Propositional and Predicate Logic - II

Petr Gregor

KTIML MFF UK

WS 2022/2023

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.

Theory

Informally, a theory is a description of "world" to which we restrict ourselves.

- A propositional *theory* over the language $\mathbb{P}$ is any set T of propositions from $\mathrm{VF}_{\mathbb{P}}$. We say that propositions of T are *axioms* of the theory T.
- A *model of theory* T over $\mathbb{P}$ is an assignment $v \in M(\mathbb{P})$ (i.e. a model of the language) in which all axioms of T are true, denoted by $v \models T$.
- A *class of models* of T is $M^{\mathbb{P}}(T) = \{v \in M(\mathbb{P}) \mid v \models \varphi \text{ for every } \varphi \in T\}$.

  For example, for $T = \{p,\ \neg p \vee \neg q,\ q \to r\}$ over $\mathbb{P} = \{p, q, r\}$ we have

$$M^{\mathbb{P}}(T) = \{(1,0,0), (1,0,1)\}$$

- If a theory is finite, it can be replaced by a *conjunction* of its axioms.
- We write $M(T, \varphi)$ as a shortcut for $M(T \cup \{\varphi\})$.

Semantics with respect to a theory

Semantic notions can be defined with respect to a theory, more precisely, with respect to its models. Let T be a theory over $\mathbb{P}$. A proposition φ over $\mathbb{P}$ is

- *valid in T* (*true in T*) if it is true in every model of T, denoted by $T \models \varphi$, We also say that φ is a (semantic) *consequence* of T.
- *unsatisfiable* (*contradictory*) *in T* (*inconsistent with T*) if it is false in every model of T,
- *independent (or contingency) in T* if it is true in some model of T and false in some other,
- *satisfiable in T* (*consistent with T*) if it is true in some model of T.

Propositions φ and ψ are *equivalent in T* (*T-equivalent*), denoted by $\varphi \sim_T \psi$, if for every model v of T, $v \models \varphi$ if and only if $v \models \psi$.

Note If all axioms of a theory T are valid (tautologies), e.g. for $T = \emptyset$, then all notions with respect to T correspond to the same notions in (pure) logic.

Adequacy

The language of propositional logic has *basic* connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$.
In general, we can introduce n-ary connective for any Boolean function, e.g.

$p \downarrow q$ *"neither p nor q"* (NOR, Peirce arrow)
$p \uparrow q$ *"not both p and q"* (NAND, Sheffer stroke)

A set of connectives is *adequate* if every Boolean function can be expressed as a proposition formed from these connectives.

Proposition $\{\neg, \wedge, \vee\}$ *is adequate.*

Proof A function $f\colon \{0,1\}^n \rightarrow \{0,1\}$ is expressed by $\bigvee_{v\in f^{-1}[1]} \bigwedge_{i=1}^{n} p_i^{v(i)}$ where $p_i^{v(i)}$ denotes the proposition p_i if $v(i) = 1$; and $\neg p_i$ if $v(i) = 0$.
For $f^{-1}[1] = \emptyset$ we take the proposition $\bot$. □

Proposition $\{\neg, \rightarrow\}$ *is adequate.*

Proof $(p \wedge q) \sim \neg(p \rightarrow \neg q)$, $(p \vee q) \sim (\neg p \rightarrow q)$. □

CNF and DNF

- A *literal* is a propositional letter or its negation. Let p^1 be the literal p and let p^0 be the literal $\neg p$. Let $\overline{l}$ denote the *complementary* literal to a literal l.
- A *clause* is a disjunction of literals, by the empty clause we mean $\bot$.
- A proposition is in *conjunctive normal form* (*CNF*) if it is a conjunction of clauses. By the empty proposition in CNF we mean $\top$.
- An *elementary conjunction* is a conjunction of literals, by the empty conjunction we mean $\top$.
- A proposition is in *disjunctive normal form* (*DNF*) if it is a disjunction of elementary conjunctions. By the empty proposition in DNF we mean $\bot$.

Note A clause or an elementary conjunction is both in CNF and DNF.

