

# Lecture 12: Predicate Calculus

10:30 AM, Mar 5, 2009

## Contents

<b>1 Overview</b>	<b>1</b>
<b>2 Syntax</b>	<b>1</b>
<b>3 Semantics</b>	<b>2</b>
<b>4 Example</b>	<b>3</b>
<b>5 Logical Entailment</b>	<b>4</b>
5.1 Herbrand's Theorem . . . . .	4
5.2 Herbrand's Method . . . . .	5

## 1 Overview

In propositional logic, there is no concise method of encoding similar statements, such as: Amy loves her mother; Carolyn loves her mother; Michele loves her mother. These statements must be encoded as distinct propositions. Moreover, in propositional logic there is not even an awkward way of encoding statements of the form: everyone loves his/her mother; someone loves his/her mother. This lecture, which is concerned with **predicate calculus**<sup>1</sup>—first-order logic without quantifiers or variables—addresses the first of these concerns. The next lecture on **first-order logic** (with quantifiers and variables) addresses the second.

The ontological basis of propositional logic is the proposition. The ontological bases of predicate calculus are **objects** and their **attributes**. Some examples of objects are people, animals, plants, and baseball games. There are two types of attributes of particular relevance to predicate calculus: (i) **functions**, such as AMY, which is a **constant** (*i.e.*, a 0-place function), and MOTHER\_OF(AMY), which is a 1-place function; and (ii) **relations**, such as FEMALE(AMY), which is a **property** (*i.e.*, 1-place relation), and LOVES(MOTHER\_OF(AMY),AMY), which is a 2-place relation. Note that propositions are 0-place relations.

## 2 Syntax

Predicate calculus generalizes propositional logic with **terms**, which represent objects, and **predicates**, which represent propositions, properties, and relations. The predicate calculus alphabet includes a set of constant symbols  $\{a, b, c, \dots\}$  (*i.e.*, 0-ary function symbols), a set of  $n$ -ary function

---

<sup>1</sup>In CS 141, we refer to first-order logic with only ground terms (no variables or quantifiers) as predicate calculus. This is not necessarily standard nomenclature. First-order logic and predicate calculus are usually synonymous.

symbols  $\{f, g, h, \dots\}$  for  $n > 0$ , a set of  $n$ -ary predicate symbols  $\{P, Q, R, \dots\}$  for  $n \geq 0$ , two select 0-ary predicates  $\{\top, \perp\}$ , and a set of logical connectives  $\{\neg, \vee, \wedge, \rightarrow\}$ . Given this alphabet, the syntactic rules of the predicate calculus govern the construction of terms and formulas.

The following inductive definition describes the terms of predicate calculus:

- constant symbols  $a, b, c, \dots$  are terms
- if  $t_1, \dots, t_n$  are terms and  $f$  is an  $n$ -ary function symbol
  - $f(t_1, \dots, t_n)$  is a term

The following inductive definition describes the formulas of predicate calculus:

- if  $t_1, \dots, t_n$  are terms and  $P$  is an  $n$ -ary predicate symbol
  - $P(t_1, \dots, t_n)$  is a (n atomic) formula
- $\top$  and  $\perp$  are (atomic) formulas
- if  $\phi$  and  $\psi$  are formulas, then the following are (complex) formulas:
  - $\neg\phi$  is a formula
  - $\phi \vee \psi$  is a formula
  - $\phi \wedge \psi$  is a formula
  - $\phi \rightarrow \psi$  is a formula

As in the case of propositional logic, **complex** formulas involve one or more of the connectives; **atomic** formulas do not.

**Example:** Given the alphabet  $\mathcal{A} = \{0, 1, \text{SUCC}, \text{ADD}, <\}$ , where 0 and 1 are constant symbols, function symbol SUCC has arity 1, function symbol ADD has arity 2, and predicate symbol  $<$  has arity 2, the following are sample terms of predicate calculus:

$$0, \text{SUCC}(0), \text{SUCC}(\text{SUCC}(0)), \dots, \text{ADD}(0, 0), \text{ADD}(0, 1), \text{ADD}(1, 0), \text{ADD}(1, 1), \dots$$

and the following are sample formulas:

$$<(0, 0), <(0, 1), <(1, 0), <(1, 1), <(\text{SUCC}(0), 0), <(0, \text{SUCC}(0)), \dots$$

### 3 Semantics

The semantics of predicate calculus give meaning to its formulas. This meaning is derived from an interpretation of terms and predicates, the syntactic units used in the construction of formulas. Terms are interpreted as objects and predicates as relations. An **interpretation**  $\mathcal{I}$  is a tuple  $\langle D, M \rangle$  where the domain  $D$  is a (non-empty) set of objects and the map  $M$  is defined *s.t.*

