# IA008: 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

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

## Examples

$$\varphi := \forall x \exists y [f(y) = x],$$
$$\psi := \forall x \forall y \forall z [x \leq y \wedge y \leq z \rightarrow x \leq z].$$

# Examples

### Structures

- graphs $\mathfrak{G} = \langle V, E \rangle$

  $E \subseteq V \times V$ binary relation

# Examples

## Structures

- graphs $\mathfrak{G} = \langle V, E \rangle$

  $E \subseteq V \times V$ binary relation

- words $\mathfrak{W} = \langle W, \leq, (P_a)_a \rangle$

  $\leq \subseteq W \times W$ linear ordering

  $P_a \subseteq W$ positions with letter $a$

# Examples

## Structures

- graphs $\mathfrak{G} = \langle V, E \rangle$

  $E \subseteq V \times V$ binary relation

- words $\mathfrak{W} = \langle W, \leq, (P_a)_a \rangle$

  $\leq \subseteq W \times W$ linear ordering

  $P_a \subseteq W$ positions with letter $a$

- transition systems $\mathfrak{S} = \langle S, (E_a)_a, (P_i)_i \rangle$

  $E_a \subseteq V \times V$ binary relation

  $P_i \subseteq V$ unary relation

# Examples

**Graphs** $\mathfrak{G} = \langle V, E \rangle,\ E \subseteq V \times V$

- 'The graph is undirected.' (i.e., $E$ is symmetric)

# Examples

**Graphs** $\quad \mathfrak{G} = \langle V, E\rangle,\ E \subseteq V \times V$

- 'The graph is undirected.' (i.e., $E$ is symmetric)

$$\forall x \forall y [E(x, y) \to E(y, x)]$$

# Examples

**Graphs** $\mathfrak{G} = \langle V, E \rangle, \ E \subseteq V \times V$

- 'The graph is undirected.' (i.e., $E$ is symmetric)

$$\forall x \forall y [E(x, y) \to E(y, x)]$$

- 'The graph has no isolated vertices.'

# Examples

**Graphs** $\quad \mathfrak{G} = \langle V, E \rangle,\ E \subseteq V \times V$

- 'The graph is undirected.' (i.e., $E$ is symmetric)

$$\forall x \forall y [E(x, y) \to E(y, x)]$$

- 'The graph has no isolated vertices.'

$$\forall x \exists y [E(x, y) \vee E(y, x)]$$

# Examples

**Graphs** $\mathfrak{G} = \langle V, E \rangle, \; E \subseteq V \times V$

- 'The graph is undirected.' (i.e., $E$ is symmetric)

$$\forall x \forall y [E(x, y) \to E(y, x)]$$

- 'The graph has no isolated vertices.'

$$\forall x \exists y [E(x, y) \vee E(y, x)]$$

- 'Every vertex has outdegree $1$.'

# Examples

**Graphs** $\mathfrak{G} = \langle V, E \rangle,\ E \subseteq V \times V$

- 'The graph is undirected.' (i.e., $E$ is symmetric)

$$\forall x \forall y [E(x, y) \to E(y, x)]$$

- 'The graph has no isolated vertices.'

$$\forall x \exists y [E(x, y) \vee E(y, x)]$$

- 'Every vertex has outdegree $1$.'

$$\forall x \exists y [E(x, y) \wedge \forall z [E(x, z) \to z = y]]$$

# Normal Forms

## Prenex normal form

$$Q_0 x_0 \cdots Q_n x_n \psi(\bar{x}), \qquad \psi \text{ quantifier-free}$$

# Normal Forms

### Prenex normal form

$$Q_0 x_0 \cdots Q_n x_n \psi(\bar{x}), \quad \psi \text{ quantifier-free}$$

### 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

$$Q_0 x_0 \cdots Q_n x_n \psi(\bar{x}), \quad \psi \text{ quantifier-free}$$

### 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

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

# Normal Forms

### Prenex normal form

$$Q_0 x_0 \cdots Q_n x_n \psi(\bar{x}), \quad \psi \text{ quantifier-free}$$

### 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

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

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

# Normal Forms

### Prenex normal form

$$Q_0 x_0 \cdots Q_n x_n \psi(\bar{x}), \quad \psi \text{ quantifier-free}$$

### 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

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

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

# Normal Forms

## Prenex normal form

$$Q_0 x_0 \cdots Q_n x_n \psi(\bar{x}), \quad \psi \text{ quantifier-free}$$

## 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

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

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

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

# Normal Forms

### Prenex normal form

$$Q_0 x_0 \cdots Q_n x_n \psi(\bar{x}), \quad \psi \text{ quantifier-free}$$

### 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

$$\begin{array}{ll}
\forall x \exists y \exists z [y > x \wedge z < x] & \forall x [f(x) > x \wedge g(x) < x] \\
\exists x \forall y [y + 1 \neq x] & \forall y [y + 1 \neq c] \\
\exists x \forall y \exists z \forall u \exists v [R(x, y, z, u, v)] & \forall y \forall u [R(c, y, f(y), u, g(y, u))]
\end{array}$$

# Normal Forms

## Prenex normal form

$$Q_0 x_0 \cdots Q_n x_n \psi(\bar{x}), \quad \psi \text{ quantifier-free}$$

## 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).

## Theorem

Let $\varphi_s$ be a Skolemisation of $\varphi$. Then $\varphi_s$ is satisfiable iff $\varphi$ 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 $\bar{t}_0, \ldots, \bar{t}_n$ 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 $\bar{t}_0, \ldots, \bar{t}_n$ such that the conjunction $\bigwedge_{i \leq n} \varphi(\bar{t}_i)$ is unsatisfiable.

