

IAoo8: Computational Logic

2. First-Order Logic

Achim Blumensath blumens@fi.muni.cz

Faculty of Informatics, Masaryk University, Brno

Basic Concepts

First-Order Logic

Syntax

- ▶ Variables x, y, z, \dots
- ▶ Terms $x, f(t_0, \dots, t_n)$
- ▶ Relations $R(t_0, \dots, t_n)$ and equality $t_0 = t_1$
- ▶ Operators $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$
- ▶ Quantifiers $\exists x\varphi, \forall x\varphi$

Semantics

$$\mathfrak{A} \models \varphi(\bar{a}) \quad \mathfrak{A} = \langle A, R_0, R_1, \dots, f_0, f_1, \dots \rangle$$

Examples

$$\varphi := \forall x \exists y [f(y) = x],$$

$$\psi := \forall x \forall y \forall z [x \leq y \wedge y \leq z \rightarrow x \leq z].$$

Normal Forms

Prenex normal form

$Q_0x_0 \cdots Q_nx_n\psi(\vec{x})$, ψ quantifier-free

Normal Forms

Prenex normal form

$Q_0x_0 \cdots Q_nx_n \psi(\bar{x})$, ψ quantifier-free

Skolem normal form

Eliminate **existential quantifiers**:

replace $\forall \bar{x} \exists y \varphi(\bar{x}, y)$ by $\forall \bar{x} \varphi(\bar{x}, f(\bar{x}))$ (f new symbol).

Example

$\forall x \exists y \exists z [y > x \wedge z < x]$

Normal Forms

Prenex normal form

$Q_0x_0 \cdots Q_nx_n \psi(\bar{x})$, ψ quantifier-free

Skolem normal form

Eliminate **existential quantifiers**:

replace $\forall \bar{x} \exists y \varphi(\bar{x}, y)$ by $\forall \bar{x} \varphi(\bar{x}, f(\bar{x}))$ (f new symbol).

Example

$\forall x \exists y \exists z [y > x \wedge z < x]$ $\forall x [f(x) > x \wedge g(x) < x]$

Normal Forms

Prenex normal form

$Q_0x_0\cdots Q_nx_n\psi(\bar{x})$, ψ quantifier-free

Skolem normal form

Eliminate **existential quantifiers**:

replace $\forall\bar{x}\exists y\varphi(\bar{x},y)$ by $\forall\bar{x}\varphi(\bar{x},f(\bar{x}))$ (f new symbol).

Example

$\forall x\exists y\exists z[y > x \wedge z < x]$ $\forall x[f(x) > x \wedge g(x) < x]$
 $\exists x\forall y[y + 1 \neq x]$

Normal Forms

Prenex normal form

$Q_0x_0\cdots Q_nx_n\psi(\bar{x})$, ψ quantifier-free

Skolem normal form

Eliminate **existential quantifiers**:

replace $\forall\bar{x}\exists y\varphi(\bar{x},y)$ by $\forall\bar{x}\varphi(\bar{x},f(\bar{x}))$ (f new symbol).

Example

$$\forall x\exists y\exists z[y > x \wedge z < x]$$

$$\exists x\forall y[y + 1 \neq x]$$

$$\forall x[f(x) > x \wedge g(x) < x]$$

$$\forall y[y + 1 \neq c]$$

Normal Forms

Prenex normal form

$Q_0x_0\cdots Q_nx_n\psi(\bar{x})$, ψ quantifier-free

Skolem normal form

Eliminate **existential quantifiers**:

replace $\forall\bar{x}\exists y\varphi(\bar{x},y)$ by $\forall\bar{x}\varphi(\bar{x},f(\bar{x}))$ (f new symbol).

Example

$\forall x\exists y\exists z[z > x \wedge z < x]$ $\forall x[f(x) > x \wedge g(x) < x]$

$\exists x\forall y[y + 1 \neq x]$ $\forall y[y + 1 \neq c]$

$\exists x\forall y\exists z\forall u\exists v[R(x, y, z, u, v)]$

Normal Forms

Prenex normal form

$Q_0x_0\cdots Q_nx_n\psi(\bar{x})$, ψ quantifier-free

Skolem normal form

Eliminate **existential quantifiers**:

replace $\forall\bar{x}\exists y\varphi(\bar{x},y)$ by $\forall\bar{x}\varphi(\bar{x},f(\bar{x}))$ (f new symbol).

