Propositional and Predicate Logic - XI

Petr Gregor

KTIML MFF UK

WS 2022/2023

Resolution method in predicate logic - introduction

- A refutation procedure - its aim is to show that a given formula (or theory) is unsatisfiable.
- It assumes open formulas in CNF (and in clausal form).

  A *literal* is *(now)* an atomic formula or its negation.

  A *clause* is a finite set of literals, $\square$ denotes the empty clause.

  A *formula (in clausal form)* is a (possibly infinite) set of clauses.

  Remark *Every formula (theory) can be converted to an equisatisfiable open formula (theory) in CNF, and then to a formula in clausal form.*
- The resolution rule is more general - it allows to resolve through literals that are unifiable.
- Resolution in predicate logic is based on resolution in propositional logic and unification.

Local scope of variables

Variables can be renamed locally within clauses.

Let φ be an *(input)* open formula in CNF.

- φ is satisfiable if and only if its universal closure φ' is satisfiable.
- For every two formulas ψ, χ and a variable x

$$\models \quad (\forall x)(\psi \land \chi) \leftrightarrow (\forall x)\psi \land (\forall x)\chi$$

(also in the case that x is free both in ψ and χ).

- Every clause in φ can thus be replaced by its universal closure.
- We can then take any variants of clauses (to rename variables apart).

For example, by renaming variables in the second clause of (1) we obtain an equisatisfiable formula (2).

(1) $\{\{P(x), Q(x,y)\}, \{\neg P(x), \neg Q(y,x)\}\}$

(2) $\{\{P(x), Q(x,y)\}, \{\neg P(v), \neg Q(u,v)\}\}$

Reduction to propositional level (grounding)

Herbrand's theorem gives us the following (inefficient) method.

- Let S be the *(input)* formula in clausal form.
- We can assume that the language contains at least one constant symbol.
- Let S' be the set of all ground instances of all clauses from S.
- By introducing propositional letters representing atomic sentences we may view S' as a (possibly infinite) propositional formula in clausal form.
- We may verify that it is unsatisfiable by resolution on propositional level.

For example, for $S = \{\{P(x,y), R(x,y)\}, \{\neg P(c,y)\}, \{\neg R(x, f(x))\}\}$ *the set*

$$S' = \{\{P(c,c), R(c,c)\}, \{P(c,f(c)), R(c,f(c))\}, \{P(f(c),f(c)), R(f(c),f(c))\} \dots, \{\neg P(c,c)\}, \{\neg P(c,f(c))\}, \dots, \{\neg R(c,f(c))\}, \{\neg R(f(c),f(f(c)))\}, \dots\}$$

is unsatisfiable since on propositional level

$$S' \supseteq \{\{P(c,f(c)), R(c,f(c))\}, \{\neg P(c,f(c))\}, \{\neg R(c,f(c))\}\} \vdash_R \square.$$

Substitutions - examples

It is more efficient to use suitable substitutions. For example, in

$a)$ $\{P(x), Q(x,a)\}$, $\{\neg P(y), \neg Q(b,y)\}$ substituting x/b, y/a gives $\{P(b), Q(b,a)\}$, $\{\neg P(a), \neg Q(b,a)\}$, which resolves to $\{P(b), \neg P(a)\}$.

Or, substituting x/y and resolving through $P(y)$ gives $\{Q(y,a), \neg Q(b,y)\}$.

$b)$ $\{P(x), Q(x,a), Q(b,y)\}$, $\{\neg P(v), \neg Q(u,v)\}$ substituting x/b, y/a, u/b, v/a gives $\{P(b), Q(b,a)\}$, $\{\neg P(a), \neg Q(b,a)\}$, resolving to $\{P(b), \neg P(a)\}$.

$c)$ $\{P(x), Q(x,z)\}$, $\{\neg P(y), \neg Q(f(y),y)\}$ substituting $x/f(z)$, y/z gives $\{P(f(z)), Q(f(z),z)\}$, $\{\neg P(z), \neg Q(f(z),z)\}$, resolving to $\{P(f(z)), \neg P(z)\}$.

Alternatively, substituting $x/f(a)$, y/a, z/a gives $\{P(f(a)), Q(f(a),a)\}$, $\{\neg P(a), \neg Q(f(a),a)\}$, which resolves to $\{P(f(a)), \neg P(a)\}$. But the previous substitution is more general.

