Predicate Logic

Lunjin Lu

Topics

- Syntax
- Semantics
- Clausal Form
- Resolution
- Substitutions

Formalization

- Predicate logic is a formalization of declarative sentences

1. Every mother loves her children
2. Mary is a mother and Tom is Mary's child
3. Mary loves Tom

(i) $\forall X(\forall Y(((mother(X) \wedge child\_of(Y,X)) \rightarrow loves(X,Y)))$

(ii) $mother(mary) \wedge child\_of(tom,mary)$

(iii) $loves(mary,tom)$

Alphabet ***A***

- Variables X ranging over ***V***
- Function symbols f ranging over Σ
- Predicate symbols p ranging over Π
- Logical connectives: $\neg\ \wedge\ \vee\ \rightarrow\ \leftrightarrow$
- Quantifiers: $\forall\ \exists$
- Auxiliary symbols (,)

Terms and Formulae

- Terms t ranging over ***T***

  $t ::= X \mid f(t_1,\ldots,t_n)$

- Formulae F ranging over ***Wff***

  $F ::= p(t_1,\ldots,t_n) \mid (\neg F_1) \mid (F_1 \wedge F_2)$

  $\mid (F_1 \vee F_2) \mid (F_1 \rightarrow F_2) \mid (F_1 \leftrightarrow F_2)$

  $\mid (\forall X\ F_1) \mid (\exists X\ F_1)$

- Closed formulae do not have ***free*** variables

Interpretation

- An interpretation J of an alphabet ***A*** is a non-empty domain D ($|J|$) and a mapping that associate
  - an n-ary function symbol f in ***A*** with a function f_J: $D^n \rightarrow D$
  - an n-ary predicate symbol p in ***A*** with a relation $p_J \subseteq D^n$

Semantics of Terms

- A valuation φ is a mapping from V to |J|.
- The meaning φ_J of t with respect to J and φ
  - $\varphi_J(X) = \varphi(X)$
  - $\varphi_J(f(t_1,\ldots,t_n)) = f_J(\varphi_J(t_1),\ldots, \varphi_J(t_n))$

Logical Consequences

- An interpretation J is a model of a set of closed formulae P iff every formula in P is true in J
- A formula F is a logical consequence of a set of formulae iff F is true in ***every*** model of P

Semantics of Formulae

- $J \models_\varphi p(t_1,\ldots,t_n)$ iff $< \varphi_J(t_1),\ldots, \varphi_J(t_n)>$ is in p_J
- $J \models_\varphi (\neg F)$ iff $J \not\models_\varphi F$
- $J \models_\varphi (F_1 \wedge F_2)$ iff $J \models_\varphi F_1$ and $J \models_\varphi F_2$
- $J \models_\varphi (F_1 \vee F_2)$ iff $J \models_\varphi F_1$ or $J \models_\varphi F_2$
- $J \models_\varphi (F_1 \rightarrow F_2)$ iff $J \models_\varphi F_2$ whenever $J \models_\varphi F_1$
- $J \models_\varphi (F_1 \leftrightarrow F_2)$ iff $J \models_\varphi (F_1 \rightarrow F_2)$ and $J \models_\varphi (F_2 \rightarrow F_1)$
- $J \models_\varphi (\forall X\ F_1)$ iff $J \models_{\varphi[X\rightarrow t]} F_1$ for **every** t in |J|
- $J \models_\varphi (\exists X\ F_1)$ iff $J \models_{\varphi[X\rightarrow t]} F_1$ for **some** t in |J|

- Write J |= F if F is a closed formula

Clausal forms

- A literal L is an atom or negation of an atom
- A clause is of the form $\forall\ (L_1 \vee L_2 \vee \ldots L_k)$ with all variable universally quantified
- Any first predicate logic formula **is logically equivalent to** a conjunction of clauses

Modus Pollens

- To prove a closed formula F is a logical consequence of a set P of closed formulae
- It is equivalent to proving that $\{\neg F\} \cup P$ is un-satisfiable.

