Propositional and Predicate Logic - VI

Petr Gregor

KTIML MFF UK

WS 2022/2023

Linear resolution - introduction

The resolution method can be significantly refined.

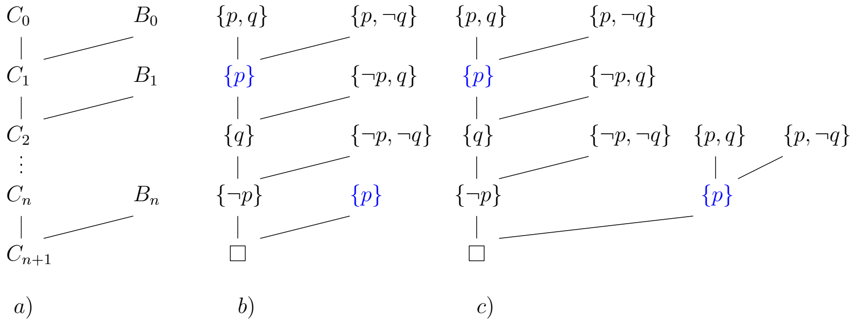
- A *linear proof* of a clause C from a formula S is a finite sequence of pairs $(C_0, B_0), \dots, (C_n, B_n)$ such that $C_0 \in S$ and for every $i \leq n$
  - *i)* $B_i \in S$ or $B_i = C_j$ for some $j < i$, and
  - *ii)* C_{i+1} is a resolvent of C_i and B_i where $C_{n+1} = C$.
- C_0 is called 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



- $a)$ 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.

Completeness of LI-resolution for Horn formulas

Theorem *If T is satisfiable Horn formula but $T \cup \{G\}$ is unsatisfiable for some goal G, then $\square$ has a LI-resolution from $T \cup \{G\}$ with starting clause G.*

Proof By the compactness theorem we may assume that T is finite.

- We proceed by induction on the number of variables in T.
- By Observation, T contains a fact $\{p\}$ for some variable p.
- By Lemma, $T' = (T \cup \{G\})^p = T^p \cup \{G^p\}$ is unsatisfiable where $G^p = G \setminus \{\overline{p}\}$.
- If $G^p = \square$, we have $G = \{\overline{p}\}$ and thus $\square$ is a resolvent of G and $\{p\} \in T$.
- Otherwise, since T^p is satisfiable (by the assignment satisfying T) and has less variables than T, by induction assumption, there is an LI-resolution of $\square$ from T' starting with G^p.
- By appending the literal $\overline{p}$ to all leaves that are not in $T \cup \{G\}$ (and nodes below) we obtain an LI-resolution of $\{\overline{p}\}$ from $T \cup \{G\}$ that starts with G.
- By an additional resolution step with the fact $\{p\} \in T$ we resolve $\square$. ■

Example of LI-resolution

$T = \{\{p, \neg r, \neg s\}, \{r, \neg q\}, \{q, \neg s\}, \{s\}\}, \qquad G = \{\neg p, \neg q\}$

$T^s = \{\{p, \neg r\}, \{r, \neg q\}, \{q\}\}$

$T^{sq} = \{\{p, \neg r\}, \{r\}\}$

$T^{sqr} = \{\{p\}\}$

$G^{sqr} = \{\neg p\} \quad \{p\}$

$G^{sqrp} = \square$

$G^{sq} = \{\neg p\} \quad \{p, \neg r\}$; $\{\neg r\} \quad \{r\}$; $\square$

$G^s = \{\neg p, \neg q\} \quad \{p, \neg r\}$; $\{\neg q, \neg r\} \quad \{r, \neg q\}$; $\{\neg q\} \quad \{q\}$; $\square$

$G = \{\neg p, \neg q\} \quad \{p, \neg r, \neg s\}$; $\{\neg q, \neg r, \neg s\} \quad \{r, \neg q\}$; $\{\neg q, \neg s\} \quad \{q, \neg s\}$; $\{\neg s\} \quad \{s\}$; $\square$