Substitutions

- A *substitution* is a (finite) set $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$, where x_i's are distinct variables, t_i's are terms, and the term t_i is not x_i.
- If all t_i's are ground terms, then σ is a *ground substitution*.
- If all t_i's are distinct variables, then σ is a *renaming of variables*.
- An *expression* is a literal or a term.
- An *instance* of an expression E *by substitution* $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$ is the expression $E\sigma$ obtained from E by simultaneous replacing all occurrences of all x_i's for t_i's, respectively.
- For a set S of expressions, let $S\sigma = \{E\sigma \mid E \in S\}$.

Remark Since we substitute for all variables simultaneously, a possible occurrence of x_i in t_j does not lead to a chain of substitutions.

For example, for $S = \{P(x), R(y,z)\}$ and $\sigma = \{x/f(y,z), y/x, z/c\}$ we have

$$S\sigma = \{P(f(y,z)), R(x,c)\}.$$

Composing substitutions

For substitutions $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$ and $\tau = \{y_1/s_1, \ldots, y_n/s_n\}$ we define

$$\sigma\tau = \{x_i/t_i\tau \mid x_i \in X,\ t_i\tau \text{ is not } x_i\} \cup \{y_j/s_j \mid y_j \in Y \setminus X\}$$

to be the *composition* of σ and τ, where $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_m\}$.

For example, for $\sigma = \{x/f(y), w/v\}$, $\tau = \{x/a, y/g(x), v/w, u/c\}$ *we have* $\sigma\tau = \{x/f(g(x)), y/g(x), v/w, u/c\}$.

Proposition (without proof) *For every expression* E *and substitutions* σ, τ, ϱ,

(i) $(E\sigma)\tau = E(\sigma\tau)$,

(ii) $(\sigma\tau)\varrho = \sigma(\tau\varrho)$.

Remark *Composition of substitutions is not commutative, for the above* σ, τ,

$$\tau\sigma = \{x/a, y/g(f(y)), u/c, w/v\} \neq \sigma\tau.$$

Unification

Let $S = \{E_1, \dots, E_n\}$ be a (finite) set of expressions.

- A *unification* of S is a substitution σ such that $E_1\sigma = E_2\sigma = \cdots = E_n\sigma$, i.e. $S\sigma$ is a singleton.
- S is *unifiable* if it has a unification.
- A unification σ of S is a *most general unification (mgu)* if for every unification τ of S there is a substitution λ such that $\tau = \sigma\lambda$.

For example, $S = \{P(f(x), y), P(f(a), w)\}$ *is unifiable by a most general* unification $\sigma = \{x/a, y/w\}$. A unification $\tau = \{x/a, y/b, w/b\}$ is obtained as $\sigma\lambda$ for $\lambda = \{w/b\}$. τ is not mgu, it cannot give us $\varrho = \{x/a, y/c, w/c\}$.

Observation *If σ, τ are two most general unifications of S, they differ only in renaming of variables.*

Unification algorithm

Let S be a (finite) nonempty set of expressions and p be the leftmost position in which some expressions of S differ. Then the *difference* in S is the set $D(S)$ of subexpressions of all expressions from S starting at the position p.

For example, $S = \{P(x,y), P(f(x),z), P(z,f(x))\}$ *has* $D(S) = \{x, f(x), z\}$.

Input Nonempty (finite) set of expressions S.

Output A most general unification σ of S or *"S is not unifiable"*.

(0) Let $S_0 := S$, $\sigma_0 := \emptyset$, $k := 0$. *(initialization)*

(1) If S_k is a singleton, output the substitution $\sigma = \sigma_0\sigma_1 \cdots \sigma_k$. *(mgu of S)*

(2) Find if $D(S_k)$ contains a variable x and a term t with no occurrence of x.

(3) If not, output *"S is not unifiable"*.

(4) Otherwise, let $\sigma_{k+1} := \{x/t\}$, $S_{k+1} := S_k\sigma_{k+1}$, $k := k+1$ and go to (1).

