# Formal Specification and Verification

1. Classical Logic

27.10.2016

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

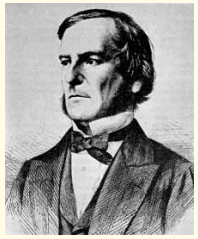
# Mathematical foundations:

## Classical first-order logic

# Mathematical foundations:

## Classical first-order logic

**George Boole** (1815-1864)



George Boole is best known as the author of "The Laws of Thought" (1854). He is the inventor of the prototype of what is now called Boolean logic. Because of this Boole is also regarded as a founder of the field of computer science.

# Mathematical foundations:

## Classical first-order logic

**David Hilbert** (1862-1943)

David Hilbert is recognized as one of the most influential and universal mathematicians of the 19th and early 20th centuries.

A famous example of his leadership in mathematics is his 1900 presentation of a collection of 23 problems that set the course for much of the mathematical research of the 20th century.

Hilbert is known as one of the founders of proof theory and mathematical logic.

# Mathematical foundations:

## Classical first-order logic

**Gottlob Frege** (1848-1925)

Gottlob Frege is considered to be one of the founders of modern logic and made major contributions to the foundations of mathematics.

Frege invented axiomatic predicate logic, in large part thanks to the fact that he introduced and used quantified variables.

# Formal logics

**A formal logic consists of:**

- **Syntax:** a formal language (formula expressing facts)
- **Semantics:** to define the meaning of the language, that is which facts are valid)
- **Deductive system:** made of axioms and inference rules to formaly derive theorems, that is facts that are provable

# Questions about formal logics

**The main questions about a formal logic are:**

- The soundness of the deductive system: no provable formula is invalid
- The completeness of the deductive system: all valid formulae are provable

# Part 1: Propositional classical logic

**Literature** (also for first-order logic)

Schöning: Logik für Informatiker, Spektrum

Fitting: First-Order Logic and Automated Theorem Proving, Springer

# 1.1 Syntax

- propositional variables
- logical symbols

  $\Rightarrow$ Boolean combinations

# Propositional Variables

Let $\Pi$ be a set of propositional variables.

We use letters $P$, $Q$, $R$, $S$, to denote propositional variables.

# Propositional Formulas

$F_\Pi$ is the set of propositional formulas over $\Pi$ defined as follows:

$$
\begin{array}{lcll}
F, G, H & ::= & \bot & \text{(falsum)} \\
& | & \top & \text{(verum)} \\
& | & P, \quad P \in \Pi & \text{(atomic formula)} \\
& | & \neg F & \text{(negation)} \\
& | & (F \wedge G) & \text{(conjunction)} \\
& | & (F \vee G) & \text{(disjunction)} \\
& | & (F \rightarrow G) & \text{(implication)} \\
& | & (F \leftrightarrow G) & \text{(equivalence)}
\end{array}
$$

# Notational Conventions

- We omit brackets according to the following rules:
  - $\neg \quad >_p \quad \wedge \quad >_p \quad \vee \quad >_p \quad \rightarrow \quad >_p \quad \leftrightarrow$ (binding precedences
  - $\vee$ and $\wedge$ are associative and commutative

# 1.2 Semantics

In classical logic (dating back to Aristoteles) there are "only" two truth values "true" and "false" which we shall denote, respectively, by 1 and 0.

There are multi-valued logics having more than two truth values.

# Valuations

A propositional variable has no intrinsic meaning. The meaning of a propositional variable has to be defined by a valuation.

A $\Pi$-valuation is a map

$$\mathcal{A} : \Pi \to \{0, 1\}.$$

where $\{0, 1\}$ is the set of truth values.

