Propositional and Predicate Logic - I

Petr Gregor

KTIML MFF UK

WS 2022/23

What is logic good for?

For mathematicians: *“mathematics about mathematics”*.

For computer scientists:

- formal specification (case EU vs. Microsoft),
- software and hardware verification (formal verification, model checking),
- declarative programming (e.g. Prolog),
- complexity theory (Boolean functions, circuits, decision trees),
- computability (undecidability, incompleteness theorems),
- artificial intelligence (automatic reasoning, planning, ...),
- universal tools: SAT and SMT solvers (SAT modulo theory),
- database design (finite relation structures, Datalog), ...

Recommended reading

- M. Pilát, *Lecture Notes on Propositional and Predicate Logic*, 2020.
- A. Nerode, R. A. Shore, *Logic for Applications*, Springer, 2^{nd} edition, 1997.
- P. Pudlák, *Logical Foundations of Mathematics and Computational Complexity - A Gentle Introduction*, Springer, 2013.
- J. R. Shoenfield, *Mathematical Logic*, A. K. Peters, 2001.
- W. Hodges, *Shorter Model Theory*, Cambridge University Press, 1997.
- W. Rautenberg, *A concise introduction to mathematical logic*, Springer, 2009.
- lecture slides, appendix, ...

Historical overview

- Aristotle (384-322 B.C.E.) - theory of syllogistic, e.g.

  from *'no Q is R'* and *'every P is Q'* infer *'no P is R'*.

- Euclid: *Elements* (about 330 B.C.E.) - axiomatic approach to geometry

  "There is at most one line that can be drawn parallel to another given one through an external point." (5th postulate)

- Descartes: *Geometry* (1637) - algebraic approach to geometry
- Leibniz - dream of *"lingua characteristica, calculus ratiocinator"* (1679-90)
- De Morgan - introduction of propositional connectives (1847)

$$\neg(p \vee q) \leftrightarrow \neg p \wedge \neg q$$
$$\neg(p \wedge q) \leftrightarrow \neg p \vee \neg q$$

- Boole - propositional functions, algebra of logic (1847)
- Schröder - semantics of predicate logic, concept of a model (1890-1905)

Historical overview - set theory

- Cantor - *intuitive* set theory (1878), e.g. the comprehension principle

  "For every property $\varphi(x)$ there exists a set $\{x \mid \varphi(x)\}$."

- Frege - first formal system with quantifiers and relations, concept of proofs based on inference, axiomatic set theory (1879, 1884)
- Russel - Frege's set theory is contradictory (1903)

  For a set $a = \{x \mid \neg(x \in x)\}$ is $a \in a$?

- Russel, Whitehead - theory of types (1910-13)
- Zermelo (1908), Fraenkel (1922) - *standard* set theory ZFC, e.g.

  "For every property $\varphi(x)$ and a set y there is a set $\{x \in y \mid \varphi(x)\}$."

- Bernays (1937), Gödel (1940) - set theory based on classes, e.g.

  "For every property of sets $\varphi(x)$ there exists a class $\{x \mid \varphi(x)\}$."

Historical overview - algorithmization

- Hilbert - complete axiomatizaton of Euclidean geometry (1899),
  formalism - strict divorce from the intended meanings
  "It could be shown that all of mathematics follows from a correctly chosen finite system of axioms."
- Brouwer - intuitionism, emphasis on explicit constructive proofs
  "A mathematical statement corresponds to a mental construction, and its validity is verified by intuition."
- Post - completeness of propositional (and Gödel - predicate) logic
- Gödel - incompleteness theorems (1931)
- Kleene, Post, Church, Turing - formalizations of the notion of algorithm, an existence of algorithmically undecidable problems (1936)
- Robinson - resolution method (1965)
- Kowalski; Colmerauer, Roussel - Prolog (1972)

Levels of language

We distinguish different levels of logic according to the means of language, in particular to which level of quantification is admitted.

- propositional connectives — *propositional logic*

  This allows to form combined propositions from the basic ones.
- variables for objects, symbols for relations and functions, quantifiers — *first-order logic*

  This allows to form statements on objects, their properties and relations.

  The (standard) set theory is also described by a first-order language.

In higher-order languages we have, in addition,

- variables for sets of objects (also relations, functions) — *second-order logic*
- variables for sets of sets of objects, *etc.* — *third-order logic*
- ...