Observation *A proposition in CNF is valid if and only if each of its clauses contains a pair of complementary literals. A proposition in DNF is satisfiable if and only if at least one of its elementary conjunctions does not contain a pair of complementary literals.*

Transformations by tables

Proposition *Let $K \subseteq \{0,1\}^{\mathbb{P}}$ where $\mathbb{P}$ is finite and $\overline{K} = \{0,1\}^{\mathbb{P}} \setminus K$. Then*

$$M^{\mathbb{P}}\Big(\bigvee_{v \in K} \bigwedge_{p \in \mathbb{P}} p^{v(p)}\Big) = K = M^{\mathbb{P}}\Big(\bigwedge_{v \in \overline{K}} \bigvee_{p \in \mathbb{P}} \overline{p^{v(p)}}\Big)$$

Proof The first equality follows from $w(\bigwedge_{p \in \mathbb{P}} p^{v(p)}) = 1$ if and only if $w = v$. Similarly, the second one follows from $w(\bigvee_{p \in \mathbb{P}} \overline{p^{v(p)}}) = 1$ if and only if $w \neq v$. □

For example, $K = \{(1,0,0),(1,1,0),(0,1,0),(1,1,1)\}$ can be modeled by

$$(p \wedge \neg q \wedge \neg r) \vee (p \wedge q \wedge \neg r) \vee (\neg p \wedge q \wedge \neg r) \vee (p \wedge q \wedge r) \sim$$
$$(p \vee q \vee r) \wedge (p \vee q \vee \neg r) \wedge (p \vee \neg q \vee \neg r) \wedge (\neg p \vee q \vee \neg r)$$

Corollary *Every proposition has CNF and DNF equivalents.*

Proof The value of a proposition φ depends only on the assignment of $\mathrm{var}(\varphi)$ which is finite. Hence we can apply the above proposition for $K = M^{\mathbb{P}}(\varphi)$ and $\mathbb{P} = \mathrm{var}(\varphi)$. □

Transformations by rules

Proposition *Let φ' be the proposition obtained from φ by replacing some occurrences of a subformula ψ with ψ'. If $\psi \sim \psi'$, then $\varphi \sim \varphi'$.*

Proof By induction on the structure of the formula. □

(1) $(\varphi \to \psi) \sim (\neg\varphi \vee \psi)$, $(\varphi \leftrightarrow \psi) \sim ((\neg\varphi \vee \psi) \wedge (\neg\psi \vee \varphi))$

(2) $\neg\neg\varphi \sim \varphi$, $\neg(\varphi \wedge \psi) \sim (\neg\varphi \vee \neg\psi)$, $\neg(\varphi \vee \psi) \sim (\neg\varphi \wedge \neg\psi)$

(3) $(\varphi \vee (\psi \wedge \chi)) \sim ((\psi \wedge \chi) \vee \varphi) \sim ((\varphi \vee \psi) \wedge (\varphi \vee \chi))$

(3)' $(\varphi \wedge (\psi \vee \chi)) \sim ((\psi \vee \chi) \wedge \varphi) \sim ((\varphi \wedge \psi) \vee (\varphi \wedge \chi))$

Proposition *Every proposition can be transformed into CNF / DNF applying the transformation rules $(1), (2), (3)/(3)'$.*

Proof By induction on the structure of the formula. □

Proposition *Assume that φ contains only $\neg$, $\wedge$, $\vee$ and φ^* is obtained from φ by interchanging $\wedge$ and $\vee$, and by complementing all literals. Then* $\neg\varphi \sim \varphi^*$.

Proof By induction on the structure of the formula. □

SAT problem and solvers

- Problem SAT: Is φ in CNF satisfiable?
- *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 by a SAT solver.
- Best SAT solvers: `www.satcompetition.org`.
- SAT solver in the demo: `Glucose`, CNF format: `DIMACS`.
- *Can all the mathematics be translated into logical formulas?*
  AI, theorem proving, Peano: *Formulario* (1895-1908), Mizar system
- How can we solve it more *elegantly*? What is our approach based on?

2-SAT

- A proposition in CNF is in *k-CNF* if every its clause has at most k literals.
- *k-SAT* is the problem of satisfiability of a given proposition in k-CNF.

Although for $k = 3$ it is an NP-complete problem, we show that 2-SAT can be solved in *linear* time (with respect to the length of φ).

We neglect implementation details (computational model, representation in memory) and we use the following fact, see [ADS I].

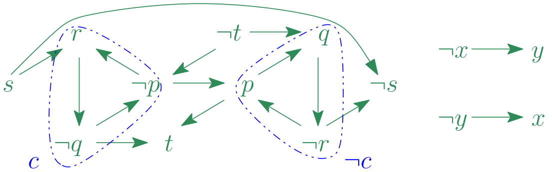
Proposition *A partition of a directed graph (V, E) to strongly connected components can be found in time $\mathcal{O}(|V| + |E|)$.*

- A directed graph G is *strongly connected* if for every two vertices u and v there are directed paths in G both from u to v and from v to u.
- A strongly connected *component* of a graph G is a maximal strongly connected subgraph of G.

