

First Order Logic – Semantics (3A)

Copyright (c) 2016 - 2017 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice

Based on

Contemporary Artificial Intelligence,
R.E. Neapolitan & X. Jiang

Logic and Its Applications,
Burkey & Foxley

Examples of Terms

no expression involving a **predicate symbol** is a term.

x y $f(x)$ $g(x, y)$

$father(x)$ A function returns neither True nor False

The father of x

$Father(x)$ A predicate returns always True or False

Is x a father?

$\forall x \text{ love}(x, y)$: free variable y
 $\forall x \text{ tall}(x)$: no free variable

Bound variable x

Free variable y

https://en.wikipedia.org/wiki/First-order_logic#Formation_rules

Terms

Terms

1. **Variables**. Any variable is a term.
2. **Functions**. Any expression $f(t_1, \dots, t_n)$ of n arguments is a term where each argument t_i is a term and f is a function symbol of valence n . In particular, symbols denoting **individual constants** are **0-ary function symbols**, and are thus terms.

Only expressions which can be obtained by finitely many applications of rules 1 and 2 are terms.

no expression involving a **predicate symbol** is a term.

https://en.wikipedia.org/wiki/First-order_logic#Formation_rules

Formulas

Formulas (wffs)

Predicate symbols.

Equality.

Negation.

Binary connectives.

Quantifiers.

$P(x)$ $Q(x, y)$

$x = f(y)$

$\neg Q(x, y)$

$P(x) \wedge \neg Q(x, y)$

$\forall x, y (P(x) \wedge \neg Q(x, y))$

Only expressions which can be obtained by finitely many applications of rules 1–5 are formulas.

The formulas obtained from the first two rules are said to be **atomic formulas**.

https://en.wikipedia.org/wiki/First-order_logic#Formation_rules

Formulas

Formulas (wffs)

Predicate symbols. If P is an n -ary predicate symbol and t_1, \dots, t_n are terms then $P(t_1, \dots, t_n)$ is a formula.

Equality. If the equality symbol is considered part of logic, and t_1 and t_2 are terms, then $t_1 = t_2$ is a formula.

Negation. If φ is a formula, then $\neg\varphi$ is a formula.

Binary connectives. If φ and ψ are formulas, then $(\varphi \rightarrow \psi)$ is a formula. Similar rules apply to other binary logical connectives.

Quantifiers. If φ is a formula and x is a variable, then $\forall x \varphi$ (for all x , holds) and $\exists x \varphi$ (there exists x such that φ) are formulas.

$$P(x) \quad Q(x, y)$$

$$x = f(y)$$

$$\neg Q(x, y)$$

$$P(x) \wedge \neg Q(x, y)$$

$$\forall x, y (P(x) \wedge \neg Q(x, y))$$

https://en.wikipedia.org/wiki/First-order_logic#Formation_rules

Atoms and Compound Formulas

a formula that contains **no logical connectives**

a formula that has **no strict subformulas**

Atoms :

the **simplest** well-formed formulas of the logic.

$$P(x) \quad Q(x, y)$$

Compound formulas :

formed by combining the atomic formulas using the **logical connectives**.

$$P(x) \wedge \neg Q(x, y)$$

$$\forall x, y (P(x) \wedge \neg Q(x, y))$$

https://en.wikipedia.org/wiki/Atomic_formula

Atomic Formula

for **propositional logic**

the atomic formulas are the **propositional variables**

p

q

for **predicate logic**

the atoms are **predicate symbols** together with their **arguments**,
each argument being a **term**.

$P(x)$

$Q(x, f(y))$

In **model theory**

atomic formula are merely strings of symbols with a given signature
which may or may not be satisfiable with respect to a given model

https://en.wikipedia.org/wiki/Atomic_formula

A Signature and a Language

First specify a **signature**

Constant Symbols $\{c_1, c_2, \dots, c_n\} = D$

Predicate Symbols $\{P_1, P_2, \dots, P_m\}$

Function Symbols $\{f_1, f_2, \dots, f_l\}$

Determines the **language**

Given a language

A **model** is specified

A **domain of discourse**

a set of entities

$\{\text{entity}_1, \text{entity}_2, \dots, \text{entity}_n\}$

An **interpretation**

constant assignments

$\{c_1, c_2, \dots, c_n\} = D$

function assignments

$f_1(), f_2(), \dots, f_l()$

truth value assignments

$P_1(), P_2(), \dots, P_m()$

Model – domain of discourse

1. a nonempty set D of **entities** called a **domain of discourse**
 - this domain is a set
 - each element in the set : entity
 - each constant symbol : one entity in the domain

If we considering all individuals in a class,
The constant symbols might be

'Mary', - an entity
'Fred', - an entity
'John', - an entity
'Tom' - an entity

Model – interpretation

2. an **interpretation**

(a) an entity in D is assigned to each of the constant symbols.