Resolution

- If $\{\neg F\} \cup P$ is in clausal form then it can be done mechanically by repeated applying resolution to derive an empty clause (false).
- Let $\forall\, (L_1 \vee L_2 \vee \ldots L_n)$ and $\forall\, (K_1 \vee K_2 \vee \ldots K_m)$ be clauses that do not share variables, $L_i = p(\underline{t})$ and $K_j = \neg p(\underline{s})$ such that $\theta(\underline{t}) = \theta(\underline{s})$ then $\theta(\bigvee_{o \neq i} L_o \vee \bigvee_{o \neq j} K_o)$ is a logical consequence of $L_1 \vee L_2 \vee \ldots L_n$ and $K_1 \vee K_2 \vee \ldots K_m$

odd(s(0)).

odd(s(s(X))) ← odd(X).

Least Herbrand Model

Th. 2.12. If J' is a model of $P \cup \{G\}$ then $J = \{A \in B_P \mid J' \models A\}$ is a Herbrand model of $P \cup \{G\}$.

Th. 2.14. Let M be a family of Herbrand models of P. Then $\cap M$ is a Herbrand model of P.

M_P is the least Herbrand Model of P.

Th. 2.16. $M_P = \{A \in B_P \mid P \models A\}$

Def. $T_P(I) = \left\{ A_0 \;\middle|\; \begin{array}{l} A_0 \leftarrow A_1 \cdots A_m \in \mathrm{ground}(P) \text{ and} \\ \{A_1 \cdots A_m\} \subseteq I \end{array} \right\}$

Th 2.20. $M_P = T_P \uparrow \omega$ and M_P is the lfp. of T_P

Least Herbrand Model

Herbrand Universe ^{U_A}: All ground terms constructible from the alphabet A with at least one constant.

Herbrand base ^{B_A}: All ground atoms ....

U_P: } functors are those in P
B_P: } predicates

Herbrand interpretation: $\mathfrak{J}$ is a Herbrand interpretation if

- Domain of $\mathfrak{J}$ is U_P
- $f_J(x_1 \cdots x_n) = f(x_1 \cdots x_n)$ } pre-defined
- P_J is a subset of B_P^n

Herbrand model: A Herbrand interpretation that is a model

Definite Programs

Program:

child(tom, john)

child(ann, tom)

child(john, mark)

child(alice, john)

$\forall X \forall Y\, (\, grandchild(X,Y) \leftarrow \exists Z\, (child(X,Z) \wedge child(Z,Y))$

$\forall X \forall Y \forall Z\, (\, grandchild(X,Y) \leftarrow (child(X,Z) \wedge child(Z,Y))$

Queries: $\leftarrow child(X, john)$ $\qquad \exists X .\, child(X, john$

$\leftarrow grandchild(X, john)$

Definite Programs

* Definite clauses: clauses with at most one positive literal.

Rules: $\forall(A_0 \vee \neg A_1 \vee \neg A_2 \cdots \vee \neg A_m)$

$$\forall(A_1 \wedge A_2 \cdots \wedge A_m \rightarrow A_0)$$

$$\forall(A_0 \leftarrow A_1 \wedge A_2 \cdots \wedge A_m)$$

$$\underset{\uparrow \atop \text{Head}}{A_0} \leftarrow \underset{\uparrow \atop \text{Body}}{\underline{A_1 \wedge A_2 \cdots \wedge A_m}}$$

Facts: No body.

$$A_0 \leftarrow \blacksquare$$

Goals: No head

$$\square \leftarrow A_1 \wedge \cdots \wedge A_m$$

Empty clause: $\square \leftarrow \blacksquare$

Substitution

Application: $f(t_1 \cdots t_m)\theta \triangleq f(t_1\theta, \cdots t_m\theta)$

Thus, a substitution induces a mapping from terms to terms.

Composition: Let $\theta = \{x_1/s_1 \ldots x_m/s_m\}$

$\sigma = \{y_1/t_1 \cdots y_n/t_n\}$

$$\theta\sigma = \{x_1/s_1\sigma, \ldots x_m/s_m\sigma\} \cup \sigma \restriction (\{y_1 \cdots y_n\} \setminus \{x_1 \cdots x_m\})$$

Functional composition

Restriction:

$$\eta \restriction v = \{\, x_i/t_i \in \eta \mid x_i \in v \,\}$$

Idempotence: θ is idempotent iff $\theta = \theta\theta$

Substitution

Def. A substition θ is a finite set of pairs of terms $\{x_1/t_1, \dots x_n/t_n\}$ with x_i a variable, t_i a term, $x_i \neq t_i$ and $\forall i \neq j.\ x_i \neq x_j$.