- for all constant symbols  $a$ ,  $a^M$  is an object in the set  $D$

- for all  $n$ -ary function symbols  $f$ ,  $f^M : D^n \rightarrow D$   
(i.e.,  $f^M$  is an  $n$ -ary function on  $D$ )
- for all  $n$ -ary predicate symbols  $P$ ,  $P^M : D^n \rightarrow \{T, F\}$   
(i.e.,  $P^M$  is an  $n$ -ary relation on  $D$ )

**Example:** Continuing our example, given alphabet  $\mathcal{A}$  defined above, in the standard interpretation  $\mathcal{I} = \langle D, M \rangle$  with domain  $D = \mathbb{N}$ , the constant symbols 0 and 1 would map to themselves, the function symbol `SUCC` would map to the function  $n \mapsto n + 1$ , the function symbol `ADD` would map to the addition function, and the predicate symbol `<` would map to the less-than relation (i.e.,  $\{(0, 1), (0, 2), \dots, (1, 2), (1, 3), \dots, (2, 3), (2, 4), \dots\}$ ).

The terms of predicate calculus are interpreted under  $\mathcal{I}$  as follows:

- for all constant symbols  $a$ ,  $\mathcal{I}[a] = a^M$
- for all terms  $t_1, \dots, t_n$  and  $n$ -ary function symbols  $f$ 
  - $\mathcal{I}[f(t_1, \dots, t_n)] = f^M(\mathcal{I}[t_1], \dots, \mathcal{I}[t_n])$

As in propositional logic,  $\mathcal{I} \models \phi$  is defined inductively. Only in the base case does the definition for predicate calculus differ from that of propositional logic:

$$\mathcal{I} \models P(t_1, \dots, t_n) \text{ iff } (\mathcal{I}[t_1], \dots, \mathcal{I}[t_n]) \in P^M$$

The notions of satisfiability, validity, and unsatisfiability extend immediately to predicate calculus.

**Example:** Continuing our example yet again, given alphabet  $\mathcal{A}$  defined above, in the standard interpretation with domain  $\mathbb{N}$ , the term `SUCC(0)` is interpreted as  $\text{SUCC}^M(0^M)$ , which is 1. Similarly, the term `ADD(0, SUCC(0))` is interpreted as  $\text{ADD}^M(0^M, \text{SUCC}^M(0^M))$ , which is also 1.

## 4 Example

Given constant symbols  $\{\text{AMY}, \text{TISH}, \text{MERRIE}, \text{MICHAEL}\}$ ; function symbols  $\{\text{INSTRUCTOR\_OF}(\cdot)\}$ ; predicate symbols  $\{\text{OLDER}(\cdot, \cdot)\}$ . Is the formula  $\text{OLDER}(\text{INSTRUCTOR\_OF}(\text{MERRIE}), \text{MERRIE})$  satisfiable? What about the formula  $\text{OLDER}(\text{INSTRUCTOR\_OF}(\text{MICHAEL}), \text{MICHAEL})$ ?

Construct an interpretation  $\mathcal{I} = \langle D, M \rangle$  as follows:

- $D = \{\text{Amy}, \text{Tish}, \text{Merrie}, \text{Michael}\}$
- constant symbols
  - $\text{AMY}^M = \text{Amy}$
  - $\text{TISH}^M = \text{Tish}$
  - $\text{MERRIE}^M = \text{Merrie}$
  - $\text{MICHAEL}^M = \text{Michael}$
- function symbols

–  $\text{INSTRUCTOR\_OF}^M = \{(\text{Tish}, \text{Amy}), (\text{Merrie}, \text{Amy}), (\text{Michael}, \text{Amy})\}$

• predicate symbols

–  $\text{OLDER}^M = \{(\text{Michael}, \text{Amy}), (\text{Michael}, \text{Tish}), (\text{Michael}, \text{Merrie}),$   
 $(\text{Amy}, \text{Tish}), (\text{Amy}, \text{Merrie}), (\text{Tish}, \text{Merrie})\}$

$\mathcal{I}$  satisfies the formula  $\text{OLDER}(\text{INSTRUCTOR\_OF}(\text{MERRIE}), \text{MERRIE})$ .

$\mathcal{I} \models \text{OLDER}(\text{INSTRUCTOR\_OF}(\text{MERRIE}), \text{MERRIE})$   
iff  $(\mathcal{I}[\text{INSTRUCTOR\_OF}(\text{MERRIE})], \mathcal{I}[\text{MERRIE}]) \in \text{OLDER}^M$   
iff  $(\text{INSTRUCTOR\_OF}^M(\text{MERRIE}^M), \text{MERRIE}^M) \in \text{OLDER}^M$   
iff  $(\text{Amy}, \text{Merrie}) \in \text{OLDER}^M$

$\mathcal{I}$  does not satisfy  $\text{OLDER}(\text{INSTRUCTOR\_OF}(\text{MICHAEL}), \text{MICHAEL})$ , however.

