

First-Order Logic

Syntax and Semantics

David Walker

Princeton University

Following notes by Zak Kincaid

Limits of Propositional Logic

Atomic propositions are *indivisible*. Examples:

```
p = "every package has at most one installed version"
q1 = "gcc has at most one installed version"
q2 = "binutils has at most one installed version"
q3 = "cpp has ... "
...
qn = ...
```

p should imply q_i but the propositions themselves are unrelated to one another. We need a formula containing n copies of $p \rightarrow q_i$ to model this notion.

First-order logic is much more powerful, adding:

Relations Between Terms Instead of Indivisible Propositions:

$x = y$, $x < y$, $x \in y$, $\text{oneVersion}(p)$, $\text{installed}(p, v)$

Functions to Compute Complex Terms (instead of no terms at all):

$f(x)$, $c \cdot (x + y)$, $\text{select}(a, i)$

Quantifiers:

At most one version installed: $\forall p. \forall v1. \forall v2. \text{installed}(p, v1) \wedge \text{installed}(p, v2) \rightarrow v1 = v2$

A Language for Mathematics

First-order logic is a general-purpose language for doing mathematics.
Including the mathematics of programs.

Algebra

$\forall x. \forall y. \forall z.$

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

term

term

relation between
terms

function
symbol

the operator $\cdot$ is associative

Number theory

$\forall n. n > 2 \rightarrow$

$\neg \exists x, y, z.$

$$x > 0 \wedge y > 0 \wedge z > 0 \wedge$$

$$x^n + y^n = z^n$$

constants are
functions that
take no arguments

Fermat's last theorem (over $\mathbb{N}$)

Set theory

$\forall x. \exists y. \forall z.$

$$(\forall w. w \in z \rightarrow w \in x) \leftrightarrow z \in y$$

z is subset
of x

powerset of x

every set has a powerset (ZF)

Program verification

$\forall i. \forall j.$

$$(0 \leq i \wedge i < j \wedge j < n) \rightarrow$$

$$\text{select}(a, i) \leq \text{select}(a, j)$$

array a is sorted on $[0, n)$

First-order Logic Syntax

Signatures

A **Propositional Signature** was a set of atomic propositions $P = \{p, q, \dots\}$

A **First-order Signature** tells us which **functions** and **relations** appear in the logic

Definition (First-order signature)

A signature is a triple $\Sigma = \langle F, R, ar \rangle$ where

- **F** is a set of **function symbols**
- **R** is a set of **relation symbols** (also called predicates)
- **ar** : $F \cup R \rightarrow \mathbb{N}$ gives the **arity** (number of arguments) of each symbol

Examples:

- $\in$ is a relation symbol of arity 2: $w \in z$
 - we often write common binary relations (and binary functions) infix
- **incr** (the +1 operation) is a function symbol of arity 1: **incr**(incr(n))
- **3** is a function symbol of arity 0 (ie: constants are function symbols with arity 0)
 - 0, 1, 2, 4.5, -7, “hello” could all be constants
- an atomic proposition is a relation with arity 0

Signature Examples

Notation “symbol/int” tells us the arity of a symbol:

$+ / 2$ means “+ has arity 2”

Signature Notation:

$(f/2, c/0 \ ; \ P/1, Q/2)$

means $F = \{f, c\},$

$R = \{P, Q\},$

$\text{ar}(f) = 2, \text{ar}(c) = 0, \text{ar}(P) = 1, \text{ar}(Q) = 2$

} Σ

Signature for set theory:

- $(\ ; \ \in / 2)$
- No function symbols. **One** relation (set membership)
- All other concepts (empty set, subset, union, etc) may be defined via axioms

Terms and Formulas

Given a signature $\Sigma = \langle F, R, ar \rangle$, FOL is defined by **two** syntactic categories:

Formula (Σ) :

```
A, B ::= r(t1, ..., tn)           where n = ar(r)
        | t1 = t2
        | true | false
        | ¬A | A ∧ B | A ∨ B | A → B
        | ∀x. A | ∃x. A
```

Term (Σ) :

```
t ::= x | f(t1, ..., tn)           where n = ar(f)
```

Atomic formula:

- $r(t_1, \dots, t_n)$
- $s = t$
- true, false.

The body of $\forall x.$ and $\exists x.$ extends as far right as possible:

$\forall x. p(x) \rightarrow q(x)$ means $\forall x. (p(x) \rightarrow q(x))$

Implementing Syntax in OCaml

Two grammars, two datatypes:

```
type var = string
type fsym = string (* function symbol *)
type rsym = string (* relation symbol *)

type term =
  | Var of var
  | App of fsym * term list

type formula =
  | Rel of rsym * term list
  | Eq of term * term
  | True
  | False
  | Not of formula
  | And of formula * formula
  | Or of formula * formula
  | Implies of formula * formula
  | All of var * formula
  | Exists of var * formula
```

`App ("f", ts)` is $f(t_1, \dots, t_n)$

A constant is `App ("c", [])`.

Quantifiers carry the **name** of the variable they bind.