Remark *The occurrence check of x in t in step (2) can be "expensive".*

Unification algorithm - an example

$$S = \{P(f(y, g(z)), h(b)),\ P(f(h(w), g(a)), t),\ P(f(h(b), g(z)), y)\}$$

1) $S_0 = S$ is not a singleton and $D(S_0) = \{y, h(w), h(b)\}$ has a term $h(w)$ and a variable y not occurring in $h(w)$. Let $\sigma_1 = \{y/h(w)\}$, $S_1 = S_0\sigma_1$, i.e.
$$S_1 = \{P(f(h(w), g(z)), h(b)),\ P(f(h(w), g(a)), t),\ P(f(h(b), g(z)), h(w))\}.$$

2) $D(S_1) = \{w, b\}$, $\sigma_2 = \{w/b\}$, $S_2 = S_1\sigma_2$, i.e.
$$S_2 = \{P(f(h(b), g(z)), h(b)),\ P(f(h(b), g(a)), t)\}.$$

3) $D(S_2) = \{z, a\}$, $\sigma_3 = \{z/a\}$, $S_3 = S_2\sigma_3$, i.e.
$$S_3 = \{P(f(h(b), g(a)), h(b)),\ P(f(h(b), g(a)), t)\}.$$

4) $D(S_3) = \{h(b), t\}$, $\sigma_4 = \{t/h(b)\}$, $S_4 = S_3\sigma_4$, i.e.
$$S_4 = \{P(f(h(b), g(a)), h(b))\}.$$

5) S_4 is a singleton and a most general unification of S is
$$\sigma = \{y/h(w)\}\{w/b\}\{z/a\}\{t/h(b)\} = \{y/h(b), w/b, z/a, t/h(b)\}.$$

Unification algorithm - correctness

Proposition *The unification algorithm outputs a correct answer in finite time for any input S, i.e. a most general unification σ of S or it detects that S is not unifiable.* $(*)$ *Moreover, for every unification τ of S it holds that $\tau = \sigma\tau$.*

Proof It eliminates one variable in each round, so it ends in finite time.

- If it ends negatively after k rounds, $D(S_k)$ is not unifiable, thus also S.
- If it outputs $\sigma = \sigma_0\sigma_1 \cdots \sigma_k$, clearly σ is a unification of S.
- If we show the property $(*)$ for σ, then σ is a most general unification of S.

(1) Let τ be a unification of S. We show that $\tau = \sigma_0\sigma_1 \cdots \sigma_i\tau$ for all $i \leq k$.

(2) For $i = 0$ it holds. Let $\sigma_{i+1} = \{x/t\}$ and assume that $\tau = \sigma_0\sigma_1 \cdots \sigma_i\tau$.

(3) It suffices to show that $v\sigma_{i+1}\tau = v\tau$ for every variable v.

(4) If $v \neq x$, $v\sigma_{i+1} = v$, so (3) holds. Otherwise $v = x$ and $v\sigma_{i+1} = x\sigma_{i+1} = t$.

(5) Since τ unifies $S_i = S\sigma_0\sigma_1 \cdots \sigma_i$ and both the variable x and the term t are in $D(S_i)$, τ has to unify x and t, i.e. $t\tau = x\tau$, as required for (3). □

The general resolution rule

Let C_1, C_2 be clauses with distinct variables such that

$$C_1 = C_1' \sqcup \{A_1, \ldots, A_n\}, \quad C_2 = C_2' \sqcup \{\neg B_1, \ldots, \neg B_m\},$$

where $S = \{A_1, \ldots, A_n, B_1, \ldots, B_m\}$ is unifiable and $n, m \geq 1$. Then the clause

$$C = C_1'\sigma \cup C_2'\sigma,$$

where σ is a most general unification of S, is the *resolvent* of C_1 and C_2.

For example, in clauses $\{P(x), Q(x,z)\}$ *and* $\{\neg P(y), \neg Q(f(y), y)\}$ *we can* unify $S = \{Q(x,z), Q(f(y), y)\}$ applying a most general unification $\sigma = \{x/f(y), z/y\}$, and then resolve to a clause $\{P(f(y)), \neg P(y)\}$.

Remark *The condition on distinct variables can be satisfied by renaming variables apart. This is sometimes necessary, e.g. from* $\{\{P(x)\}, \{\neg P(f(x))\}\}$ *after renaming we can get* $\square$*, but* $\{P(x), P(f(x))\}$ *is not unifiable.*

Resolution proof

We have the same notions as in propositional logic, up to renaming variables.