$\mathcal{I} \not\models \text{OLDER}(\text{INSTRUCTOR\_OF}(\text{MICHAEL}), \text{MICHAEL})$   
iff  $(\mathcal{I}[\text{INSTRUCTOR\_OF}(\text{MICHAEL})], \mathcal{I}[\text{MICHAEL}]) \notin \text{OLDER}^M$   
iff  $(\text{INSTRUCTOR\_OF}^M(\text{MICHAEL}^M), \text{MICHAEL}^M) \notin \text{OLDER}^M$   
iff  $(\text{Amy}, \text{Michael}) \notin \text{OLDER}^M$

But  $\text{OLDER}(\text{INSTRUCTOR\_OF}(\text{MICHAEL}), \text{MICHAEL})$  is *not* unsatisfiable, even though it is not satisfied by  $\mathcal{I}$ .

**Exercise:** Modify the interpretation  $\mathcal{I}$  to produce an alternative interpretation  $\mathcal{I}'$  that satisfies  $\text{OLDER}(\text{INSTRUCTOR\_OF}(\text{MICHAEL}), \text{MICHAEL})$ .

## 5 Logical Entailment

Recall that the logical entailment problem is the following: given knowledge base KB and formula  $\phi$ , does the knowledge base KB semantically entail  $\phi$ : *i.e.*,  $\text{KB} \models \phi$ ? In other words, is it the case that all models of KB are also models of  $\phi$ ? In propositional logic, this question can be answered via the truth-table method. The Herbrand method is an analogous procedure that is applicable to knowledge bases of predicate calculus in normal form.

### 5.1 Herbrand's Theorem

Given an alphabet  $\mathcal{A}$ , the **Herbrand universe**  $A$  is the set of all ground terms, and the **Herbrand base**  $B$  is the set of all ground formulas. For example, if  $\mathcal{A} = \{a, b, P(\cdot)\}$ , then  $A = \{a, b\}$  and  $B = \{P(a), P(b)\}$ . Note that if an alphabet  $\mathcal{A}$  contains any function symbols, then both the Herbrand universe and the Herbrand base are infinite: *e.g.*, if  $\mathcal{A} = \{a, f, P\}$ , then  $A = \{a, f(a), f(f(a)), \dots\}$  and  $B = \{P(a), P(f(a)), P(f(f(a))), \dots\}$ .

A **Herbrand interpretation**, or **Herbrand model**,  $\mathcal{H} = \langle D, M \rangle$ , is one in which the domain is the Herbrand universe (*i.e.*,  $D = A$ ) and  $M$  maps terms into themselves (*i.e.*,  $c^M = c$ , for all

constant symbols  $c$ , and  $f^M = \{(a, f(a)) \mid a \in A\}$ , for all function symbols  $f$ ). For example, if  $\mathcal{A} = \{a, f, g\}$ , so that  $A = \{a, f(a), g(a), f(f(a)), f(g(a)), g(f(a)), g(g(a)), \dots\}$ , then

$$f^M = \{(a, f(a)), (f(a), f(f(a))), (g(a), f(g(a))), (f(f(a)), f(f(f(a))))\dots\}$$

and

$$g^M = \{(a, g(a)), (f(a), g(f(a))), (g(a), g(g(a))), (f(f(a)), g(f(f(a))))\dots\}$$

Herbrand interpretations do not restrict the interpretation of predicates. Thus, each Herbrand interpretation is identified with some subset of the Herbrand base. If  $\mathcal{A} = \{a, b, P(\cdot)\}$ , then in every Herbrand interpretation  $D = \{a, b\}$ ,  $a^M = a$  and  $b^M = b$ , and  $P^M \in \{\{\}, \{a\}, \{b\}, \{a, b\}\}$ . In other words, the four possible interpretations are: the empty set,  $\{P(a)\}$ ,  $\{P(b)\}$ , and  $\{P(a), P(b)\}$ .

**Theorem:** A predicate calculus formula has a model iff it has a Herbrand model.

**Proof:** Let  $\phi$  denote a formula of predicate calculus. Clearly, if  $\phi$  has a Herbrand model, then  $\phi$  has a model. Thus, it suffices to show that if  $\phi$  has a model, then  $\phi$  has a Herbrand model. Suppose  $\mathcal{I} \models \phi$ . Given an (Herbrand) interpretation  $\mathcal{H} = \langle D', M' \rangle$ , with  $D'$  equal to the Herbrand base and  $M'$  mapping terms into themselves, extend the mapping  $M'$  as follows: for all  $n$ -ary predicates  $P$ ,

$$P^{M'} = \{(\mathcal{H}[t_1], \dots, \mathcal{I}[t_n]) \mid \mathcal{I} \models P(t_1, \dots, t_n)\}$$