Domain $dom(\theta) = \{x_1, \cdots x_n\}$

Range $Range(\theta) = \bigcup_{i=1}^{n} vars(t_i)$
where $vars(t_i)$ is the set of variables in t_i

A substitution θ is an (almost identity) function from Vars to Terms

Empty substitution is ε (identity)

Application $x\theta \triangleq \begin{cases} t & \text{if } x/t \in \theta \\ x & \text{otherwise} \end{cases}$

Unification

Unifier

- Let E be a set of equations $\{s_1 \cong t_1, s_2 \cong t_2, \ldots, s_n \cong t_n\}$ where s_i and t_i are terms
- A substitution θ is a ***unifier*** of E iff $s_k\theta \equiv t_k\theta$ for all k in [1..n]

Most General Unifier

- Let θ and σ be substitutions. θ is said to be more general than σ, denoted $\sigma \leq \theta$, if there is a substitution ω such that $\sigma = \theta\omega$.
- A substitution θ is a **most general unifier** (m.g.u.) of E iff θ is more general than any other unifier of E.
- Renaming substitution ρ is a permutation of V. Each renaming substitution ρ has an inverse ρ^{-1} such that $\rho\rho^{-1}=\varepsilon$
- Let o be a syntactic object (term, atom, equation, substitution and so on). Then $o\rho$ is called a variant of o.

Most General Unifiers

- **Fact:** If a set E of equations has a unifier then it has a most general unifier.
- **Fact:** The m.g.u.s of E are variant of each other. That is, E has a unique m.g.u. modulo renaming.
- **Fact:** All unifiers of E are instances of a m.g.u. of E.
- **Notation:** The m.g.u. of E is denoted mgu(E). The m.g.u. of $\{s \cong t\}$ is denoted mgu(s,t).

Solved form

- A set of equations $\{x_1 \cong t_1, x_2 \cong t_2, \ldots, x_n \cong t_n\}$ is said to be in ***solved form*** iff $x_1, x_2, \ldots, x_n$ are variables and none of $x_1, x_2, \ldots, x_n$ occur in any of $t_1, t_2, \ldots, t_n$
- ***Fact:*** Let $E = \{x_1 \cong t_1, x_2 \cong t_2, \ldots, x_n \cong t_n\}$ be in solved form then $\theta = \{x_1/t_1, x_2/t_2, \ldots, x_n/t_n\}$ is a m.g.u. of E.

Equivalent sets of equations

- Two sets of equations are ***equivalent*** iff they have the same set of unifiers.
- Solving a set of equations reduces to transform a set of equations into an equivalent set of equations in solved form.
- Algorithm in Figure 3.2 terminates. If E is unifiable that it returns a m.g.u. Otherwise, it returns ***failure***.

Unification Algorithm

Input: A set $\mathcal{E}$ of equations.
Output: An equivalent set of equations in solved form or **failure**.

repeat
    select an arbitrary $s \doteq t \in \mathcal{E}$;
    case $s \doteq t$ **of**
        $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)$ where $n \geq 0 \Rightarrow$
            replace equation by $s_1 \doteq t_1, \ldots, s_n \doteq t_n$;   *% case 1*
        $f(s_1, \ldots, s_m) \doteq g(t_1, \ldots, t_n)$ where $f/m \neq g/n \Rightarrow$
            halt with **failure**;   *% case 2*
        $X \doteq X \Rightarrow$
            remove the equation;   *% case 3*
        $t \doteq X$ where t is not a variable $\Rightarrow$
            replace equation by $X \doteq t$;   *% case 4*
        $X \doteq t$ where $X \neq t$ and X has more than one occurrence in $\mathcal{E} \Rightarrow$
            if X is a proper subterm of t **then**
                halt with **failure**   *% case 5a*
            else
                replace all other occurrences of X by t;   *% case 5b*
    esac
until no action is possible on any equation in $\mathcal{E}$;
halt with $\mathcal{E}$;

Figure 3.2: Solved form algorithm

Prolog

- Prolog is a programming language.

Prolog = Definite Programs + …