$T^{sqr}, G^{sqr} \vdash_{LI} \square \qquad T^{sq}, G^{sq} \vdash_{LI} \square \qquad T^s, G^s \vdash_{LI} \square \qquad T, G \vdash_{LI} \square$

Program in Prolog

A (propositional) *program* (in Prolog) is a Horn formula containing only program clauses, i.e. facts or rules.

a rule	$p :\!-\ q, r.$	$q \wedge r \to p$	$\{p, \neg q, \neg r\}$	
	$p :\!-\ s.$	$s \to p$	$\{p, \neg s\}$	
	$q :\!-\ s.$	$s \to q$	$\{q, \neg s\}$	
a fact	$r.$	r	$\{r\}$	
	$s.$	s	$\{s\}$	*a program*
a query	$?\!-\ p, q.$		$\{\neg p, \neg q\}$	*a goal*

We would like to know whether a given query follows from a given program.

Corollary *For every program* P *and query* $(p_1 \wedge \ldots \wedge p_n)$ *it is equivalent that*

(1) $P \models p_1 \wedge \ldots \wedge p_n$,

(2) $P \cup \{\neg p_1, \ldots, \neg p_n\}$ is unsatisfiable,

(3) $\square$ has LI-resolution from $P \cup \{G\}$ starting by goal $G = \{\neg p_1, \ldots, \neg p_n\}$.

Hilbert's calculus

- basic connectives: $\neg$, $\rightarrow$ (others can be defined from them)
- *logical axioms* (schemes of axioms):

$$
\begin{array}{rc}
(i) & \varphi \rightarrow (\psi \rightarrow \varphi) \\
(ii) & (\varphi \rightarrow (\psi \rightarrow \chi)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \chi)) \\
(iii) & (\neg\varphi \rightarrow \neg\psi) \rightarrow (\psi \rightarrow \varphi)
\end{array}
$$

  where φ, ψ, χ are any propositions (of a given language).
- *a rule of inference*:

$$\frac{\varphi,\ \varphi \rightarrow \psi}{\psi} \qquad \text{(modus ponens)}$$

A *proof* (in *Hilbert-style*) of a formula φ from a theory T is a finite sequence $\varphi_0, \ldots, \varphi_n = \varphi$ of formulas such that for every $i \leq n$

- φ_i is a logical axiom or $\varphi_i \in T$ (an axiom of the theory), or
- φ_i can be inferred from the previous formulas applying a rule of inference.

Remark *Choice of axioms and inference rules differs in various Hilbert-style proof systems.*

Example and soundness

A formula φ is *provable* from T if it has a proof from T, denoted by $T \vdash_H \varphi$.
If $T = \emptyset$, we write $\vdash_H \varphi$. E.g. for $T = \{\neg\varphi\}$ we have $T \vdash_H \varphi \to \psi$ for every ψ.

$1)$	$\neg\varphi$	an axiom of T
$2)$	$\neg\varphi \to (\neg\psi \to \neg\varphi)$	a logical axiom (i)
$3)$	$\neg\psi \to \neg\varphi$	by modus ponens from 1), 2)
$4)$	$(\neg\psi \to \neg\varphi) \to (\varphi \to \psi)$	a logical axiom (iii)
$5)$	$\varphi \to \psi$	by modus ponens from 3), 4)

Theorem *For every theory T and formula φ, $T \vdash_H \varphi \ \Rightarrow \ T \models \varphi$.*

Proof

- If φ is an axiom (logical or from T), then $T \models \varphi$ (l. axioms are tautologies),
- if $T \models \varphi$ and $T \models \varphi \to \psi$, then $T \models \psi$, i.e. modus ponens is sound,
- thus every formula in a proof from T is valid in T. □

Remark *The completeness holds as well, i.e. $T \models \varphi \Rightarrow T \vdash_H \varphi$.*

Predicate logic

Deals with statements about objects, their properties and relations.