- *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 $i \le n$, we have $C_i = C_i'\sigma$ for some $C_i' \in S$ and a renaming of variables σ, 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 a formula S is a resolution proof of $\square$ from S.
- S is (resolution) *refutable* if $S \vdash_R \square$.

Remark *Elimination of several literals at once is sometimes necessary, e.g.* $S = \{\{P(x), P(y)\}, \{\neg P(x), \neg P(y)\}\}$ *is resolution refutable, but it has no refutation that eliminates only a single literal in each resolution step.*

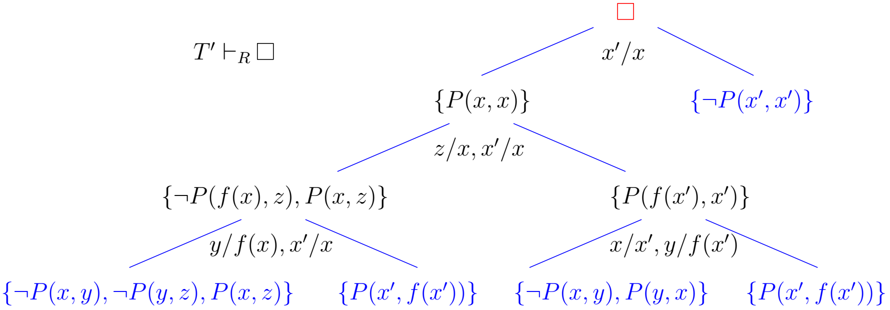
Resolution in predicate logic - an example

Consider $T = \{\neg P(x,x),\ P(x,y) \rightarrow P(y,x),\ P(x,y) \wedge P(y,z) \rightarrow P(x,z)\}$.

Is $T \models (\exists x)\neg P(x, f(x))$? Equivalently, is the following T' unsatisfiable?

$T' = \{\{\neg P(x,x)\}, \{\neg P(x,y), P(y,x)\}, \{\neg P(x,y), \neg P(y,z), P(x,z)\}, \{P(x, f(x))\}\}$

$T' \vdash_R \square$



Soundness of resolution

First we show soundness of the general resolution rule.

Proposition Let C be a resolvent of clauses C_1, C_2. For every L-structure $\mathcal{A}$,

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

Proof Let $C_1 = C_1' \sqcup \{A_1, \ldots, A_n\}$, $C_2 = C_2' \sqcup \{\neg B_1, \ldots, \neg B_m\}$, σ be a most general unification for $S = \{A_1, \ldots, A_n, B_1, \ldots, B_m\}$, and $C = C_1'\sigma \cup C_2'\sigma$.

- Since C_1, C_2 are open, it holds also $\mathcal{A} \models C_1\sigma$ and $\mathcal{A} \models C_2\sigma$.
- We have $C_1\sigma = C_1'\sigma \cup \{S\sigma\}$ and $C_2\sigma = C_2'\sigma \cup \{\neg(S\sigma)\}$.
- We show $\mathcal{A} \models C[e]$ for every e. If $\mathcal{A} \models S\sigma[e]$, then $\mathcal{A} \models C_2'\sigma[e]$, and thus $\mathcal{A} \models C[e]$. Otherwise $\mathcal{A} \not\models S\sigma[e]$, so $\mathcal{A} \models C_1'\sigma[e]$, and thus $\mathcal{A} \models C[e]$. $\square$

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

Proof Let $S \vdash_R \square$. Suppose $\mathcal{A} \models S$ for some structure $\mathcal{A}$. By soundness of the general resolution rule we have $\mathcal{A} \models \square$, which is impossible. $\blacksquare$

Lifting lemma

A resolution proof on propositional level can be "lifted" to predicate level.

Lemma *Let $C_1^* = C_1\tau_1$, $C_2^* = C_2\tau_2$ be ground instances of clauses C_1, C_2 with distinct variables and C^* be a resolvent of C_1^* a C_2^*. Then there exists a resolvent C of C_1 and C_2 such that $C^* = C\tau_1\tau_2$ is a ground instance of C.*

Proof Assume that C^* is a resolvent of C_1^*, C_2^* through a literal $P(t_1, \ldots, t_k)$.

