Propositional and Predicate Logic - V

Petr Gregor

KTIML MFF UK

ZS 2014/2015

Properties of theories

We introduce syntactic variants of previous semantically defined notions.

Let T be a theory over $\mathbb{P}$. If φ is provable from T, we say that φ is a *theorem* of T. The set of theorems of T is denoted by

$$\mathrm{Thm}^{\mathbb{P}}(T) = \{\varphi \in \mathrm{VF}_{\mathbb{P}} \mid T \vdash \varphi\}.$$

We say that a theory T is

- *inconsistent* if $T \vdash \bot$, otherwise T is *consistent*,
- *complete* if it is consistent and every proposition is provable or refutable from T, i.e. $T \vdash \varphi$ or $T \vdash \neg\varphi$ for every $\varphi \in \mathrm{VF}_{\mathbb{P}}$,
- *extension* of a theory T' over $\mathbb{P}'$ if $\mathbb{P}' \subseteq \mathbb{P}$ and $\mathrm{Thm}^{\mathbb{P}'}(T') \subseteq \mathrm{Thm}^{\mathbb{P}}(T)$; we say that an extension T of a theory T' is *simple* if $\mathbb{P} = \mathbb{P}'$; and *conservative* if $\mathrm{Thm}^{\mathbb{P}'}(T') = \mathrm{Thm}^{\mathbb{P}}(T) \cap \mathrm{VF}_{\mathbb{P}'}$,
- *equivalent* with a theory T' if T is an extension of T' and vice-versa.

Corollaries

From the soundness and completeness of the tableau method it follows that these syntactic definitions agree with their semantic variants.

Corollary *For every theory T and propositions φ, ψ over $\mathbb{P}$,*

- $T \vdash \varphi$ *if and only if* $T \models \varphi$,
- $\mathrm{Thm}^{\mathbb{P}}(T) = \theta^{\mathbb{P}}(T)$,
- T *is inconsistent if and only if* T *is unsatisfiable, i.e. it has no model,*
- T *is complete if and only if* T *is semantically complete, i.e. it has a single model,*
- $T, \varphi \vdash \psi$ *if and only if* $T \vdash \varphi \to \psi$ *(Deduction theorem).*

Remark Deduction theorem can be proved directly by transformations of tableaux.

Theorem on compactness

Theorem *A theory T has a model iff every finite subset of T has a model.*

Proof 1 The implication from left to right is obvious. If T has no model, then it is inconsistent, i.e. $\perp$ is provable by a systematic tableau τ from T. Since τ is finite, $\perp$ is provable from some finite $T' \subseteq T$, i.e. T' has no model. □

Remark *This proof is based on finiteness of proofs, soundness and completeness. We present an alternative proof (applying König's lemma).*

Proof 2 Let $T = \{\varphi_i \mid i \in \mathbb{N}\}$. Consider a tree S on (certain) finite binary strings σ ordered by being a prefix. We put $\sigma \in S$ if and only if there exists an assignment v with prefix σ such that $v \models \varphi_i$ for every $i \leq \mathrm{lth}(\sigma)$.

Observation *S has an infinite branch if and only if T has a model.*

Since $\{\varphi_i \mid i \in n\} \subseteq T$ has a model for every $n \in \mathbb{N}$, every level in S is nonempty. Thus S is infinite and moreover binary, hence by König's lemma, S contains an infinite branch. □

Application of compactness

A graf (V,E) is *k-colorable* if there exists $c\colon V \to k$ such that $c(u) \neq c(v)$ for every edge $\{u,v\} \in E$.

Theorem *A countable graph $G = (V,E)$ is k-colorable if and only if every finite subgraph of G is k-colorable.*

Proof The implication $\Rightarrow$ is obvious. Assume that every finite subgraph of G is k-colorable. Consider $\mathbb{P} = \{p_{u,i} \mid u \in V, i \in k\}$ and a theory T with axioms

$p_{u,0} \vee \cdots \vee p_{u,k-1}$	for every $u \in V$,
$\neg(p_{u,i} \wedge p_{u,j})$	for every $u \in V, i < j < k$,
$\neg(p_{u,i} \wedge p_{v,i})$	for every $\{u,v\} \in E, i < k$.