# Truth Value of a Formula in $\mathcal{A}$

Given a $\Pi$-valuation $\mathcal{A}$, the function $\mathcal{A}^*$ : $\Sigma$-formulas $\rightarrow \{0, 1\}$ is defined inductively over the structure of $F$ as follows:

$$\mathcal{A}^*(\bot) = 0$$

$$\mathcal{A}^*(\top) = 1$$

$$\mathcal{A}^*(P) = \mathcal{A}(P)$$

$$\mathcal{A}^*(\neg F) = 1 - \mathcal{A}^*(F)$$

$$\mathcal{A}^*(F \rho G) = \mathrm{B}_\rho(\mathcal{A}^*(F), \mathcal{A}^*(G))$$

with $\mathrm{B}_\rho$ the Boolean function associated with $\rho$

For simplicity, we write $\mathcal{A}$ instead of $\mathcal{A}^*$.

# Truth Value of a Formula in $\mathcal{A}$

**Example:** Let's evaluate the formula

$$(P \to Q) \wedge (P \wedge Q \to R) \to (P \to R)$$

w.r.t. the valuation $\mathcal{A}$ with

$$\mathcal{A}(P) = 1, \mathcal{A}(Q) = 0, \mathcal{A}(R) = 1$$

(On the blackboard)

# 1.3 Models, Validity, and Satisfiability

$F$ is valid in $\mathcal{A}$ ($\mathcal{A}$ is a model of $F$; $F$ holds under $\mathcal{A}$):

$$\mathcal{A} \models F :\Leftrightarrow \mathcal{A}(F) = 1$$

$F$ is valid (or is a tautology):

$$\models F :\Leftrightarrow \mathcal{A} \models F \text{ for all } \Pi\text{-valuations } \mathcal{A}$$

$F$ is called satisfiable iff there exists an $\mathcal{A}$ such that $\mathcal{A} \models F$.

Otherwise $F$ is called unsatisfiable (or contradictory).

# 1.3 Models, Validity, and Satisfiability

**Examples:**

$F \to F$ and $F \vee \neg F$ are valid for all formulae $F$.

Obviously, every valid formula is also satisfiable

$F \wedge \neg F$ is unsatisfiable

The formula $P$ is satisfiable, but not valid

# Entailment and Equivalence

$F$ entails (implies) $G$ (or $G$ is a consequence of $F$), written $F \models G$, if for all $\Pi$-valuations $\mathcal{A}$, whenever $\mathcal{A} \models F$ then $\mathcal{A} \models G$.

$F$ and $G$ are called equivalent if for all $\Pi$-valuations $\mathcal{A}$ we have $\mathcal{A} \models F \Leftrightarrow \mathcal{A} \models G$.

**Proposition 1.1:**

$F$ entails $G$ iff $(F \rightarrow G)$ is valid

**Proposition 1.2:**

$F$ and $G$ are equivalent iff $(F \leftrightarrow G)$ is valid.

# Entailment and Equivalence

Extension to sets of formulas $N$ in the "natural way", e.g., $N \models F$ if for all $\Pi$-valuations $\mathcal{A}$: if $\mathcal{A} \models G$ for all $G \in N$, then $\mathcal{A} \models F$.

A set $N$ of formulae is satisfiable iff there exists an $\mathcal{A}$ such that $\mathcal{A} \models F$ for all $F \in N$.
Otherwise $N$ is called unsatisfiable (or contradictory).

Thus, $N$ is unsatisfiable iff $N \models \bot$.

# Validity vs. Unsatisfiability

Validity and unsatisfiability are just two sides of the same medal as explained by the following proposition.

**Proposition 1.3:**

$$F \text{ valid} \quad \Leftrightarrow \quad \neg F \text{ unsatisfiable}$$
$$N \models F \quad \Leftrightarrow \quad N \cup \{\neg F\} \text{ unsatisfiable}$$

Hence in order to design a theorem prover (validity/entailment checker) it is sufficient to design a checker for unsatisfiability.

# Checking Unsatisfiability

Every formula $F$ contains only finitely many propositional variables. Obviously, $\mathcal{A}(F)$ depends only on the values of those finitely many variables in $F$ under $\mathcal{A}$.