Normally, every entity is assigned to a constant symbol.

(b) for each **function**,

an entity is assigned to each possible input of entities to the **function**

(c) the predicate '**True**' is always assigned **the value T**

The predicate '**False**' is always assigned **the value F**

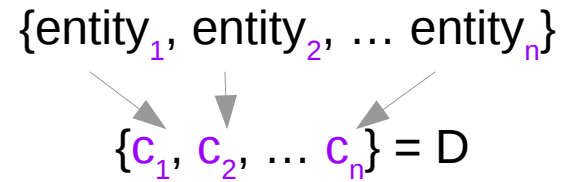
(d) for every other **predicate**,

the value T or F is assigned

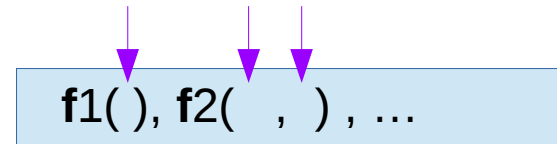
to each possible input of entities to the **predicate**

Interpretation

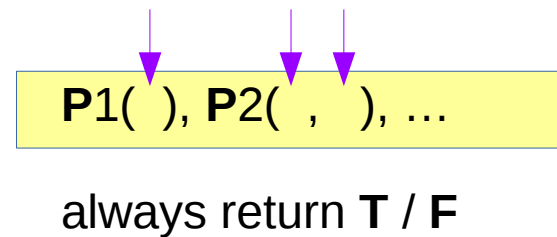
Constant assignments



Function assignments



Truth value assignments



Interpretation

Propositional Logic

	A	B
Interpretation I_1 →	T	T
Interpretation I_2 →	T	F
Interpretation I_3 →	F	T
Interpretation I_4 →	F	F

First Order Logic

	P1()	P2()	...	Sentences	
	P1()	P2()	...	S1	S2
Interpretation I_1 →	T	T			
Interpretation I_2 →	T	F			
Interpretation I_3 →	F	T			
Interpretation I_4 →	F	F			

$\{\text{entity}_1, \text{entity}_2, \dots, \text{entity}_n\}$

$\{c_1, c_2, \dots, c_n\} = D$

$f_1(), f_2(,), \dots$

$P_1(), P_2(,), \dots$

always return **T / F**

Each possible input of entities

Arity one: $C(n, 1)$
Arity two: $C(n, 2)$
Arity three: $C(n, 3)$

...

Arity one functions & predicates: $C(n, 1)$
Arity two: $C(n, 2)$
Arity three: $C(n, 3)$

...

$\{\text{entity}_1, \text{entity}_2, \dots, \text{entity}_n\}$

$\{c_1, c_2, \dots, c_n\} = D$

$f1(), f2(,), \dots$

$P1(), P2(,), \dots$

always return **T / F**

Interpretation

Constant assignments

(a) an entity → the constant symbols.

Function assignments

(b) an entity → each possible input of entities to the **function**

Truth value assignments

(c) the value **T** → the predicate '**True**'
the value **F** → the predicate '**False**'

(d) for every other **predicate**,
the value **T** or **F** is assigned → every other predicate
to each possible input of entities to the **predicate**

Signature Model Examples A – (1)

Signature

1. constant symbols = { Mary, Fred, Sam }
2. predicate symbols = { married, young }
 - married(x, y) : arity two
 - young(x) : arity one

Model

1. domain of discourse D : the set of three particular *individuals*

- this domain is a set
- each element in the set : entity (= *individuals*)
- each constant symbol : one entity in the domain (= one *individual*)

2. interpretation

(a) a different *individual* is assigned to each of the **constant symbols**

(a) an entity in D is assigned to each of the constant symbols.
Normally, every entity is assigned to a constant symbol.