Then G is k-colorable if and only if T has a model. By compactness, it suffices to show that every finite $T' \subseteq T$ has a model. Let G' be the subgraph of G induced by vertices u such that $p_{u,i}$ appears in T' for some i. Since G' is k-colorable by the assumption, the theory T' has a model. $\square$

Resolution method - introduction

Main features of the resolution method (*informally*)

- is the underlying method of many systems, e.g. Prolog interpreters, SAT solvers, automated deduction / verification systems, . . .
- assumes input formulas in CNF (in general, *"expensive"* transformation),
- works under set representation (clausal form) of formulas,
- has a single rule, so called a resolution rule,
- has no explicit axioms (or atomic tableaux), but certain axioms are incorporated *"inside"* via various formatting rules,
- is a *refutation* procedure, similarly as the tableau method; that is, it tries to show that a given formula (or theory) is unsatisfiable,
- has several refinements e.g. with specific conditions on when the resolution rule may be applied.

Set representation (clausal from) of CNF formulas

- A *literal* l is a prop. letter or its negation. $\overline{l}$ is its *complementary* literal.
- A *clause* C is a finite set of literals (*"forming disjunction"*). The empty clause, denoted by $\square$, is never satisfied (has no satisfied literal).
- A *formula* S is a (possibly infinite) set of clauses (*"forming conjunction"*). An empty formula $\emptyset$ is always satisfied (is has no unsatisfied clause). Infinite formulas represent infinite theories (as conjunction of axioms).
- A (*partial*) *assignment* $\mathcal{V}$ is a consistent set of literals, i.e. not containing any pair of complementary literals. An assignment $\mathcal{V}$ is *total* if it contains a positive or negative literal for each propositional letter.
- $\mathcal{V}$ *satisfies* S, denoted by $\mathcal{V} \models S$, if $C \cap \mathcal{V} \neq \emptyset$ for every $C \in S$.

$((\neg p \vee q) \wedge (\neg p \vee \neg q \vee r) \wedge (\neg r \vee \neg s) \wedge (\neg t \vee s) \wedge s)$ is represented by

$$S = \{\{\neg p, q\}, \{\neg p, \neg q, r\}, \{\neg r, \neg s\}, \{\neg t, s\}, \{s\}\} \quad \text{and}$$

$$\mathcal{V} \models S \quad \text{for} \quad \mathcal{V} = \{s, \neg r, \neg p\}$$

Resolution rule

Let C_1, C_2 be clauses with $l \in C_1$, $\bar{l} \in C_2$ for some literal l. Then from C_1 and C_2 infer through the literal l the clause C, called a *resolvent*, where

$$C = (C_1 \setminus \{l\}) \cup (C_2 \setminus \{\bar{l}\}).$$

Equivalently, if $\sqcup$ means union of disjoint sets,

$$\frac{C_1' \sqcup \{l\}, C_2' \sqcup \{\bar{l}\}}{C_1' \cup C_2'}$$

For example, from $\{p, q, r\}$ and $\{\neg p, \neg q\}$ we can infer $\{q, \neg q, r\}$ or $\{p, \neg p, r\}$.

Observation *The resolution rule is sound; that is, for every assignment $\mathcal{V}$*

$$\mathcal{V} \models C_1 \text{ and } \mathcal{V} \models C_2 \quad \Rightarrow \quad \mathcal{V} \models C.$$

Remark *The resolution rule is a special case of the (so called) cut rule*

$$\frac{\varphi \vee \psi, \ \neg\varphi \vee \chi}{\psi \vee \chi}$$

where φ, ψ, χ are arbitrary formulas.

Resolution proof

- A *resolution proof* (*deduction*) of a clause C from a formula S is a finite sequence $C_0, \ldots, C_n = C$ such that for every $i \leq n$, we have $C_i \in S$ or C_i is a resolvent of some previous clauses,
- a clause C is (resolution) *provable* from S, denoted by $S \vdash_R C$, if it has a resolution proof from S,
- a (resolution) *refutation* of formula S is a resolution proof of $\square$ from S,
- S is (resolution) *refutable* if $S \vdash_R \square$.

Theorem (soundness) *If S is resolution refutable, then S is unsatisfiable.*

Proof Let $S \vdash_R \square$. If it was $\mathcal{V} \models S$ for some assignment $\mathcal{V}$, from the soundness of the resolution proof we would have $\mathcal{V} \models \square$, which is impossible. ∎

Resolution trees and closures

A *resolution tree* of a clause C from formula S is finite binary tree with nodes labeled by clauses so that

(i) the root is labeled C,

(ii) the leaves are labeled with clauses from S,

(iii) every inner node is labeled with a resolvent of the clauses in his sons.