Example

| | |
|---|---|
| $\forall x\exists y\exists z[z > x \wedge z < x]$ | $\forall x[f(x) > x \wedge g(x) < x]$ |
| $\exists x\forall y[y + 1 \neq x]$ | $\forall y[y + 1 \neq c]$ |
| $\exists x\forall y\exists z\forall u\exists v[R(x, y, z, u, v)]$ | $\forall y\forall u[R(c, y, f(y), u, g(y, z))]$ |

Normal Forms

Prenex normal form

$Q_0x_0 \cdots Q_nx_n\psi(\bar{x})$, ψ quantifier-free

Skolem normal form

Eliminate **existential quantifiers**:

replace $\forall\bar{x}\exists y\varphi(\bar{x}, y)$ by $\forall\bar{x}\varphi(\bar{x}, f(\bar{x}))$ (f new symbol).

Example

| | |
|---|---|
| $\forall x\exists y\exists z[y > x \wedge z < x]$ | $\forall x[f(x) > x \wedge g(x) < x]$ |
| $\exists x\forall y[y + 1 \neq x]$ | $\forall y[y + 1 \neq c]$ |
| $\exists x\forall y\exists z\forall u\exists v[R(x, y, z, u, v)]$ | $\forall y\forall u[R(c, y, f(y), u, g(y, z))]$ |

Theorem

Let φ_s be a Skolemisation of φ . Then φ_s is satisfiable iff φ is satisfiable.

Theorem of Herbrand

Theorem of Herbrand

A formula $\exists \bar{x} \varphi(\bar{x})$ is valid if, and only if, there are terms $\bar{t}_0, \dots, \bar{t}_n$ such that the disjunction $\bigvee_{i \leq n} \varphi(\bar{t}_i)$ is valid.

Corollary

A formula $\forall \bar{x} \varphi(\bar{x})$ is unsatisfiable if, and only if, there are terms $\bar{t}_0, \dots, \bar{t}_n$ such that the conjunction $\bigwedge_{i \leq n} \varphi(\bar{t}_i)$ is unsatisfiable.

Substitution

Definition

A **substitution** σ is a function that replaces in a formula every free variable by a term (and renames bound variables if necessary).

Instead of $\sigma(\varphi)$ we also write $\varphi[x \mapsto s, y \mapsto t]$ if $\sigma(x) = s$ and $\sigma(y) = t$.

Examples

$$\begin{aligned}(x = f(y))[x \mapsto g(x), y \mapsto c] &= g(x) = f(c) \\ \exists z(x = z + z)[x \mapsto z] &= \exists u(z = u + u)\end{aligned}$$

Unification

Definition

A **unifier** of two terms $s(\bar{x})$ and $t(\bar{x})$ is a pair of substitution σ, τ such that $\sigma(s) = \tau(t)$.

A unifier σ, τ is **most general** if every other unifier σ', τ' can be written as $\sigma' = \rho \circ \sigma$ and $\tau' = \nu \circ \tau$, for some ρ, ν .

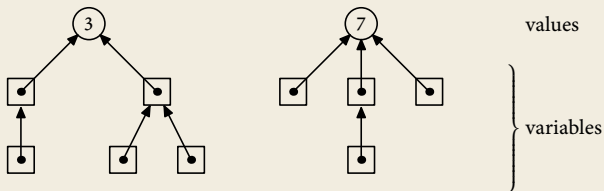
Examples

| | | | |
|------------------|---------------|--------------------------|---------------------|
| $s = f(x, g(x))$ | $t = f(c, x)$ | $x \mapsto c$ | $x \mapsto g(c)$ |
| $s = f(x, g(x))$ | $t = f(x, y)$ | $x \mapsto x$ | $x \mapsto x$ |
| | | | $y \mapsto g(x)$ |
| | | $x \mapsto g(x)$ | $x \mapsto g(x)$ |
| | | | $y \mapsto g(g(x))$ |
| $s = f(x)$ | $t = g(x)$ | unification not possible | |

