law |
| $(F \leftrightarrow G) \equiv (F \rightarrow G) \wedge (G \rightarrow F)$ | equivalence |
| $(F \rightarrow G) \equiv (\neg F \vee G)$ | material implication |
| $(F \rightarrow G) \equiv (\neg G \rightarrow \neg F)$ | contraposition |

Propositional Logic - normal forms

- ▶ Some named equivalences (without redundant parentheses):

| | |
|---|-------------------------|
| $F \wedge F \equiv F$ | \wedge -idempotency |
| $F \vee F \equiv F$ | \vee -idempotency |
| $F \wedge G \equiv G \wedge F$ | \wedge -commutativity |
| $F \vee G \equiv G \vee F$ | \vee -commutativity |
| $F \wedge (G \wedge H) \equiv (F \wedge G) \wedge H$ | \wedge -associativity |
| $F \vee (G \vee H) \equiv (F \vee G) \vee H$ | \vee -associativity |
| $F \wedge G \vee F \equiv F$ | absorption |
| $(F \vee G) \wedge F \equiv F$ | absorption |
| $F \wedge (G \vee H) \equiv F \wedge G \vee F \wedge H$ | distributivity |
| $F \vee G \wedge H \equiv (F \vee G) \wedge (F \vee H)$ | distributivity |
| $\neg\neg F \equiv F$ | double negation |
| $\neg(F \wedge G) \equiv \neg F \vee \neg G$ | de Morgan's law |
| $\neg(F \vee G) \equiv \neg F \wedge \neg G$ | de Morgan's law |
| $F \leftrightarrow G \equiv (F \rightarrow G) \wedge (G \rightarrow F)$ | equivalence |
| $F \rightarrow G \equiv \neg F \vee G$ | material implication |
| $F \rightarrow G \equiv \neg G \rightarrow \neg F$ | contraposition |

Propositional Logic - normal forms

- ▶ Whenever we replace a subformula G of F by a formula H , the resulting formula is indicated $F[G \setminus H]$

Theorem (Substitution Theorem)

If G is a subformula of F , and $G \equiv H$ then $F \equiv F[G \setminus H]$

- ▶ Following the given equivalences, not all connectives are needed
- ▶ A set of connectives is said to be **functionally complete** iff any propositional formula can be transformed into a semantically equivalent one which only contains connectives in the set
- ▶ Example: $\{\neg, \wedge\}$ is functionally complete
- ▶ Other boolean connectives can be defined that, alone, are functionally complete: **nand** and **nor**

Propositional Logic - normal forms

- ▶ The Substitution Theorem and the given equivalences can be used to introduce so-called **normal forms**
- ▶ A formula is in **negation normal form** iff it is built only by literals, conjunctions and disjunctions
- ▶ A formula is in **conjunctive normal form** (CNF) iff it has the form $C_1 \wedge C_2 \wedge \dots \wedge C_n$ where each C_i is a disjunction of literals
- ▶ A formula is in **disjunctive normal form** (DNF) iff it has the form $D_1 \vee D_2 \vee \dots \vee D_n$ where each D_i is a conjunction of literals
- ▶ $C_1 \wedge C_2 \wedge \dots \wedge C_n \equiv C_1 \wedge C_2 \wedge \dots \wedge C_n \wedge \top$, so we can say that \top is in CNF taking $n = 0$
- ▶ $D_1 \vee D_2 \vee \dots \vee D_n \equiv D_1 \vee D_2 \vee \dots \vee D_n \vee \perp$, so we can say that \perp is in DNF taking $n = 0$
- ▶ The C_i 's are called **clauses**; the D_i 's are **dual clauses**; set notation is generally used
- ▶ **Factoring** in clauses consists in eliminating duplicates of predicates in the same clause
- ▶ Equivalent to using idempotency laws (implicit in set notation)

Propositional Logic - normal forms

Problem: transform a propositional formula into CNF

Input: a propositional formula F

Output: a propositional formula G in CNF such that $F \equiv G$

- ▶ Eliminate all \leftrightarrow connectives using the equivalence law
- ▶ Eliminate all \rightarrow connectives using the implication law
- ▶ Eliminate all \neg connectives (except those in front of propositions) using de Morgan's laws and the double negation law
- ▶ Distribute all disjunctions over conjunctions using distributivity, commutativity and associativity
- ▶ This is a **correct** and **terminating** algorithm
- ▶ Example: $(D \vee \neg A \vee B) \wedge (D \vee \neg A \vee \neg C)$ is the CNF of $(A \wedge (B \rightarrow C)) \rightarrow D$
- ▶ However, the size of G can be **exponential** in the size of F
 - ▶ take, e.g., $(A \wedge B) \vee (C \wedge D) \vee (E \wedge F)$