Implication graphs

An *implication graph* of a proposition φ in 2-CNF is a directed graph G_φ s.t.

- vertices are all the propositional letters in φ and their negations,
- a clause $l_1 \vee l_2$ in φ is represented by a pair of edges $\overline{l_1} \to l_2$, $\overline{l_2} \to l_1$,
- a clause l_1 in φ is represented by an edge $\overline{l_1} \to l_1$.



$$p \wedge (\neg p \vee q) \wedge (\neg q \vee \neg r) \wedge (p \vee r) \wedge (r \vee \neg s) \wedge (\neg p \vee t) \wedge (q \vee t) \wedge \neg s \wedge (x \vee y)$$

Proposition *φ is satisfiable if and only if no strongly connected component of G_φ contains a pair of complementary literals.*

Proof Every satisfying assignment assigns the same value to all the literals in a same component. Thus the implication from left to right holds (necessity).

Satisfying assignment

For the implication from right to left (sufficiency), let G_φ^* be the graph obtained from G_φ by contracting strongly connected components to single vertices.

Observation *G_φ^* is acyclic, and therefore has a topological ordering $<$.*

- A directed graph is *acyclic* if it is has no directed *cycles*.
- A linear ordering $<$ of vertices of a directed graph is *topological* if $p < q$ for every edge from p to q.

Now for every unassigned component in an increasing order by $<$, assign 0 to all its literals and 1 to all literals in the complementary component.

It remains to show that such assignment v satisfies φ. If not, then G_φ^* contains edges $p \to q$ and $\overline{q} \to \overline{p}$ with $v(p) = 1$ and $v(q) = 0$. But this contradicts the order of assigning values to components since $p < q$ and $\overline{q} < \overline{p}$. □

Corollary *2-SAT can be solved in a linear time.*

Horn-SAT

- A *unit clause* is a clause containing a single literal,
- a *Horn clause* is a clause containing at most one positive literal,

$$\neg p_1 \vee \cdots \vee \neg p_n \vee q \quad \sim \quad (p_1 \wedge \cdots \wedge p_n) \to q$$

- a *Horn formula* is a conjunction of Horn clauses,
- *Horn-SAT* is the problem of satisfiability of a given Horn formula.

Algorithm

(1) *if φ contains a pair of unit clauses l and $\overline{l}$, then it is not satisfiable,*

(2) *if φ contains a unit clause l, then assign 1 to l, remove all clauses containing l, remove $\overline{l}$ from all clauses, and repeat from the start,*

(3) *if φ does not contain a unit clause, then it is satisfied by assigning 0 to all remaining propositional variables.*

Step (2) is called *unit propagation*.

Unit propagation

$$(\neg p \vee q) \wedge (\neg p \vee \neg q \vee r) \wedge (\neg r \vee \neg s) \wedge (\neg t \vee s) \wedge s \qquad v(s) = 1$$

$$(\neg p \vee q) \wedge (\neg p \vee \neg q \vee r) \wedge \neg r \qquad v(\neg r) = 1$$

$$(\neg p \vee q) \wedge (\neg p \vee \neg q) \qquad v(p) = v(q) = v(t) = 0$$

Observation *Let φ^l be the proposition obtained from φ by unit propagation. Then φ^l is satisfiable if and only if φ is satisfiable.*

Corollary *The algorithm is correct (it solves Horn-SAT).*

Proof The correctness in Step (1) is obvious, in Step (2) it follows from the observation, in Step (3) it follows from the *Horn form* since every remaining clause contains at least one negative literal.

Note A direct implementation requires quadratic time, but with an appropriate representation in memory, one can achieve linear time (w.r.t. the length of φ).

DPLL algorithm

A literal l is *pure* in a CNF formula φ if l occurs in φ and $\overline{l}$ does not occur in φ.

Algorithm DPLL(φ)

(1) *while φ contains a unit clause l, assign 1 to l, remove all clauses containing l, remove $\overline{l}$ from all clauses, and repeat, (unit propagation)*

(2) *while φ contains a pure literal l, assign 1 to l, remove all clauses containing l and repeat, (pure literal elimination)*

(3) *if φ contains an empty clause, then it is not satisfiable,*

(4) *if φ does not contain any clause, then it is satisfiable,*

(5) *choose an unassigned propositional letter p and run DPLL($\varphi \wedge p$) and DPLL($\varphi \wedge \neg p$). (branching)*

Note The algoritm runs in exponentional time in the worst case. Its correctness is easy to verify.