"She is intelligent and her father knows the rector." $I(x) \land K(f(x), r)$

- x is a variable, representing an object,
- r is a constant symbol, representing a particular object,
- f is a function symbol, representing a function,
- I, K are relation (predicate) symbols, representing relations (the property of *"being intelligent"* and the relation *"to know"*).

"Everybody has a father." $(\forall x)(\exists y)(y = f(x))$

- $(\forall x)$ is the universal quantifier (*for every* x),
- $(\exists y)$ is the existential quantifier (*there exists* y),
- $=$ is a (binary) relation symbol, representing the identity relation.

Language

A first-order language consists of

- variables $x, y, z, \ldots, x_0, x_1, \ldots$ (countable many), the set of all variables is denoted by Var,
- function symbols $f, g, h, \ldots$, including constant symbols $c, d, \ldots$, which are nullary function symbols,
- relation (predicate) symbols $P, Q, R, \ldots$, eventually the symbol $=$ (equality) as a special relation symbol,
- quantifiers $(\forall x)$, $(\exists x)$ for every variable $x \in \mathrm{Var}$,
- logical connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$
- parentheses (,)

Every function and relation symbol S has an associated *arity* $\mathrm{ar}(S) \in \mathbb{N}$.

Remark *Compared to propositional logic we have no (explicit) propositional variables, but they can be introduced as nullary relation symbols.*

Signatures

- *Symbols of logic* are variables, quantifiers, connectives and parentheses.
- *Non-logical symbols* are function and relation symbols except the equality symbol. The equality is (usually) considered separately.
- A *signature* is a pair $\langle \mathcal{R}, \mathcal{F} \rangle$ of disjoint sets of relation and function symbols with associated arities, whereas none of them is the equality symbol. A signature lists all non-logical symbols.
- A *language* is determined by a signature $L = \langle \mathcal{R}, \mathcal{F} \rangle$ and by specifying whether it is a language with equality or not. A language must contain at least one relation symbol (non-logical or the equality).

Remark *The meaning of symbols in a language is not assigned, e.g. the symbol $+$ does not have to represent the standard addition.*

Examples of languages

We describe a language by a list of all non-logical symbols with eventual clarification of arity and whether they are relation or function symbols.

The following examples of languages are all with equality.

- $L = \langle\ \rangle$ is the language of pure equality,
- $L = \langle c_i \rangle_{i \in \mathbb{N}}$ is the language of countable many constants,
- $L = \langle \leq \rangle$ is the language of orderings,
- $L = \langle E \rangle$ is the language of the graph theory,
- $L = \langle +, -, 0 \rangle$ is the language of the group theory,
- $L = \langle +, -, \cdot, 0, 1 \rangle$ is the language of the field theory,
- $L = \langle -, \wedge, \vee, 0, 1 \rangle$ is the language of Boolean algebras,
- $L = \langle S, +, \cdot, 0, \leq \rangle$ is the language of arithmetic,

where c_i, 0, 1 are constant symbols, S, $-$ are unary function symbols, $+$, $\cdot$, $\wedge$, $\vee$ are binary function symbols, E, $\leq$ are binary relation symbols.

Terms

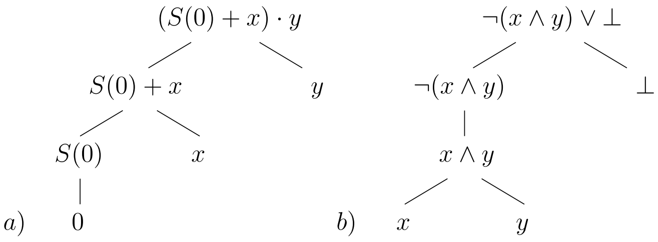
Are expressions representing values of (composed) functions.