If  $\phi$  is an atomic formula, then by construction,  $\mathcal{I} \models \phi$  iff  $\mathcal{H} \models \phi$ . Otherwise (*i.e.*, if  $\phi$  is a complex formula), the proof follows by induction on the structure of  $\phi$ .  $\diamond$

**Example:** Continuing with the above example, assume  $\mathcal{A} = \{a, b, P(\cdot)\}$ .

An interpretation  $\mathcal{I}$  with  $D = \{1, 2\}$  with  $a^M = 1$ ,  $b^M = 2$ , and  $P^M = \{1\}$ , corresponds to Herbrand interpretation  $\mathcal{H} = \langle D', M' \rangle$  with  $D' = D$  and  $a^{M'} = a$ ,  $b^{M'} = b$ , and  $P^{M'} = \{a\}$ .

But an interpretation  $\mathcal{I}$  with  $D = \{1, 2\}$  with  $a^M = 1$ ,  $b^M = 1$ , and  $P^M = \{1\}$ , corresponds to Herbrand interpretation  $\mathcal{H} = \langle D', M' \rangle$  with  $D' = D$  and  $a^{M'} = a$ ,  $b^{M'} = b$ , and  $P^{M'} = \{a, b\}$ .

**Corollary:** [Herbrand's Theorem] A set of predicate calculus formulas has a model iff it has a Herbrand model.

## 5.2 Herbrand's Method

Assuming a finite number of constant and predicate symbols, it is theoretically possible to solve the logical entailment problem for predicate calculus via brute force by enumerating all Herbrand interpretations. For example, we can determine whether or not  $P(a) \vee Q(b)$  and  $\neg P(a)$  entail  $Q(b)$  by enumerating all Herbrand interpretations in which  $P(a) \vee Q(b)$  and  $\neg P(a)$  hold, and checking whether or not  $Q(b)$  also holds in these interpretations. The Herbrand interpretations in which  $P(a) \vee Q(b)$  and  $\neg P(a)$  hold are listed in Table 1. Indeed, in all such interpretations in which both these formulas hold, (namely  $P^M = \emptyset, Q^M = \{b\}$ ;  $P^M = \emptyset, Q^M = \{a, b\}$ ;  $P^M = \{b\}, Q^M = \{b\}$ ;  $P^M = \{b\}, Q^M = \{a, b\}$ ) so too does  $Q(b)$ . This procedure is called Herbrand's method.

However, Herbrand's method can be exceedingly expensive. Assuming  $n$  constants (*i.e.*, objects in the domain), and one predicate of arity  $k$ , there are  $2^{n^k}$  Herbrand interpretations. Worse, if the number of constant symbols is infinite (*e.g.*,  $\mathbb{N}$ ), or if the alphabet contains any function symbols, then this method is not effective, because the Herbrand universe is infinite. Consequently, we resort to proof theory to solve the logical entailment problem for predicate calculus and first-order logic.

$P^M$	$Q^M$	$P(a) \vee Q(b)$		$\neg P(a)$	
$\emptyset$	$\emptyset$	-	-	$\emptyset$	$\emptyset$
$\emptyset$	$\{a\}$	-	-	$\emptyset$	$\{a\}$
$\emptyset$	$\{b\}$	$\emptyset$	$\{b\}$	$\emptyset$	$\{b\}$
$\emptyset$	$\{a, b\}$	$\emptyset$	$\{a, b\}$	$\emptyset$	$\{a, b\}$
$\{a\}$	$\emptyset$	$\{a\}$	$\emptyset$	-	-
$\{a\}$	$\{a\}$	$\{a\}$	$\{a\}$	-	-
$\{a\}$	$\{b\}$	$\{a\}$	$\{b\}$	-	-
$\{a\}$	$\{a, b\}$	$\{a\}$	$\{a, b\}$	-	-
$\{b\}$	$\emptyset$	-	-	$\{b\}$	$\emptyset$
$\{b\}$	$\{a\}$	-	-	$\{b\}$	$\{a\}$
$\{b\}$	$\{b\}$	$\{b\}$	$\{b\}$	$\{b\}$	$\{b\}$
$\{b\}$	$\{a, b\}$	$\{b\}$	$\{a, b\}$	$\{b\}$	$\{a, b\}$
$\{a, b\}$	$\emptyset$	$\{a, b\}$	$\emptyset$	-	-
$\{a, b\}$	$\{a\}$	$\{a, b\}$	$\{a\}$	-	-
$\{a, b\}$	$\{b\}$	$\{a, b\}$	$\{b\}$	-	-
$\{a, b\}$	$\{a, b\}$	$\{a, b\}$	$\{a, b\}$	-	-

Table 1: Herbrand's Method: An Example