Examples of statements of various orders

- *"If it will not rain, we will not get wet. And if it will rain, we will get wet, but then we will get dry on the sun."* *proposition*

$$(\neg r \to \neg w) \wedge (r \to (w \wedge d))$$

- *"There exists the smallest element."* *first-order*

$$\exists x \, \forall y \, (x \leq y)$$

- The axiom of induction. *second-order*

$$\forall X \, ((X(0) \wedge \forall y(X(y) \to X(y+1))) \to \forall y \, X(y))$$

- *"Every union of open sets is an open set."* *third-order*

$$\forall \mathcal{X} \forall Y ((\forall X(\mathcal{X}(X) \to \mathcal{O}(X)) \wedge \forall z(Y(z) \leftrightarrow \exists X(\mathcal{X}(X) \wedge X(z)))) \to \mathcal{O}(Y))$$

Syntax and semantics

We will study relation between syntax and semantics:

- *syntax*: language, rules for formation of formulas, interference rules, formal proof system, proof, provability,
- *semantics*: interpreted meaning, structures, models, satisfiability, validity.

We will introduce the notion of proof as a well-defined syntactical object.

A formal proof system is

- *sound*, if every provable formula is valid,
- *complete*, if every valid formula is provable.

We will show that predicate logic (first-order logic) has formal proof systems that are both sound and complete. This does not hold for higher order logics.

Paradoxes

"Paradoxes" show us the need of precise definitions of foundational concepts.

- *Cretan paradox*

  Cretan said: "All Cretans are liars."

- *Barber paradox*

  There is a barber in a town who shaves all that do not shave themselves. Does he shave himself?

- *Liar paradox*

  This sentence is false.

- *Berry paradox*

  The expression "The smallest positive integer not definable in under eleven words" defines it in ten words.

Propositional Logic

Language

Propositional logic is a *"logic of propositional connectives"*. We start from a (nonempty) set $\mathbb{P}$ of *propositional letters* (*variables*), e.g.

$$\mathbb{P} = \{p, p_1, p_2, \ldots, q, q_1, q_2, \ldots\}$$

We usually assume that $\mathbb{P}$ is countable.

The *language* of propositional logic (over $\mathbb{P}$) consists of symbols

- propositional letters from $\mathbb{P}$
- propositional connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$
- parentheses (,)

Thus the language is given by the set $\mathbb{P}$. We say that connectives and parentheses are *symbols of logic*.

We also use symbols for constants $\top$ (true), $\bot$ (false) which are introduced as shortcuts for $p \vee \neg p$, resp. $p \wedge \neg p$ where p is any fixed variable from $\mathbb{P}$.

Formula

Propositional formulas (*propositions*) (over $\mathbb{P}$) are given inductively by

(i) every propositional letter from $\mathbb{P}$ is a proposition,

(ii) if φ, ψ are propositions, then also

$$(\neg\varphi)\ ,(\varphi \wedge \psi)\ ,(\varphi \vee \psi)\ ,(\varphi \rightarrow \psi)\ ,(\varphi \leftrightarrow \psi)$$

are propositions,

(iii) every proposition is formed by a finite number of steps (i), (ii).

- Thus propositions are (well-formed) finite sequences of symbols from the given language (strings).
- A proposition that is a part of another proposition φ as a substring is called a *subformula* (*subproposition*) of φ.
- The set of all propositions over $\mathbb{P}$ is denoted by $\mathrm{VF}_{\mathbb{P}}$.
- The set of all letters (variables) that occur in φ is denoted by $\mathrm{var}(\varphi)$.

Conventions

After introducing (standard) *priorities* for connectives we are allowed in a concise form to omit parentheses that are around a subformula formed by a connective of a higher priority.

(1) $\rightarrow, \leftrightarrow$

(2) $\wedge, \vee$

(3) $\neg$

The outer parentheses can be omitted as well, e.g.

$$(((\neg p) \wedge q) \rightarrow (\neg(p \vee (\neg q)))) \quad \text{is shortly} \quad \neg p \wedge q \rightarrow \neg(p \vee \neg q)$$

Note If we do not respect the priorities, we can obtain an ambiguous form or even a concise form of a non-equivalent proposition.

Further possibilities to omit parentheses follow from semantical properties of connectives (associativity of $\vee$, $\wedge$).

Formation trees

A *formation tree* is a finite ordered tree whose nodes are labeled with propositions according to the following rules

- leaves (and only leaves) are labeled with propositional letters,
- if a node has label $(\neg\varphi)$, then it has a single son labeled with φ,
- if a node has label $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, or $(\varphi \leftrightarrow \psi)$, then it has two sons, the left son labeled with φ, and the right son labeled with ψ.

A *formation tree of a proposition* φ is a formation tree with the root labeled with φ.

Proposition *Every proposition is associated with a unique formation tree.*

Proof By induction on the number of nested parentheses. □

Semantics

- We consider only two-valued logic.
- Propositional letters represent (atomic) statements whose ‘meaning’ is given by an assignment of *truth values* 0 *(false)* or 1 *(true)*.
- Semantics of propositional connectives is given by their *truth tables*.