Terms of a language L are defined inductively by

- (i) every variable or constant symbol in L is a term,
- (ii) if f is a function symbol in L of arity $n > 0$ and $t_1, \ldots, t_n$ are terms, then also the expression $f(t_1, \ldots, t_n)$ is a term,
- (iii) every term is formed by a finite number of steps (i), (ii).

- A *ground term* is a term with no variables.
- The set of all terms of a language L is denoted by Term_L.
- A term that is a part of another term t is called a *subterm* of t.
- The structure of terms can be represented by their formation trees.
- For binary function symbols we often use infix notation, e.g. we write $(x + y)$ instead of $+(x, y)$.

Examples of terms



- $a)$ The formation tree of the term $(S(0)+x)\cdot y$ of the language of arithmetic.
- $b)$ Propositional formulas only with connectives $\neg$, $\wedge$, $\vee$, eventually with constants $\top$, $\bot$ can be viewed as terms of the language of Boolean algebras.

Atomic formulas

Are the simplest formulas.

- An *atomic formula* of a language L is an expression $R(t_1, \ldots, t_n)$ where R is an n-ary relation symbol in L and $t_1, \ldots, t_n$ are terms of L.
- The set of all atomic formulas of a language L is denoted by AFm_L.
- The structure of an atomic formula can be represented by a formation tree from the formation subtrees of its terms.
- For binary relation symbols we often use infix notation, e.g. $t_1 = t_2$ instead of $=(t_1, t_2)$ or $t_1 \leq t_2$ instead of $\leq(t_1, t_2)$.
- *Examples of atomic formulas*

$$K(f(x), r), \quad x \cdot y \leq (S(0) + x) \cdot y, \quad \neg(x \wedge y) \vee \bot = \bot.$$

Formula

Formulas of a language L are defined inductively by

(i) every atomic formula is a formula,

(ii) if φ, ψ are formulas, then also the following expressions are formulas

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

(iii) if φ is a formula and x is a variable, then also the expressions $((\forall x)\varphi)$ and $((\exists x)\varphi)$ are formulas.

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

- The set of all formulas of a language L is denoted by Fm_L.
- A formula that is a part of another formula φ is called a *subformula* of φ.
- The structure of formulas can be represented by their formation trees.

Conventions

- After introducing *priorities* for binary function symbols e.g. $+$, $\cdot$ we are in infix notation allowed to omit parentheses that are around a subterm formed by a symbol of higher priority, e.g. $x \cdot y + z$ instead of $(x \cdot y) + z$.
- After introducing *priorities* for connectives and quantifiers we are allowed to omit parentheses that are around subformulas formed by connectives of higher priority.

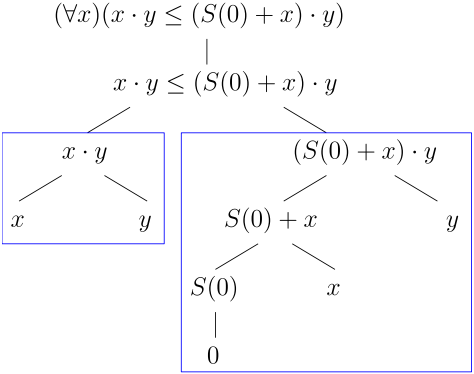
$$(1)\quad \rightarrow,\ \leftrightarrow \qquad (2)\quad \wedge,\ \vee \qquad (3)\quad \neg,\ (\forall x),\ (\exists x)$$

- They can be always omitted around subformulas formed by $\neg$, $(\forall x)$, $(\exists x)$.
- We may also omit parentheses in $(\forall x)$ and $(\exists x)$ for every $x \in \mathrm{Var}$.
- The outer parentheses may be omitted as well.

$$(((\neg((\forall x)R(x))) \wedge ((\exists y)P(y))) \rightarrow (\neg(((\forall x)R(x)) \vee (\neg((\exists y)P(y))))))$$

$$\neg(\forall x)R(x) \wedge (\exists y)P(y) \rightarrow \neg((\forall x)R(x) \vee \neg(\exists y)P(y))$$

An example of a formula



The formation tree of the formula $(\forall x)(x \cdot y \leq (S(0) + x) \cdot y)$.