# Resolution

# Substitution

## Definition

A **substitution** $\sigma$ 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

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

# Unification

### Definition

A **unifier** of two terms $s(\bar{x})$ and $t(\bar{x})$ is a pair of substitutions $\sigma, \tau$ such that $\sigma(s) = \tau(t)$.
A unifier $\sigma, \tau$ is **most general** if every other unifier $\sigma', \tau'$ can be written as $\sigma' = \rho \circ \sigma$ and $\tau' = \upsilon \circ \tau$, for some $\rho, \upsilon$.

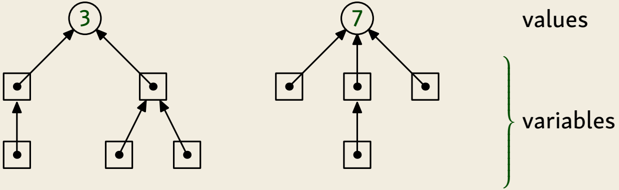
### Examples

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

# Unification Algorithm

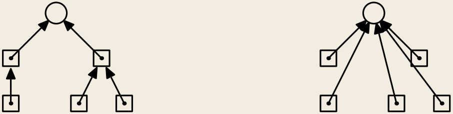
```
unify(s, t)
  if s is a variable x then
    if x already has some value u then
      unify(u, t)
    else
      set x to t
  else if t is a variable x then
    if x already has some value v then
      unify(s, v)
    else
      set x to s
  else s = f(ū) and t = g(v̄)
    if f = g then
      forall i   unify(u_i, v_i)
    else
      fail
```

# Union-Find-Algorithm



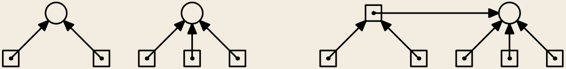

find : *variable* → *value*

- follows pointers to the root and creates shortcuts



union : (*variable* × *variable*) → *unit*

- links roots by a pointer



# 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**

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

(no existential quantifiers)

clauses $\big\{R(x,y)\,, \neg R(x, f(x))\big\}$, $\big\{\neg R(f(y), y)\,, P(y)\big\}$

# Resolution

### Resolution Step

Consider two clauses

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

where $\bar{s}$ and $\bar{s}'$ have no common variables, and let $\sigma, \tau$ 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 $\Phi$. Then

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

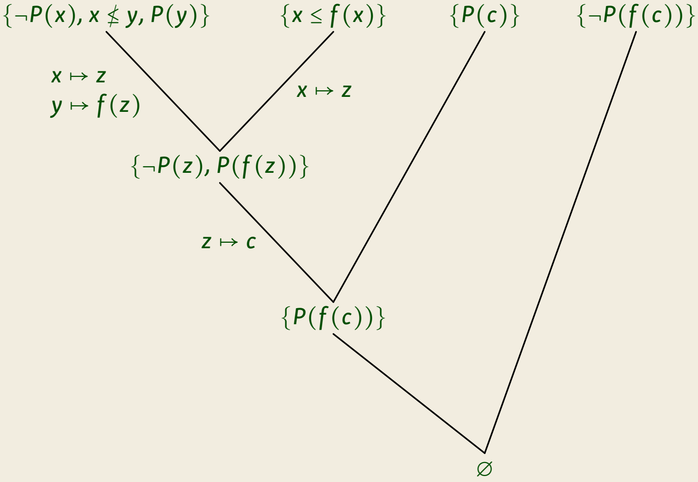
# Example

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

$\{\neg P(x), x \nleq y, P(y)\}$ $\{x \leq f(x)\}$ $\{P(c)\}$ $\{\neg P(f(c))\}$

# 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.**

# Satisfiability

**Theorem**

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

# Proof

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

| | |
|---|---|
| $Q$ | set of states |
| $\Sigma$ | tape alphabet |
| $\Delta$ | 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 |

By adding a counter to $\mathcal{M}$ we may assume that every run of $\mathcal{M}$ terminates.

# Proof

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

| | |
|---|---|
| $Q$ | set of states |
| $\Sigma$ | tape alphabet |
| $\Delta$ | 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 $s(n) = n + 1$ |
| $0$ | **zero** |

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

# 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 $s(n) = n + 1$ |
| $0$ | **zero** |

**Admissibility formula**

$$\begin{aligned} \mathrm{ADM} := \forall t \bigwedge_{p \neq q} \neg[S_p(t) \wedge S_q(t)] & \qquad \text{unique state} \\ \wedge\, \forall t \forall k \bigwedge_{a \neq b} \neg[W_a(t,k) \wedge W_b(t,k)] & \qquad \text{unique letter} \end{aligned}$$

# 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: $a_0 \dots a_{n-1}$

$$\begin{aligned} \mathrm{INIT} := S_{q_0}(0) & \qquad \text{initial state} \\ \wedge\, h(0) = 0 & \qquad \text{initial head position} \\ \wedge \bigwedge_{k<n} W_{a_k}(0, \underline{k}) \wedge \forall k W_{\Box}(0, k+n)] & \qquad \text{initial tape content} \end{aligned}$$

(here $\underline{k} := s(s(\cdots s(0)))$ and $k+n := s^n(k)$)

**Acceptance formula**

$$\mathrm{ACC} := \forall t \bigwedge_{q \in F_-} \neg S_q(t) \qquad \text{no rejecting states}$$

# 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

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

where

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

# 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.