If $F$ contains $n$ distinct propositional variables, then it is sufficient to check $2^n$ valuations to see whether $F$ is satisfiable or not.
$\Rightarrow$ truth table.

So the satisfiability problem is clearly decidable
(but, by Cook's Theorem, NP-complete).

Nevertheless, in practice, there are (much) better methods than truth tables to check the satisfiability of a formula. (later more)

# Some Important Equivalences

The following equivalences are valid for all formulas $F$, $G$, $H$:

$$(F \wedge F) \leftrightarrow F$$

$$(F \vee F) \leftrightarrow F \qquad \text{(Idempotency)}$$

$$(F \wedge G) \leftrightarrow (G \wedge F)$$

$$(F \vee G) \leftrightarrow (G \vee F) \qquad \text{(Commutativity)}$$

$$(F \wedge (G \wedge H)) \leftrightarrow ((F \wedge G) \wedge H)$$

$$(F \vee (G \vee H)) \leftrightarrow ((F \vee G) \vee H) \qquad \text{(Associativity)}$$

$$(F \wedge (G \vee H)) \leftrightarrow ((F \wedge G) \vee (F \wedge H))$$

$$(F \vee (G \wedge H)) \leftrightarrow ((F \vee G) \wedge (F \vee H)) \qquad \text{(Distributivity)}$$

# Some Important Equivalences

The following equivalences are valid for all formulas $F, G, H$:

$$(F \wedge (F \vee G)) \leftrightarrow F$$

$$(F \vee (F \wedge G)) \leftrightarrow F \qquad \text{(Absorption)}$$

$$(\neg\neg F) \leftrightarrow F \qquad \text{(Double Negation)}$$

$$\neg(F \wedge G) \leftrightarrow (\neg F \vee \neg G)$$

$$\neg(F \vee G) \leftrightarrow (\neg F \wedge \neg G) \qquad \text{(De Morgan's Laws)}$$

$(F \wedge G) \leftrightarrow F$, if $G$ is a tautology

$(F \vee G) \leftrightarrow \top$, if $G$ is a tautology (Tautology Laws)

$(F \wedge G) \leftrightarrow \bot$, if $G$ is unsatisfiable

$(F \vee G) \leftrightarrow F$, if $G$ is unsatisfiable (Tautology Laws)

# 1.4 Normal Forms

We define conjunctions of formulas as follows:

$$\bigwedge_{i=1}^{0} F_i = \top.$$

$$\bigwedge_{i=1}^{1} F_i = F_1.$$

$$\bigwedge_{i=1}^{n+1} F_i = \bigwedge_{i=1}^{n} F_i \wedge F_{n+1}.$$

and analogously disjunctions:

$$\bigvee_{i=1}^{0} F_i = \bot.$$

$$\bigvee_{i=1}^{1} F_i = F_1.$$

$$\bigvee_{i=1}^{n+1} F_i = \bigvee_{i=1}^{n} F_i \vee F_{n+1}.$$

# Literals and Clauses

A literal is either a propositional variable $P$ or a negated propositional variable $\neg P$.

A clause is a (possibly empty) disjunction of literals.

# Literals and Clauses

A literal is either a propositional variable $P$ or a negated propositional variable $\neg P$.

A clause is a (possibly empty) disjunction of literals.

**Example of clauses:**

| | |
|---|---|
| $\bot$ | the empty clause |
| $P$ | positive unit clause |
| $\neg P$ | negative unit clause |
| $P \vee Q \vee R$ | positive clause |
| $P \vee \neg Q \vee \neg R$ | clause |
| $P \vee P \vee \neg Q \vee \neg R \vee R$ | allow repetitions/complementary literals |

# CNF and DNF

A formula is in conjunctive normal form (CNF, clause normal form), if it is a conjunction of disjunctions of literals (or in other words, a conjunction of clauses).

A formula is in disjunctive normal form (DNF), if it is a disjunction of conjunctions of literals.