Occurrences of variables

Let φ be a formula and x be a variable.

- An *occurrence* of x in φ is a leaf labeled by x in the formation tree of φ.
- An occurrence of x in φ is *bound* if it is in some subformula ψ that starts with $(\forall x)$ or $(\exists x)$. An occurrence of x in φ is *free* if it is not bound.
- A variable x is *free* in φ if it has at least one free occurrence in φ. It is *bound* in φ if it has at least one bound occurrence in φ.
- A variable x can be both free and bound in φ. For example in

$$(\forall x)(\exists y)(x \leq y) \vee x \leq z.$$

- We write $\varphi(x_1, \ldots, x_n)$ to denote that $x_1, \ldots, x_n$ are all free variables in the formula φ. *(φ states something about these variables.)*

Remark *We will see that the truth value of a formula (in a given interpretation of symbols) depends only on the assignment of free variables.*

Open and closed formulas

- A formula is *open* if it is without quantifiers. For the set OFm_L of all open formulas in a language L it holds that $\mathrm{AFm}_L \subsetneq \mathrm{OFm}_L \subsetneq \mathrm{Fm}_L$.
- A formula is *closed* (a *sentence*) if it has no free variable; that is, all occurrences of variables are bound.
- A formula can be both open and closed. In this case, all its terms are ground terms.

$x + y \leq 0$	*open*, $\varphi(x, y)$
$(\forall x)(\forall y)(x + y \leq 0)$	*a sentence*,
$(\forall x)(x + y \leq 0)$	*neither open nor a sentence*, $\varphi(y)$
$1 + 0 \leq 0$	*open sentence*

Remark *We will see that in a fixed interpretation of symbols a sentence has a fixed truth value; that is, it does not depend on the assignment of variables.*

Instances

After substituting a term t for a free variable x in a formula φ, we would expect that the new formula (newly) says about t "the same" as φ did about x.

$\varphi(x)$	$(\exists y)(x + y = 1)$	*"there is an element $1 - x$"*
for $t = 1$ we can $\varphi(x/t)$	$(\exists y)(1 + y = 1)$	*"there is an element $1 - 1$"*
for $t = y$ we cannot	$(\exists y)(y + y = 1)$	*"1 is divisible by 2"*

- A term t is *substitutable* for a variable x in a formula φ if substituting t for all free occurrences of x in φ does not introduce a new bound occurrence of a variable from t.
- Then we denote the obtained formula $\varphi(x/t)$ and we call it an *instance* of the formula φ after a *substitution* of a term t for a variable x.
- t is not substitutable for x in φ if and only if x has a free occurrence in some subformula that starts with $(\forall y)$ or $(\exists y)$ for some variable y in t.
- Ground terms are always substitutable.

Variants

Quantified variables can be (under certain conditions) renamed so that we obtain an equivalent formula.

Let $(Qx)\psi$ be a subformula of φ where Q means $\forall$ or $\exists$ and y is a variable such that the following conditions hold.

1) y is substitutable for x in ψ, and
2) y does not have a free occurrence in ψ.

Then by replacing the subformula $(Qx)\psi$ with $(Qy)\psi(x/y)$ we obtain a *variant* of φ *in subformula* $(Qx)\psi$. After variation of one or more subformulas in φ we obtain a *variant* of φ. *For example,*

$(\exists x)(\forall y)(x \leq y)$	*is a formula* φ,
$(\exists u)(\forall v)(u \leq v)$	*is a variant of* φ,
$(\exists y)(\forall y)(y \leq y)$	*is not a variant of* φ, 1) *does not hold*,
$(\exists x)(\forall x)(x \leq x)$	*is not a variant of* φ, 2) *does not hold*.