Signature Model Examples A – (2)

(b) for each **function**,
an entity is assigned to each possible input of entities to the **function**

(c) the predicate '**True**' is always assigned the value T
The predicate '**False**' is always assigned the value F

(d) the truth value assignments for every predicate

$\text{young}(\text{Mary}) = \text{F}$, $\text{young}(\text{Fred}) = \text{F}$, $\text{young}(\text{Sam}) = \text{T}$

$\text{married}(\text{Mary}, \text{Mary}) = \text{F}$, $\text{married}(\text{Mary}, \text{Fred}) = \text{T}$, $\text{married}(\text{Mary}, \text{Sam}) = \text{F}$
 $\text{married}(\text{Fred}, \text{Mary}) = \text{T}$, $\text{married}(\text{Fred}, \text{Fred}) = \text{F}$, $\text{married}(\text{Fred}, \text{Sam}) = \text{F}$
 $\text{married}(\text{Sam}, \text{Mary}) = \text{F}$, $\text{married}(\text{Sam}, \text{Fred}) = \text{F}$, $\text{married}(\text{Sam}, \text{Sam}) = \text{F}$

(d) for every other **predicate**,
the value T or F is assigned
to each possible input of entities to the **predicate**

(Mary, Mary), (Mary, Fred), (Mary, Sam)
(Fred, Mary), (Fred, Fred), (Fred, Sam)
(Sam, Mary), (Sam, Fred), (Sam, Sam)

Signature Model Examples B – (1)

Signature

1. constant symbols = { Fred, Mary, Sam }
2. predicate symbols = { love } love(x, y) : arity two
3. function symbols = { mother } mother(x) : arity one

Model

1. domain of discourse D : the set of three particular individuals
2. interpretation
 - (a) a different individual is assigned to each of the **constant symbols**
 - (b) **the truth value assignments for every predicate**
love(Fred, Fred) = F, love(Fred, Mary) = F, love(Fred, Ann) = F
love(Mary, Fred) = T, love(Mary, Mary) = F, love(Mary, Ann) = T
love(Ann, Fred) = T, love(Ann, Mary) = T, love(Ann, Ann) = F
 - (c) **the function assignments**
mother(Fred) = Mary, mother(Mary) = Ann, mother(Ann) = - (no assignment)

Signature Model Examples B – (2)

2. interpretation

(a) a different individual is assigned to each of the **constant symbols**

(a) an entity in D is assigned to each of the constant symbols.
Normally, every entity is assigned to a constant symbol.

(b) **the truth value assignments**

(b) for each **function**,
an entity is assigned to each possible input of entities to the **function**

love(Fred, Fred) = F, **love**(Fred, Mary) = F, **love**(Fred, Ann) = F
love(Mary, Fred) = T, **love**(Mary, Mary) = F, **love**(Mary, Ann) = T
love(Ann, Fred) = T, **love**(Ann, Mary) = T, **love**(Ann, Ann) = F

(c) **the function assignments**

(d) for every other **predicate**,
the value T or F is assigned
to each possible input of entities to the **predicate**

mother(Fred) = Mary, **mother**(Mary) = Ann, **mother**(Ann) = - (no assignment)

The truth value of sentences

The truth values of **all sentences** are assigned :

1. the truth values for **sentences** developed with the symbols \neg , \wedge , \vee , \Rightarrow , \Leftrightarrow are assigned as in propositional logic.
2. the truth values for two terms connected by the $=$ symbol is **T** if both terms refer to the same entity; otherwise it is **F**
3. the truth values for $\forall x p(x)$ has value **T** if $p(x)$ has value **T** for **every assignment** to x of an **entity** in the domain D ; otherwise it has value **F**
4. the truth values for $\exists x p(x)$ has value **T** if $p(x)$ has value **T** for **at least one assignment** to x of an **entity** in the domain D ; otherwise it has value **F**
5. the operator **precedence** is as follows \neg , $=$, \wedge , \vee , \Rightarrow , \Leftrightarrow
6. the **quantifiers** have precedence over the operators
7. **parentheses** change the order of the precedence

Formulas and Sentences

An **formula**

- A **atomic formula**
- The operator \neg followed by a **formula**
- Two formulas separated by \wedge , \vee , \Rightarrow , \Leftrightarrow
- A **quantifier** following by a variable followed by a formula

A **sentence**

- A **formula** with **no free variables**

$\forall x \text{ love}(x,y)$: free variable y	: not a sentence
$\forall x \text{ tall}(x)$: no free variable	: a sentence