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

  This determines the truth value of every proposition based on the values assigned to its propositional letters.
- Thus we may assign *“truth tables”* also to all propositions. We say that propositions represent Boolean functions (up to the order of variables).
- A *Boolean function* is an n-ary operation on $\{0,1\}$,i.e. $f\colon \{0,1\}^n \rightarrow \{0,1\}$.

Truth valuations

- A *truth assignment* is a function $v\colon \mathbb{P} \to \{0,1\}$.
- A *truth value* $\overline{v}(\varphi)$ of a proposition φ for a truth assignment v is given by

$$\overline{v}(p) = v(p) \text{ if } p \in \mathbb{P} \qquad \overline{v}(\neg\varphi) = -_1(\overline{v}(\varphi))$$
$$\overline{v}(\varphi \wedge \psi) = \wedge_1(\overline{v}(\varphi), \overline{v}(\psi)) \qquad \overline{v}(\varphi \vee \psi) = \vee_1(\overline{v}(\varphi), \overline{v}(\psi))$$
$$\overline{v}(\varphi \to \psi) = \to_1(\overline{v}(\varphi), \overline{v}(\psi)) \qquad \overline{v}(\varphi \leftrightarrow \psi) = \leftrightarrow_1(\overline{v}(\varphi), \overline{v}(\psi))$$

where $-_1, \wedge_1, \vee_1, \to_1, \leftrightarrow_1$ are the Boolean functions given by the tables.

Proposition *The truth value of a proposition φ depends only on the truth assignment of $\mathrm{var}(\varphi)$.*

Proof Easily by induction on the structure of the formula. □

Note Since the function $\overline{v}\colon \mathrm{VF}_{\mathbb{P}} \to \{0,1\}$ is a unique extension of the function v, we can (unambiguously) write v instead of $\overline{v}$.

Semantic notions

A proposition φ over $\mathbb{P}$ is

- *is true in* (*satisfied by*) *an assignment* $v\colon \mathbb{P} \to \{0,1\}$, if $\overline{v}(\varphi) = 1$. Then v is a *satisfying assignment* for φ, denoted by $v \models \varphi$.
- *valid* (*a tautology*), if $\overline{v}(\varphi) = 1$ for every $v\colon \mathbb{P} \to \{0,1\}$, i.e. φ is satisfied by every assignment, denoted by $\models \varphi$.
- *unsatisfiable* (*a contradiction*), if $\overline{v}(\varphi) = 0$ for every $v\colon \mathbb{P} \to \{0,1\}$, i.e. $\neg\varphi$ is valid.
- *independent* (*a contingency*), if $\overline{v_1}(\varphi) = 0$ and $\overline{v_2}(\varphi) = 1$ for some $v_1, v_2\colon \mathbb{P} \to \{0,1\}$, i.e. φ is neither a tautology nor a contradiction.
- *satisfiable*, if $\overline{v}(\varphi) = 1$ for some $v\colon \mathbb{P} \to \{0,1\}$, i.e. φ is not a contradiction.

Propositions φ and ψ are (logically) *equivalent*, denoted by $\varphi \sim \psi$, if $\overline{v}(\varphi) = \overline{v}(\psi)$ for every $v\colon \mathbb{P} \to \{0,1\}$, i.e. the proposition $\varphi \leftrightarrow \psi$ is valid.

Models

We reformulate these semantic notions in the terminology of models.

A *model of a language* $\mathbb{P}$ is a truth assignment of $\mathbb{P}$. The class of all models of $\mathbb{P}$ is denoted by $M(\mathbb{P})$. A proposition φ over $\mathbb{P}$ is

- *true in a model* $v \in M(\mathbb{P})$, if $\overline{v}(\varphi) = 1$. Then v is a *model of* φ, denoted by $v \models \varphi$ and $M^{\mathbb{P}}(\varphi) = \{v \in M(\mathbb{P}) \mid v \models \varphi\}$ is the *class of all models* of φ.
- *valid* (*a tautology*) if it is true in every model of the language, denoted by $\models \varphi$.
- *unsatisfiable* (*a contradiction*) if it does not have a model.
- *independent* (*a contingency*) if it is true in some model and false in other.
- *satisfiable* if it has a model.

Propositions φ and ψ are (logically) *equivalent*, denoted by $\varphi \sim \psi$, if they have same models.

Epilogue

- *Can all the mathematics be translated into logical formulas?*
  programming, AI, theorem proving, Peano: *Formulario* (1895-1908)
- *Why people (usually) do not do it?*
- *Example Is it possible to perfectly cover the chessboard without two diagonally removed corners using the domino tiles?*

  We can easily form a propositional formula that is satisfiable, if and only if the answer is yes. Then we can test its satisfiability e.g. by resolution.

  How can we solve it more *elegantly*? What is our approach based on?