- We have $C_1 = C_1' \sqcup \{A_1, \ldots, A_n\}$ and $C_2 = C_2' \sqcup \{\neg B_1, \ldots, \neg B_m\}$, where $\{A_1, \ldots, A_n\}\tau_1 = \{P(t_1, \ldots, t_k)\}$ and $\{\neg B_1, \ldots, \neg B_m\}\tau_2 = \{\neg P(t_1, \ldots, t_k)\}$
- Thus $(\tau_1\tau_2)$ unifies $S = \{A_1, \ldots, A_n, B_1, \ldots, B_m\}$ and if σ is mgu of S from the unification algorithm, then $C = C_1'\sigma \cup C_2'\sigma$ is a resolvent of C_1, C_2.
- Moreover, $(\tau_1\tau_2) = \sigma(\tau_1\tau_2)$ by the property $(*)$ for σ, and hence

$$\begin{aligned} C\tau_1\tau_2 &= (C_1'\sigma \cup C_2'\sigma)\tau_1\tau_2 = C_1'\sigma\tau_1\tau_2 \cup C_2'\sigma\tau_1\tau_2 = C_1'\tau_1 \cup C_2'\tau_2 \\ &= (C_1 \setminus \{A_1, \ldots, A_n\})\tau_1 \cup (C_2 \setminus \{\neg B_1, \ldots, \neg B_m\})\tau_2 \\ &= (C_1^* \setminus \{P(t_1, \ldots, t_k)\}) \cup (C_2^* \setminus \{\neg P(t_1, \ldots, t_k)\}) = C^*. \quad \square \end{aligned}$$

Completeness

Corollary *Let S' be the set of all ground instances of clauses of formula S. If $S' \vdash_R C'$ (on prop. level) where C' is a ground clause, then $C' = C\sigma$ for some clause C and a ground substitution σ such that $S \vdash_R C$ (on pred. level).*

Proof By induction on the length of resolution proof using lifting lemma. $\square$

Theorem (completeness) *If S is unsatisfiable, then $S \vdash_R \square$.*

Proof If S is unsatisfiable, then by the (corollary of) Herbrand's theorem, also the set S' of all ground instances of clauses of S is unsatisfiable.

- By completeness of resolution in prop. logic, $S' \vdash_R \square$ (on prop. level).
- By the above corollary, there is a clause C and a ground substitution σ such that $\square = C\sigma$ and $S \vdash_R C$ (on pred. level).
- The only clause that has $\square$ as a ground instance is the clause $C = \square$. $\blacksquare$

Linear resolution

Resolution can be significantly refined (without loss of completeness).

- A *linear proof* of a clause C from a formula S is a finite sequence of pairs $(C_0, B_0), \ldots, (C_n, B_n)$ s.t. C_0 is a variant of a clause from S and for $i \leq n$
  - *i*) B_i is a variant of a clause from S or $B_i = C_j$ for some $j < i$,
  - *ii*) C_{i+1} is a resolvent of C_i and B_i, and $C_{n+1} = C$.
- 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 $S \vdash_L \square$.

Theorem *S is linearly refutable if and only if S is unsatisfiable.*

Proof $(\Rightarrow)$ Every linear proof can be transformed to a resolution proof.
$(\Leftarrow)$ Follows from completeness of linear resolution in prop. logic (omitted) since the lifting lemma preserves linearity of resolution proofs. $\square$

LI-resolution

For Horn formulas we can refine the linear resolution further.

- *LI-resolution* *("linear input")* from a formula S is a linear resolution where each side clause B_i is a variant of a clause from the (input) formula S,
- $S \vdash_{LI} C$ denotes that C is provable by LI-resolution from S,
- a *Horn formula* is a set (possibly infinite) of Horn clauses,
- a *Horn clause* is a clause containing at most one positive literal,
- a *fact* is a (Horn) clause with exactly one positive and no negative literal,
- a *rule* is a (Horn) clause with exactly one positive and at least one negative literal, rules and facts are called *program clauses*,
- a *goal* is a nonempty (Horn) clause without positive literals.

