Formal Specification and Verification

1. Classical Logic

23.10.2018

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

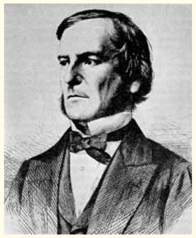
Mathematical foundations:

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 Π be a set of propositional variables.

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

Propositional Formulas

F_Π is the set of propositional formulas over Π defined as follows:

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

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

$$
\begin{aligned}
\mathcal{A}^*(\bot) &= 0 \\
\mathcal{A}^*(\top) &= 1 \\
\mathcal{A}^*(P) &= \mathcal{A}(P) \\
\mathcal{A}^*(\neg F) &= 1 - \mathcal{A}^*(F) \\
\mathcal{A}^*(F \rho G) &= \mathsf{B}_\rho(\mathcal{A}^*(F), \mathcal{A}^*(G)) \\
&\quad \text{with } \mathsf{B}_\rho \text{ the Boolean function associated with } \rho
\end{aligned}
$$

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

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

Example: Let's evaluate the formula

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

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 Π-valuations $\mathcal{A}$, whenever $\mathcal{A} \models F$ then $\mathcal{A} \models G$.

F and G are called equivalent if for all Π-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 Π-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:

$$
\begin{aligned}
F \text{ valid} \quad &\Leftrightarrow \quad \neg F \text{ unsatisfiable} \\
N \models F \quad &\Leftrightarrow \quad N \cup \{\neg F\} \text{ unsatisfiable}
\end{aligned}
$$

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:

$$
\begin{aligned}
(F \wedge F) &\leftrightarrow F \\
(F \vee F) &\leftrightarrow F && \text{(Idempotency)} \\
(F \wedge G) &\leftrightarrow (G \wedge F) \\
(F \vee G) &\leftrightarrow (G \vee F) && \text{(Commutativity)} \\
(F \wedge (G \wedge H)) &\leftrightarrow ((F \wedge G) \wedge H) \\
(F \vee (G \vee H)) &\leftrightarrow ((F \vee G) \vee H) && \text{(Associativity)} \\
(F \wedge (G \vee H)) &\leftrightarrow ((F \wedge G) \vee (F \wedge H)) \\
(F \vee (G \wedge H)) &\leftrightarrow ((F \vee G) \wedge (F \vee H)) && \text{(Distributivity)}
\end{aligned}
$$

Some Important Equivalences

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

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

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

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

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

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

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

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

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

$$(F \vee G) \leftrightarrow F, \text{ if } G \text{ is unsatisfiable} \qquad \text{(Tautology Laws)}$$

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:

$\perp$	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:

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

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:

$$(F \land G) \lor H \;\Rightarrow_K\; (F \lor H) \land (G \lor H)$$

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

$$\begin{aligned}
(F \land \top) &\;\Rightarrow_K\; F \\
(F \land \bot) &\;\Rightarrow_K\; \bot \\
(F \lor \top) &\;\Rightarrow_K\; \top \\
(F \lor \bot) &\;\Rightarrow_K\; F \\
\neg\bot &\;\Rightarrow_K\; \top \\
\neg\top &\;\Rightarrow_K\; \bot
\end{aligned}$$

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 Γ (proof calculi) are sets of tuples

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

called inferences or inference rules, and written

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

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

Proofs

A proof in Γ of a formula F from a a set of formulas N (called assumptions) is a sequence $F_1, \dots, F_k$ 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 $(F_{i_1},\ \dots,\ F_{i_{n_i}},\ F_i)$ in Γ, 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 Γ:

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

Γ is called sound $:\Leftrightarrow$

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

Γ is called complete $:\Leftrightarrow$

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

Γ is called refutationally complete $:\Leftrightarrow$

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

A deductive system for Propositional logic

Variant of the system of Hilbert-Ackermann

(Signature: $\vee, \neg$; $x \rightarrow y \equiv_{\text{Def}} \neg x \vee y$)

Axiom Schemata (to be instantiated for all possible formulae)

(1) $(p \vee p) \rightarrow p$

(2) $p \rightarrow (q \vee p)$

(3) $(p \vee q) \rightarrow (q \vee p)$

(4) $(p \rightarrow q) \rightarrow (r \vee p \rightarrow r \vee q)$

Inference rules

Modus Ponens: $\dfrac{p, \quad p \rightarrow q}{q}$

Example of proof

Prove $\phi \vee \neg\phi$

1. $((\phi \vee \phi) \rightarrow \phi) \rightarrow (\neg\phi \vee (\phi \vee \phi) \rightarrow \neg\phi \vee \phi)$ [Instance of (4)]