Warning: definitions in the literature differ:

- are complementary literals permitted?
- are duplicated literals permitted?
- are empty disjunctions/conjunctions permitted?

# Conversion to CNF/DNF

**Proposition 1.4:**

For every formula there is an equivalent formula in CNF (and also an equivalent formula in DNF).

Proof:

We consider the case of CNF.

Apply the following rules as long as possible (modulo associativity and commutativity of $\wedge$ and $\vee$):

**Step 1:** Eliminate equivalences:

$$(F \leftrightarrow G) \;\Rightarrow_K\; (F \rightarrow G) \wedge (G \rightarrow F)$$

# Conversion to CNF/DNF

**Step 2:** Eliminate implications:

$$(F \rightarrow G) \;\Rightarrow_K\; (\neg F \vee G)$$

**Step 3:** Push negations downward:

$$\neg(F \vee G) \;\Rightarrow_K\; (\neg F \wedge \neg G)$$
$$\neg(F \wedge G) \;\Rightarrow_K\; (\neg F \vee \neg G)$$

**Step 4:** Eliminate multiple negations:

$$\neg\neg F \;\Rightarrow_K\; F$$

The formula obtained from a formula $F$ after applying steps 1-4 is called the negation normal form (NNF) of $F$

# Conversion to CNF/DNF

**Step 5:** Push disjunctions downward:

$$(F \wedge G) \vee H \;\Rightarrow_K\; (F \vee H) \wedge (G \vee H)$$

**Step 6:** Eliminate $\top$ and $\bot$:

$$\begin{aligned}
(F \wedge \top) &\Rightarrow_K F \\
(F \wedge \bot) &\Rightarrow_K \bot \\
(F \vee \top) &\Rightarrow_K \top \\
(F \vee \bot) &\Rightarrow_K F \\
\neg\bot &\Rightarrow_K \top \\
\neg\top &\Rightarrow_K \bot
\end{aligned}$$

# Conversion to CNF/DNF

Proving termination is easy for most of the steps; only step 3 and step 5 are a bit more complicated.

The resulting formula is equivalent to the original one and in CNF.

The conversion of a formula to DNF works in the same way, except that disjunctions have to be pushed downward in step 5.

# Complexity

Conversion to CNF (or DNF) may produce a formula whose size is exponential in the size of the original one.

# Satisfiability-preserving Transformations

The goal

"find a formula $G$ in CNF such that $\models F \leftrightarrow G$"

is unpractical.

But if we relax the requirement to

"find a formula $G$ in CNF such that $F \models \bot$ iff $G \models \bot$"

we can get an efficient transformation.

# Satisfiability-preserving Transformations

**Idea:**

A formula $F[F']$ is satisfiable iff $F[P] \wedge (P \leftrightarrow F')$ is satisfiable (where $P$ new propositional variable that works as abbreviation for $F'$).

We can use this rule recursively for all subformulas in the original formula (this introduces a linear number of new propositional variables).

Conversion of the resulting formula to CNF increases the size only by an additional factor (each formula $P \leftrightarrow F'$ gives rise to at most one application of the distributivity law).

# Optimized Transformations

A further improvement is possible by taking the polarity of the subformula $F$ into account.

Assume that $F$ contains neither $\rightarrow$ nor $\leftrightarrow$. A subformula $F'$ of $F$ has positive polarity in $F$, if it occurs below an even number of negation signs; it has negative polarity in $F$, if it occurs below an odd number of negation signs.

# Optimized Transformations

**Proposition 1.5:**

Let $F[F']$ be a formula containing neither $\rightarrow$ nor $\leftrightarrow$; let $P$ be a propositional variable not occurring in $F[F']$.

If $F'$ has positive polarity in $F$, then $F[F']$ is satisfiable if and only if $F[P] \wedge (P \rightarrow F')$ is satisfiable.

If $F'$ has negative polarity in $F$, then $F[F']$ is satisfiable if and only if $F[P] \wedge (F' \rightarrow P)$ is satisfiable.