Theorem *If a Horn formula T is satisfiable and $T \cup \{G\}$ is unsatisfiable for a goal G, then $T \cup \{G\}$ can be refuted by LI-resolution starting with clause G.*

Proof Follows by Herbrand's theorem, the same statement in prop. logic and the lifting lemma. $\square$

Program in Prolog

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

$son(X,Y) :\!-\ father(Y,X), man(X).$	$\{son(X,Y), \neg father(Y,X), \neg man(X)\}$
$son(X,Y) :\!-\ mother(Y,X), man(X).$	$\{son(X,Y), \neg mother(Y,X), \neg man(X)\}$
$man(jan).$	$\{man(jan)\}$
$father(jiri, jan).$	$\{father(jiri, jan)\}$
$mother(julie, jan).$	$\{mother(julie, jan)\}$
$?\!-\ son(jan, X)$ $\quad P \models (\exists X) son(jan, X)$?	$\{\neg son(jan, X)\}$

We are interested whether a given existential query holds in a given program.

Corollary *For a program P and a goal $G = \{\neg A_1, \ldots, \neg A_n\}$ in var. $X_1, \ldots, X_m$*

(1) $P \models (\exists X_1) \ldots (\exists X_m)(A_1 \wedge \ldots \wedge A_n)$, *if and only if*

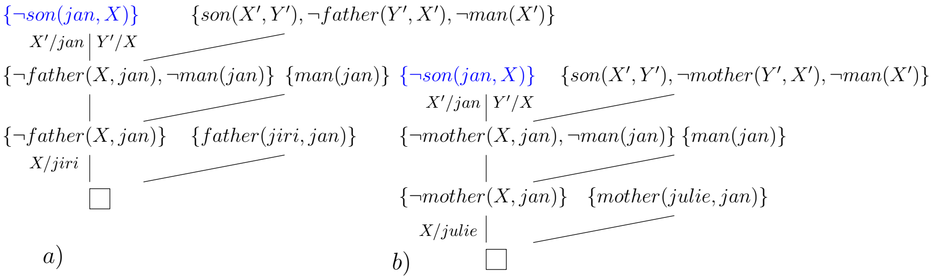
(2) $P \cup \{G\}$ *can be refuted by LI-resolution starting with (a variant of) G.*

LI-resolution over a program

If the answer is positive, we want to know the output substitution.

The *output substitution* σ of a LI-refutation from $P \cup \{G\}$ starting with a goal $G = \{\neg A_1, \ldots, \neg A_n\}$ is a composition of mgu's in all steps (restricted only to variables in G). It holds that

$$P \models (A_1 \wedge \ldots \wedge A_n)\sigma.$$



The output substitutions $a)\ X = jiri,\ \ b)\ X = julie$.

Hilbert's calculus in predicate logic

- basic connectives and quantifier: $\neg$, $\rightarrow$, $(\forall x)$ (others are derived)
- allows to prove any formula (not just sentences)
- *logical axioms* (schemes of axioms):

$$
\begin{array}{rll}
(i) & \varphi \rightarrow (\psi \rightarrow \varphi) & \\
(ii) & (\varphi \rightarrow (\psi \rightarrow \chi)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \chi)) & \\
(iii) & (\neg\varphi \rightarrow \neg\psi) \rightarrow (\psi \rightarrow \varphi) & \\
(iv) & (\forall x)\varphi \rightarrow \varphi(x/t) & \text{if } t \text{ is substitutable for } x \text{ to } \varphi \\
(v) & (\forall x)(\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow (\forall x)\psi) & \text{if } x \text{ is not free in } \varphi
\end{array}
$$

  where φ, ψ, χ are any formulas (of a given language), t is any term, and x is any variable
- in a language with equality we include also the axioms of equality
- *rules of inference*

$$\frac{\varphi,\ \varphi \rightarrow \psi}{\psi} \quad \text{(modus ponens)}, \qquad \frac{\varphi}{(\forall x)\varphi} \quad \text{(generalization)}$$

Hilbert-style proofs

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.

A formula φ is *provable* from T if it has a proof from T, denoted by $T \vdash_H \varphi$.

Theorem (soundness) *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 $T \models \varphi \rightarrow \psi$, then $T \models \psi$, i.e. modus ponens is sound,
- if $T \models \varphi$, then $T \models (\forall x)\varphi$, i.e. generalization is sound,
- thus every formula in a proof from T is valid in T. $\square$

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