Observation *C has a resolution tree from S if and only if $S \vdash_R C$.*

A *resolution closure* $\mathcal{R}(S)$ of a formula S is the smallest set satisfying

(i) $C \in \mathcal{R}(S)$ for every $C \in S$,

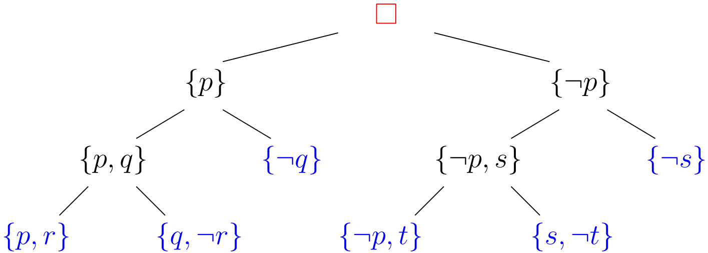
(ii) if $C_1, C_2 \in \mathcal{R}(S)$ and C is a resolvent of C_1, C_2, then $C \in \mathcal{R}(S)$.

Observation *$C \in \mathcal{R}(S)$ if and only if $S \vdash_R C$.*

Remark *All notions on resolution proofs can therefore be equivalently introduced in terms of resolution trees or resolution closures.*

Example

Formula $((p \vee r) \wedge (q \vee \neg r) \wedge (\neg q) \wedge (\neg p \vee t) \wedge (\neg s) \wedge (s \vee \neg t))$ is unsatisfiable since for $S = \{\{p,r\},\{q,\neg r\},\{\neg q\},\{\neg p,t\},\{\neg s\},\{s,\neg t\}\}$ we have $S \vdash_R \square$.



The resolution closure of S (*the closure of* S *under resolution*) is

$$\mathcal{R}(S) = \{\{p,r\},\{q,\neg r\},\{\neg q\},\{\neg p,t\},\{\neg s\},\{s,\neg t\},\{p,q\},\{\neg r\},\{r,t\},\\ \{q,t\},\{\neg t\},\{\neg p,s\},\{r,s\},\{t\},\{q\},\{q,s\},\square,\{\neg p\},\{p\},\{r\},\{s\}\}.$$

Reduction by substitution

Let S be a formula and l be a literal. Let us define

$$S^l = \{C \setminus \{\bar{l}\} \mid l \notin C \in S\}.$$

Observation

- S^l is equivalent to a formula obtained from S by substituting the constant $\top$ (true, 1) for all literals l and the constant $\bot$ (false, 0) for all literals $\bar{l}$ in S,
- Neither l nor $\bar{l}$ occurs in (the clauses of) S^l.
- if $\{\bar{l}\} \in S$, then $\square \in S^l$.

Lemma *S is satisfiable if and only if S^l or $S^{\bar{l}}$ is satisfiable.*

Proof ($\Rightarrow$) Let $\mathcal{V} \models S$ for some $\mathcal{V}$ and assume (w.l.o.g.) that $\bar{l} \notin \mathcal{V}$.

- Then $\mathcal{V} \models S^l$ as for $l \notin C \in S$ we have $\mathcal{V} \setminus \{l, \bar{l}\} \models C$ and thus $\mathcal{V} \models C \setminus \{\bar{l}\}$.
- On the other hand ($\Leftarrow$), assume (w.l.o.g.) that $\mathcal{V} \models S^l$ for some $\mathcal{V}$.
- Since neither l nor $\bar{l}$ occurs in S^l, we have $\mathcal{V}' \models S^l$ for $\mathcal{V}' = (\mathcal{V} \setminus \{\bar{l}\}) \cup \{l\}$.
- Then $\mathcal{V}' \models S$ since for $C \in S$ containing l we have $l \in \mathcal{V}'$ and for $C \in S$ not containing l we have $\mathcal{V}' \models (C \setminus \{\bar{l}\}) \in S^l$. ■

Tree of reductions

Step by step reductions of literals can be represented in a binary tree.

$$S = \{\{p\}, \{\neg q\}, \{\neg p, \neg q\}\}$$

$$S^p = \{\{\neg q\}\} \qquad S^{\overline{p}} = \{\square, \{\neg q\}\}$$

$$S^{pq} = \{\square\} \qquad S^{p\overline{q}} = \emptyset$$

Corollary *S is unsatisfiable if and only if every branch contains $\square$.*

Remarks *Since S can be infinite over a countable language, this tree can be infinite. However, if S is unsatisfiable, by the compactness theorem there is a finite $S' \subseteq S$ that is unsatisfiable. Thus after reduction of all literals occurring in S', there will be $\square$ in every branch after finitely many steps.*

Completeness of resolution

Theorem *If a finite S is unsatisfiable, it is resolution refutable, i.e. $S \vdash_R \square$.*

Proof By induction on the number of variables in S we show that $S \vdash_R \square$.