2. $\phi \vee \phi \rightarrow \phi$ [Instance of (1)]

3. $\neg\phi \vee (\phi \vee \phi) \rightarrow (\neg\phi \vee \phi)$ [1., 2., and MP]

3'. $= (\phi \rightarrow (\phi \vee \phi)) \rightarrow (\neg\phi \vee \phi)$ [3 and definition of $\rightarrow$]

4. $\phi \rightarrow \phi \vee \phi$ [Instance of (2)]

5. $\neg\phi \vee \phi$ [3., 4. and MP]

6. $(\neg\phi \vee \phi) \rightarrow (\phi \vee \neg\phi)$ [Instance of (3)]

7. $\phi \vee \neg\phi$ [5., 6. and MP]

Soundness

Γ is called sound $:\Leftrightarrow$

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

Γ sound iff If $N \vdash_\Gamma F$ then $N \models F$.

Theorem. The Hilbert deductive system is sound.

Proof: The proof for propositional logic is by induction on the length of the formal proof of F from N.

Proof of length 0: show that all axioms are valid

Induction step $n \mapsto n+1$: uses the definition of a proof.
It is sufficient to show that $(\phi \wedge (\phi \to \phi')) \models \phi'$.

Completeness

Γ is called complete $:\Leftrightarrow$

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

Theorem. The Hilbert deductive system is complete.

The very first proof for propositional logic was given by Bernays (a student of Hilbert).

Completeness: Sketch of Bernay's proof

Every formula is interderivable with its conjunctive normal form.

A conjuction is provable if and only if each of its conjuncts is provable.

A disjunction of propositional variables and negations of proprositional variables is provable if and only if it contains a propositional variable and its negation. Conversely, every such disjunction is provable.

So, a formula is provable if and only if every conjunct in its conjunctive normal form contains a variable and its negation.

Now suppose that ϕ is a valid but underivable formula.

Its conjunctive normal form $CNF(\phi)$ is also underivable, so it must contain a conjunct ϕ' where every propositional variable occurs only negated or unnegated but not both.

If ϕ was added as a new axiom (so that $\models \phi$ implies soundness of the new deductive system), then $CNF(\phi)$ and ϕ' would also be derivable.

By substituting X for every unnegated variable and $\neg X$ for every negated variable in ϕ', we would obtain X as a derivable formula (after some simplification), and the system would be inconsistent, which is the desired contradiction.

The Propositional Resolution Calculus

Resolution inference rule:

$$\frac{C \vee A \qquad \neg A \vee D}{C \vee D}$$

Terminology: $C \vee D$: resolvent; A: resolved atom

(Positive) factorisation inference rule:

$$\frac{C \vee A \vee A}{C \vee A}$$

The Resolution Calculus *Res*

These are schematic inference rules; for each substitution of the schematic variables C, D, and A, respectively, by propositional clauses and atoms we obtain an inference rule.

As "$\vee$" is considered associative and commutative, we assume that A and $\neg A$ can occur anywhere in their respective clauses.

Sample Refutation

1.	$\neg P \vee \neg P \vee Q$	(given)
2.	$P \vee Q$	(given)
3.	$\neg R \vee \neg Q$	(given)
4.	R	(given)
5.	$\neg P \vee Q \vee Q$	(Res. 2. into 1.)
6.	$\neg P \vee Q$	(Fact. 5.)
7.	$Q \vee Q$	(Res. 2. into 6.)
8.	Q	(Fact. 7.)
9.	$\neg R$	(Res. 8. into 3.)
10.	$\perp$	(Res. 4. into 9.)

Resolution with Implicit Factorization *RIF*

$$\frac{C \vee A \vee \ldots \vee A \qquad \neg A \vee D}{C \vee D}$$

1.	$\neg P \vee \neg P \vee Q$	(given)
2.	$P \vee Q$	(given)
3.	$\neg R \vee \neg Q$	(given)
4.	R	(given)
5.	$\neg P \vee Q \vee Q$	(Res. 2. into 1.)
6.	$Q \vee Q \vee Q$	(Res. 2. into 5.)
7.	$\neg R$	(Res. 6. into 3.)
8.	$\perp$	(Res. 4. into 7.)

Soundness and Completeness

Theorem 1.6. Propositional resolution is sound.

for both the resolution rule and the positive factorization rule

the conclusion of the inference is entailed by the premises.

Theorem 1.7. Propositional resolution is refutationally complete.

If $N \models \bot$ we can deduce $\bot$ starting from N and using

the inference rules of the propositional resolution calculus.