Propositional Horn logic

- ▶ Clauses that contain only one literal are called **unit clauses**
- ▶ Clauses that contain at most one positive literal are called **Horn clauses**
- ▶ Horn clauses are of the form $\{H, \neg B_1, \dots, \neg B_n\}$, usually written as
$$H \leftarrow B_1, \dots, B_n$$
- ▶ **H** is called the **head** and the rhs is called the **body** of the clause
- ▶ A Horn clause with only negative literals is called a **goal clause**

$$\leftarrow B_1, \dots, B_n$$

(the head is implicitly \perp)

- ▶ A Horn clause with a positive literal is called a **definite clause**; a **definite logic program** is a set of definite clauses
- ▶ A **fact** is a clause with only a positive literal
- ▶ The **empty clause** \leftarrow is the same as $\perp \leftarrow \top$, i.e., \perp
- ▶ These clauses have both a **declarative meaning** and a **procedural meaning** (deduction as a procedure execution)

Propositional Horn logic

Theoretical properties

Theorem

Let Π be a definite logic program. If I_1 and I_2 are models of Π , then so is $I_1 \cap I_2$.

Theorem

Let Π be a definite logic program, and $M_\Pi = \{A : \Pi \models A\}$ (A an atom). Then M_Π is a model of Π , and it is contained in every other model of Π .

- ▶ M_Π is called the least model of the program Π

Propositional Horn logic

- ▶ The least model of a program Π can also be computed by means of an **immediate consequence operator** T_Π that maps interpretations onto interpretations:

$$T_\Pi(I) = I \cup \{H : H \leftarrow B_1, \dots, B_n \in \Pi \text{ and } \{B_i\}_{i=1}^n \subseteq I\}$$

- ▶ T_Π is monotonic on the lattice of interpretations, then we can define the sequence

$$T_\Pi \uparrow 0 = \emptyset$$

$$T_\Pi \uparrow (n + 1) = T_\Pi(T_\Pi \uparrow n)$$

- ▶ It can be proved that this sequence of interpretations is **monotonically increasing**, and (Knaster-Tarski theorem) has a **least fixpoint** $\text{lfp}(T_\Pi)$

Propositional Horn logic

Theorem

Let Π be a definite logic program. Then $M_{\Pi} = \text{lfp}(T_{\Pi})$

- ▶ For each finite propositional logic program the least fixpoint can be reached in a **finite** number of steps
- ▶ So in order to check whether a formula **F** is a logical consequence of a propositional definite logic program Π , just check **$\text{lfp}(T_{\Pi})$**
- ▶ This procedure can be carried out in **polynomial** time on the lengths of the formulas
- ▶ Let HORNSAT be the problem of deciding if a set of Horn clauses is satisfiable

Theorem

HORNSAT is in P

- ▶ Then it is much easier to compute consequences of Horn clauses, than to compute consequences of arbitrary sets of formulas

Propositional Logic - calculi

- ▶ Having defined a logic, we are now interested in knowing whether the logical consequences can be mechanically computed
- ▶ A **calculus** consists of a set of **axioms** and a set of **inference rules** that produce the logical consequences in a logic
- ▶ These elements define a **derivability relation** between a set of formulas \mathcal{F} and a formula G . We have

$$\mathcal{F} \vdash G$$

if G can be obtained from \mathcal{F} by applying only inference rules and axioms.

- ▶ Ideally, the derivability relation should be **sound** (i.e., if $\mathcal{F} \vdash G$ then $\mathcal{F} \models G$) and **complete** (i.e., if $\mathcal{F} \models G$ then $\mathcal{F} \vdash G$)
- ▶ If a formula F can be derived in a theory \mathcal{F} using the axioms and inference rules of a calculus, then we say that F is a **theorem**

Propositional Logic - calculi

- ▶ Additionally, a calculus can be classified as:
 - ▶ **negative** if it leads to a refutation proof, i.e., a proof of the empty clause starting from the negation of the thesis to prove
 - ▶ **positive** otherwise
- ▶ Generally, negative calculi have computational advantages over positive ones, since they have a built-in sense of direction (goal oriented).
- ▶ Some well-known calculi:
 - ▶ natural deduction
 - ▶ sequent calculus
 - ▶ DPLL procedure
 - ▶ resolution
 - ▶ semantic tableaux
 - ▶ binary decision diagrams

Propositional Logic - natural deduction

- ▶ **Natural Deduction** was introduced by Gentzen in 1935. He tried to formalize a logical reasoning in mathematics
- ▶ It is an example of a positive calculus
- ▶ The idea is that when you are stating a proof, you make some assumptions, then draw conclusions, and finally, discharge the assumptions to obtain assumption-free results
- ▶ It consists of only one axiom, \top , and several inference rules to produce proofs