Proof:

Exercise.

This satisfiability-preserving transformation to clause form is also called structure-preserving transformation to clause form.

# Optimized Transformations

**Example:** Let $F = (Q_1 \wedge Q_2) \vee (R_1 \wedge R_2)$.

The following are equivalent:

- $F \models \bot$
- $P_F \wedge (P_F \leftrightarrow (P_{Q_1 \wedge Q_2} \vee P_{R_1 \wedge R_2}) \wedge (P_{Q_1 \wedge Q_2} \leftrightarrow (Q_1 \wedge Q_2)) \wedge (P_{R_1 \wedge R_2} \leftrightarrow (R_1 \wedge R_2)) \models \bot$
- $P_F \wedge (P_F \rightarrow (P_{Q_1 \wedge Q_2} \vee P_{R_1 \wedge R_2}) \wedge (P_{Q_1 \wedge Q_2} \rightarrow (Q_1 \wedge Q_2)) \wedge (P_{R_1 \wedge R_2} \rightarrow (R_1 \wedge R_2)) \models \bot$
- $P_F \wedge (\neg P_F \vee P_{Q_1 \wedge Q_2} \vee P_{R_1 \wedge R_2}) \wedge (\neg P_{Q_1 \wedge Q_2} \vee Q_1) \wedge (\neg P_{Q_1 \wedge Q_2} \vee Q_2) \wedge (\neg P_{R_1 \wedge R_2} \vee R_1) \wedge (\neg P_{R_1 \wedge R_2} \vee R_2)) \models$

# Proof methods

- Simple Decision Procedures

  truth table method

- Deductive methods
  - Forward reasoning

    Assumptions and axioms are logically combined by inference rules to reason towards the goal

  - Backward reasoning

    Inference rules are directly applied to the goal, possibly generating new subgoals.

# Proof methods

- Simple Decision Procedures

  truth table method

- Deductive methods
  - Inference Systems and Proofs (generalities)
  - Example: Hilbert Deductive System
  - The Resolution Procedure
  - Sequent calculi
  - The DPLL procedure

# 1.5 Inference Systems and Proofs

Inference systems $\Gamma$ (proof calculi) are sets of tuples

$$(F_1, \ldots, F_n, F_{n+1}), \quad n \geq 0,$$

called inferences or inference rules, and written

$$\frac{\overbrace{F_1 \quad \ldots \quad F_n}^{\text{premises}}}{\underbrace{F_{n+1}}_{\text{conclusion}}} \ .$$

Clausal inference system: premises and conclusions are clauses. One also considers inference systems over other data structures.

# Proofs

A proof in $\Gamma$ of a formula $F$ from a a set of formulas $N$ (called assumptions) is a sequence $F_1, \ldots, F_k$ of formulas where

(i) $F_k = F$,

(ii) for all $1 \leq i \leq k$: $F_i \in N$, or else there exists an inference $(F_{i_1}, \ldots, F_{i_{n_i}}, F_i)$ in $\Gamma$, such that $0 \leq i_j < i$, for $1 \leq j \leq n_i$.

# Soundness and Completeness

Provability $\vdash_\Gamma$ of $F$ from $N$ in $\Gamma$:

$N \vdash_\Gamma F \;\; :\Leftrightarrow \;\;$ there exists a proof $\Gamma$ of $F$ from $N$.

$\Gamma$ is called sound $:\Leftrightarrow$

$$\frac{F_1 \;\; \ldots \;\; F_n}{F} \in \Gamma \quad \Rightarrow \quad F_1, \ldots, F_n \models F$$

$\Gamma$ is called complete $:\Leftrightarrow$

$$N \models F \quad \Rightarrow \quad N \vdash_\Gamma F$$

$\Gamma$ is called refutationally complete $:\Leftrightarrow$

$$N \models \bot \quad \Rightarrow \quad N \vdash_\Gamma \bot$$