- If unsatisfiable S has no variable, it is $S = \{\square\}$ and thus $S \vdash_R \square$,
- Let l be a literal occurring in S. By Lemma, S^l and $S^{\bar{l}}$ are unsatisfiable.
- Since S^l and $S^{\bar{l}}$ have less variables than S, by induction there exist resolution trees T^l and $T^{\bar{l}}$ for derivation of $\square$ from S^l resp. $S^{\bar{l}}$.
- If every leaf of T^l is in S, then T^l is a resolution tree of $\square$ from S, $S \vdash_R \square$.
- Otherwise, by appending the literal $\bar{l}$ to every leaf of T^l that is not in S, (and to all predecessors) we obtain a resolution tree of $\{\bar{l}\}$ from S.
- Similarly, we get a resolution tree $\{l\}$ from S by appending l in the tree $T^{\bar{l}}$.
- By resolution of roots $\{\bar{l}\}$ and $\{l\}$ we get a resolution tree of $\square$ from S. ■

Corollary *If S is unsatisfiable, it is resolution refutable, i.e. $S \vdash_R \square$.*

Proof Follows from the previous theorem by applying compactness.

Linear resolution - introduction

The resolution method can be significantly refined.

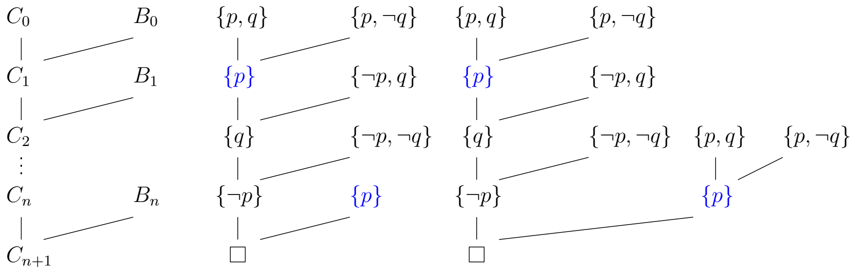
- A *linear proof* of a clause C from a formula S is a finite sequence of pairs $(C_0, B_0), \ldots, (C_n, B_n)$ such that $C_0 \in S$ and for every $i \leq n$
  - $i)$ $B_i \in S$ or $B_i = C_j$ for some $j < i$, and
  - $ii)$ C_{i+1} is a resolvent of C_i and B_i where $C_{n+1} = C$.
- C_0 is called a *starting* clause, C_i a *central* clause, B_i a *side* clause.
- C is *linearly provable* from S, $S \vdash_L C$, if it has a linear proof from S.
- A *linear refutation* of S is a linear proof of $\square$ from S.
- S is *linearly refutable* if $S \vdash_L \square$.

Observation (soundness) *If S is linearly refutable, it is unsatisfiable.*

Proof Every linear proof can be transformed to a (general) resolution proof.

Remark *The completeness is preserved as well (proof omitted here).*

Example of linear resolution



$a)$ $b)$ $c)$

- $a)$ a general form of linear resolution,
- $b)$ for $S = \{\{p,q\},\{p,\neg q\},\{\neg p,q\},\{\neg p,\neg q\}\}$ we have $S \vdash_L \square$,
- $c)$ a transformation of a linear proof to a (general) resolution proof.

LI-resolution

Linear resolution can be further refined for Horn formulas as follows.

- a *Horn clause* is a clause containing at most one positive literal,
- a *Horn formula* is a (possibly infinite) set of Horn clauses,
- a *fact* is a (Horn) clause $\{p\}$ where p is a positive literal,
- a *rule* is a (Horn) clause with exactly one positive literal and at least one negative literal. Rules and facts are *program clauses*,
- a *goal* is a nonempty (Horn) clause with only negative literals.

Observation *If a Horn formula S is unsatisfiable and $\square \notin S$, it contains some fact and some goal.*

Proof If S does not contain any fact (goal), it is satisfied by the assignment of all propositional variables to 0 (resp. to 1). ■

A *linear input resolution* (*LI-resolution*) from a formula S is a linear resolution from S in which every side clause B_i is from the (input) formula S. We write $S \vdash_{LI} C$ to denote that C is provable by LI-resolution from S.