Propositional Logic - natural deduction

Natural Deduction Inference Rules

constant rules $\frac{\perp}{F}$ ex falso quod libet

$\frac{F \quad \neg F}{\perp}$ contradiction

Propositional Logic - natural deduction

Natural Deduction Inference Rules

negation introduction rule

$$\frac{\begin{array}{c} F \\ \vdots \\ \perp \end{array}}{\neg F}$$

negation elimination rule

$$\frac{\begin{array}{c} \neg F \\ \vdots \\ \perp \end{array}}{F}$$

- ▶ These two rules formalize the principle of **reductio ad absurdum**

Propositional Logic - natural deduction

Natural Deduction Inference Rules

and-introduction rule

$$\frac{F \quad G}{F \wedge G}$$

and-elimination rules

$$\frac{F \wedge G}{F}$$

$$\frac{F \wedge G}{G}$$

Propositional Logic - natural deduction

Natural Deduction Inference Rules

or-introduction rules

$$\frac{\begin{array}{c} \neg F \\ \vdots \\ G \end{array}}{F \vee G}$$

$$\frac{\begin{array}{c} \neg G \\ \vdots \\ F \end{array}}{F \vee G}$$

or-elimination rules

$$\frac{\neg F \quad F \vee G}{G}$$

$$\frac{\neg G \quad F \vee G}{F}$$

Propositional Logic - natural deduction

Natural Deduction Inference Rules

implication introduction rules

$$\frac{\begin{array}{c} \mathbf{F} \\ \vdots \\ \mathbf{G} \end{array}}{\mathbf{F} \rightarrow \mathbf{G}}$$

contraposition

$$\frac{\begin{array}{c} \neg \mathbf{G} \\ \vdots \\ \neg \mathbf{F} \end{array}}{\mathbf{F} \rightarrow \mathbf{G}}$$

implication elimination rules

$$\frac{\mathbf{F} \quad \mathbf{F} \rightarrow \mathbf{G}}{\mathbf{G}}$$

modus ponens

$$\frac{\neg \mathbf{G} \quad \mathbf{F} \rightarrow \mathbf{G}}{\neg \mathbf{F}}$$

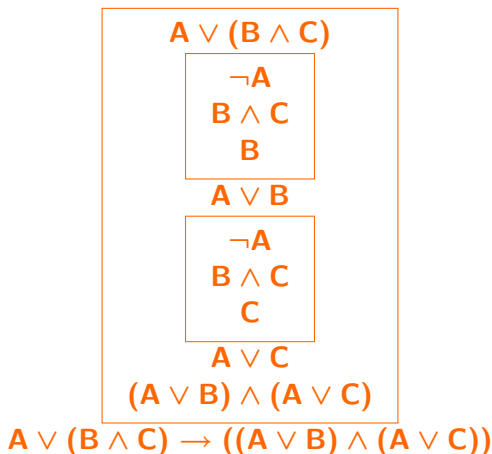
modus tollens

Propositional Logic - natural deduction

- ▶ A **deduction** is a sequence of formulas, possibly included in open or closed boxes, such that each element is either
 - ▶ the axiom **T**
 - ▶ a formula that follows from earlier elements in the currently open boxes by applying rules of inferences (this may close boxes)
 - ▶ another formula, called **assumption**, which opens a new box
- ▶ A **proof** is a deduction in which all boxes are closed; a **proof for F** is a proof in which **F** is the last formula in the deduction

Propositional Logic - natural deduction

Example of a proof for $A \vee (B \wedge C) \rightarrow ((A \vee B) \wedge (A \vee C))$:



Propositional Logic - natural deduction

- ▶ Repeated generation of the same deduction can be avoided by using lemmas
- ▶ Example:



can be stated as a lemma $\frac{\neg\neg F}{F}$

Propositional Logic - natural deduction

- ▶ Other useful lemmas

$$\begin{array}{c}
 \neg(F \wedge G) \\
 F \\
 \boxed{\begin{array}{c} G \\ (F \wedge G) \\ \perp \end{array}} \\
 \neg G
 \end{array}$$

$$\begin{array}{c}
 \neg(F \wedge G) \\
 G \\
 \boxed{\begin{array}{c} F \\ (F \wedge G) \\ \perp \end{array}} \\
 \neg F
 \end{array}$$

$$\begin{array}{c}
 \neg(F \vee G) \\
 \boxed{\begin{array}{c} \neg(\neg F \wedge \neg G) \\ \boxed{\begin{array}{c} \neg G \\ \neg\neg F \\ F \end{array}} \\ (F \vee G) \\ \perp \end{array}} \\
 (\neg F \wedge \neg G)
 \end{array}$$

$$\frac{\neg(F \wedge G) \quad F}{\neg G}$$

$$\frac{\neg(F \wedge G) \quad G}{\neg F}$$

$$\frac{\neg(F \vee G)}{(\neg F \wedge \neg G)}$$

- ▶ Lemmas can help to shorten proofs; they can also enlarge the search space in the process of searching for proofs

Propositional Logic - natural deduction

Theorem

The propositional calculus of natural deduction is sound and complete

- ▶ Natural deduction calculus is easy to understand by humans, but difficult to generate by machines
- ▶ Neither bottom-up nor top-down strategies are good in implementations