Unification Algorithm

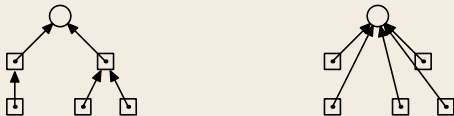
```
unify( $s, t$ )  
  if  $s$  is a variable  $x$  then  
    set  $x$  to  $t$   
  else if  $t$  is a variable  $x$  then  
    set  $x$  to  $s$   
  else  $s = f(\bar{u})$  and  $t = g(\bar{v})$   
    if  $f = g$  then  
      forall  $i$  unify( $u_i, v_i$ )  
    else  
      fail
```

Union-Find-Algorithm



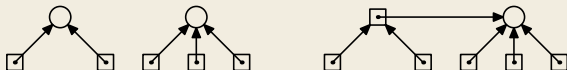
find : *variable* \rightarrow *value*

- ▶ follows pointers to the root and creates shortcuts



union : (*variable* \times *variable*) \rightarrow *unit*

- ▶ links roots by a pointer



Resolution

Clauses

Definitions

- ▶ **literal** $R(\bar{t})$ or $\neg R(\bar{t})$
- ▶ **clause** set of literals $\{P(\bar{s}), R(\bar{t}), \neg S(\bar{u})\}$

Clauses

Definitions

- ▶ **literal** $R(\bar{t})$ or $\neg R(\bar{t})$
- ▶ **clause** set of literals $\{P(\bar{s}), R(\bar{t}), \neg S(\bar{u})\}$

Example

$$\text{CNF} \quad \varphi := \forall x \forall y [R(x, y) \vee \neg R(x, f(x))] \wedge \forall y [\neg R(f(y), y) \vee P(y)]$$

(no existential quantifiers)

$$\text{clauses} \quad \{R(x, y) \vee \neg R(x, f(x))\}, \{\neg R(f(y), y), P(y)\}$$

Resolution

Resolution Step

Consider two clauses

$$C = \{P(\bar{s}), R_0(\bar{t}_0), \dots, R_m(\bar{t}_m)\}$$
$$C' = \{\neg P(\bar{s}'), S_0(\bar{u}_0), \dots, S_n(\bar{u}_n)\}$$

where \bar{s} and \bar{s}' have no common variables, and let σ, τ be the most general unifier of \bar{s} and \bar{s}' . The **resolvent** of C and C' is the clause

$$\{R_0(\sigma(\bar{t}_0)), \dots, R_m(\sigma(\bar{t}_m)), S_0(\tau(\bar{u}_0)), \dots, S_n(\tau(\bar{u}_n))\}.$$

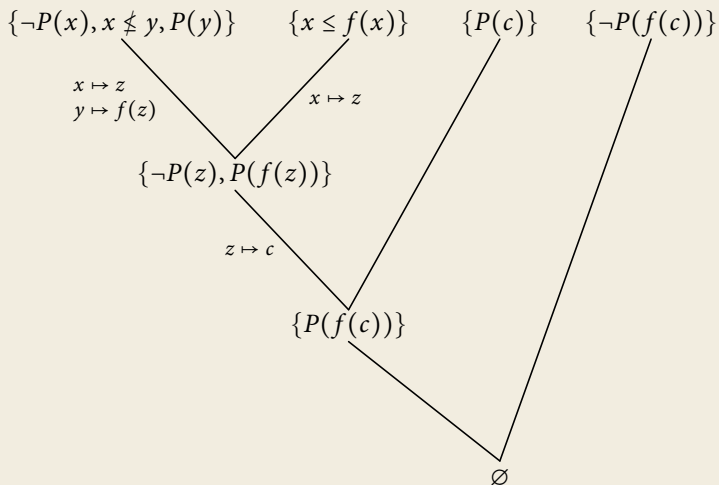
Lemma

Let C be the resolvent of two clauses in Φ . Then

$$\Phi \models \Phi \cup \{C\}.$$

Example

$$\varphi = \forall x \forall y [P(x) \wedge x \leq y \rightarrow P(y)] \wedge \forall x [x \leq f(x)] \wedge Pc \wedge \neg P(f(c))$$



The Resolution Method

Theorem

The resolution method for first-order logic (without equality) is **sound** and **complete**.

Theorem

Satisfiability for first-order logic is **undecidable**.