Finding the truth value

Find the truth values of **all sentences**

1. \neg , \wedge , \vee , \Rightarrow , \Leftrightarrow

2. = symbol

3. $\forall x p(x)$

4. $\exists x p(x)$

5. the **operator precedence** is as follows \neg , =, \wedge , \vee , \Rightarrow , \Leftrightarrow

6. the **quantifiers** (\forall , \exists) have precedence over the **operators**

7. **parentheses** change the order of the precedence

Truth values of sentences

Propositional Logic

	A	B	
Interpretation I_1 →	T	T	
Interpretation I_2 →	T	F	
Interpretation I_3 →	F	T	
Interpretation I_4 →	F	F	

terms x y $f(x)$ $g(x, y)$

atomic formulas $P(x)$ $Q(x, y)$

First Order Logic

Sentences

	P1()	P2()	...	S1	S2
Interpretation I_1 →	T	T			
Interpretation I_2 →	T	F			
Interpretation I_3 →	F	T			
Interpretation I_4 →	F	F			

formulas / sentences $\forall x, y (P(x) \wedge \neg Q(x, y))$

Sentence Examples (1)

Signature

Constant Symbols = {Socrates, Plato, Zeus, Fido}

Predicate Symbols = {human, mortal, legs} all arity one

Model

D: the set of these four particular individuals

Interpretation

(a) a different individual is assigned to each of the constant symbols

(b) the truth value assignment

human(Socrates)=T, human(Plato)=T, human(Zeus)=F, human(Fido)=F

mortal(Socrates)=T, mortal(Plato)=T, mortal(Zeus)=F, mortal(Fido)=T

legs(Socrates)=T, legs(Plato)=T, legs(Zeus)=T, legs(Fido)=T

Sentence Examples (2)

Sentence 1: $\text{human}(\text{Zeus}) \wedge \text{human}(\text{Fido}) \vee \text{human}(\text{Socrates}) = \text{T}$
F \wedge F \vee T

Sentence 2: $\text{human}(\text{Zeus}) \wedge (\text{human}(\text{Fido}) \vee \text{human}(\text{Socrates})) = \text{F}$
F \wedge (F \vee T)

Sentence 3: $\forall x \text{human}(x) = \text{F}$
 $\text{human}(\text{Zeus})=\text{F}, \text{human}(\text{Fido})=\text{F}$

Sentence 4: $\forall x \text{mortal}(x) = \text{F}$
 $\text{mortal}(\text{Zeus})=\text{F}$

Sentence 5: $\forall x \text{legs}(x) = \text{T}$
 $\text{legs}(\text{Socrates})=\text{T}, \text{legs}(\text{Plato})=\text{T}, \text{legs}(\text{Zeus})=\text{T}, \text{legs}(\text{Fido})=\text{T}$

Sentence 6: $\exists x \text{human}(x) = \text{T}$
 $\text{human}(\text{Socrates})=\text{T}, \text{human}(\text{Plato})=\text{T}$

Sentence 7: $\forall x (\text{human}(x) \Rightarrow \text{mortal}(x)) = \text{T}$

Sentence Examples (3)

Sentence 7: $\forall x (\text{human}(x) \Rightarrow \text{mortal}(x)) = T$

$\text{human}(\text{Socrates})=T$, $\text{mortal}(\text{Socrates})=T$,
 $\text{human}(\text{Plato})=T$, $\text{mortal}(\text{Plato})=T$,
 $\text{human}(\text{Zeus})=F$, $\text{mortal}(\text{Zeus})=F$,
 $\text{human}(\text{Fido})=F$ $\text{mortal}(\text{Fido})=T$

$T \Rightarrow T : T$
 $T \Rightarrow T : T$
 $F \Rightarrow F : T$
 $F \Rightarrow T : T$

References

- [1] en.wikipedia.org
- [2] en.wiktionary.org
- [3] U. Endriss, “Lecture Notes : Introduction to Prolog Programming”
- [4] <http://www.learnprolognow.org/> Learn Prolog Now!
- [5] http://www.csupomona.edu/~jrfisher/www/prolog_tutorial
- [6] www.cse.unsw.edu.au/~billw/cs9414/notes/prolog/intro.html
- [7] www.cse.unsw.edu.au/~billw/dictionaries/prolog/negation.html
- [8] <http://ilppp.cs.lth.se/>, P. Nugues, `An Intro to Lang Processing with Perl and Prolog