Proof

Turing machine $\mathcal{M} = \langle Q, \Sigma, \Delta, q_0, F_+, F_- \rangle$

Q set of states

Σ tape alphabet

Δ set of transitions $\langle p, a, b, m, q \rangle \in Q \times \Sigma \times \Sigma \times \{-1, 0, 1\} \times Q$

q_0 initial state

F_+ accepting states

F_- rejecting states

Proof

Turing machine $\mathcal{M} = \langle Q, \Sigma, \Delta, q_0, F_+, F_- \rangle$

Q set of states

Σ tape alphabet

Δ set of transitions $\langle p, a, b, m, q \rangle \in Q \times \Sigma \times \Sigma \times \{-1, 0, 1\} \times Q$

q_0 initial state

F_+ accepting states

F_- rejecting states

Encoding in FO

$S_q(t)$ **state** q at time t

$h(t)$ **head** in field $h(t)$ at time t

$W_a(t, k)$ **letter** a in field k at time t

s **successor** function $s(n) = n + 1$

$$\varphi_w := \text{ADM} \wedge \text{INIT} \wedge \text{TRANS} \wedge \text{ACC}$$

Proof

| | |
|-------------|--|
| $S_q(t)$ | state q at time t |
| $h(t)$ | head in field $h(t)$ at time t |
| $W_a(t, k)$ | letter a in field k at time t |
| s | successor function $s(n) = n + 1$ |

Admissibility formula

$$\text{ADM} := \forall t \bigwedge_{p \neq q} \neg [S_p(t) \wedge S_q(t)] \quad \text{unique state}$$

$$\wedge \forall t \forall k \bigwedge_{a \neq b} \neg [W_a(t, k) \wedge W_b(t, k)] \quad \text{unique letter}$$

Proof

| | |
|-------------|--|
| $S_q(t)$ | state q at time t |
| $h(t)$ | head in field $h(t)$ at time t |
| $W_a(t, k)$ | letter a in field k at time t |
| s | successor function $s(n) = n + 1$ |

Initialisation formula for input: $a_0 \dots a_{n-1}$

| | |
|--|-----------------------|
| $\text{INIT} := S_{q_0}(0)$ | initial state |
| $\wedge h(0) = 0$ | initial head position |
| $\wedge \bigwedge_{k < n} W_{a_k}(0, \underline{k}) \wedge \forall k [k \geq \underline{n} \rightarrow W_{\square}(0, k)]$ | initial tape content |

(here $\underline{k} := s(s(\dots s(0)))$)

Acceptance formula

| | |
|--|-----------------|
| $\text{ACC} := \exists t \bigvee_{q \in F_+} S_q(t)$ | accepting state |
|--|-----------------|

Proof

| | |
|-------------|--|
| $S_q(t)$ | state q at time t |
| $h(t)$ | head in field $h(t)$ at time t |
| $W_a(t, k)$ | letter a in field k at time t |
| s | successor function $s(n) = n + 1$ |

Transition formula

$$\begin{aligned} \text{TRANS} := & \forall t \bigvee_{\langle p, a, b, m, q \rangle \in \Delta} [S_p(t) \wedge W_a(t, h(t)) \wedge S_q(s(t)) \wedge \\ & h(s(t)) = h(t) + m \wedge W_b(s(t), h(t))] \\ & \wedge \forall t \forall k \bigwedge_{a \in \Sigma} [k \neq h(t) \rightarrow [W_a(t, k) \leftrightarrow W_a(s(t), k)]] \end{aligned}$$

where

$$h(s(t)) = h(t) + m := \begin{cases} h(s(t)) = s(h(t)) & \text{if } m = 1, \\ h(s(t)) = h(t) & \text{if } m = 0, \\ s(h(s(t))) = h(t) & \text{if } m = -1. \end{cases}$$

Linear Resolution and Horn Formulae

Horn formulae

A **Horn formulae** is a formula in CNF where each clause contains at most one positive literal.

Theorem

A set of Horn clauses is unsatisfiable if, and only if, one can use linear resolution to derive the empty clause from it.

SLD Resolution

Linear resolution where the clauses are **sequences** instead of sets and we always resolve the **leftmost literal** of the current clause.