To ensure that formula are “well-typed,” we need an arity-checker.

eg: `Rel (“<”, [t])` is ill-typed formula

eg: `App(“+”, [])` is ill-typed term

`type ar : string -> int`

`check_term : ar -> term -> bool`

`check_form : ar -> formula -> bool`

Free and Bound Variables

An occurrence of a variable inside the scope of a matching quantifier is **bound**; otherwise it is **free**.

$$(\forall x. p(x, y)) \rightarrow q(f(y), x)$$

Most of the time, we want our formula to be *closed*

- a *closed formula* contains no free variables

A *sentence* is another word for a closed formula.

First-Order Logic Semantics

Structures

A **propositional structure** gave meaning to propositions:

- a function $M : P \rightarrow \{\text{true}, \text{false}\}$.

A **first-order structure** provides a **domain** (the objects formula talk about) and tells us what every function and relation means (by mapping symbols to real functions or relations)

Definition (Σ -structure)

A Σ -structure M consists of $(U, \text{FUN}, \text{REL})$

- a **non-empty** set **U** , the **universe** (or domain)
- **FUN**, which maps each function symbol f to a function on U s:
 - **$\text{FUN}(f) : U \times \dots \times U \rightarrow U$** ($\text{ar}(f)$ arguments)
- **REL**, which maps each relation symbol r to a relation on U s:
 - **$\text{REL}(r) \subseteq U \times \dots \times U$** ($\text{ar}(r)$ components)

A constant c (arity 0 function) is an element of U . ie: $\text{FUN}(c) : U$

Equality is not in the signature: “=” always means *identity*.

Given a structure $M = (U, \text{FUN}, \text{REL})$, I will sometimes write:

U^M , f^M and r^M instead of U , $\text{FUN}(f)$, and $\text{REL}(r)$

Structures

Definition (Σ -structure)

A Σ -structure M consists of $(U, \text{FUN}, \text{REL})$

- a **non-empty** set **U** , the **universe** (or domain)
- FUN , which maps each function symbol f to a function on U :
 - **$\text{FUN}(f) : U \times \dots \times U \rightarrow U$** ($\text{ar}(f)$ arguments)
- REL , which maps each relation symbol r to a relation on U :
 - **$\text{REL}(r) \subseteq U \times \dots \times U$** ($\text{ar}(r)$ components)

Notes:

- A constant c (arity 0 function) is an element of U . ie: $\text{FUN}(c) : U$
- **Equality is not in the signature:**
 - “=” always means *identity*
- Given a structure $M = (U, \text{FUN}, \text{REL})$, I will sometimes write:

U^M , f^M and r^M instead of U , $\text{FUN}(f)$, and $\text{REL}(r)$

Example: Graphs

What kinds of symbols do we need to reason about directed graphs?

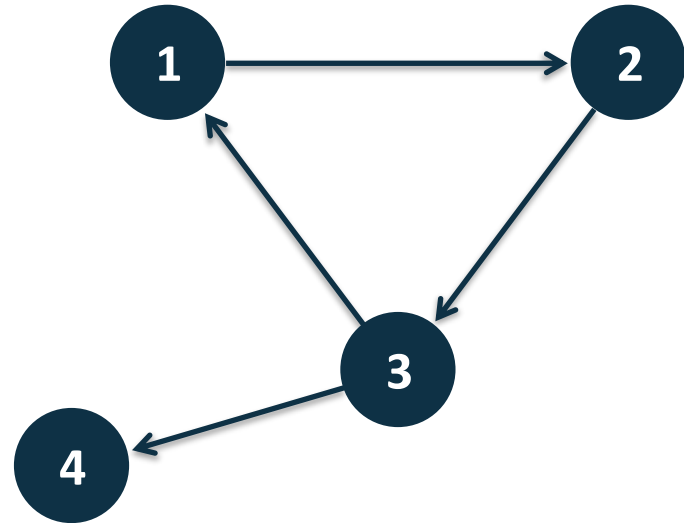
Signature:

Function symbols:

- 1, 2, 3, 4 – arity 0 constants

Relation symbol:

- $\text{link}(x,y)$ – true if there is an edge from x to y (link has arity 2)



Some facts about our graph:

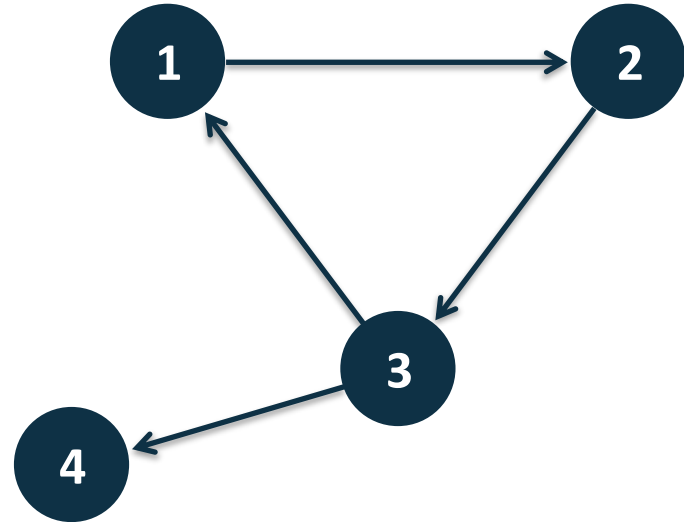
$\text{link}(1,2)$

$\text{link}(2,3)$

$\neg (\forall x. \exists y. \text{link}(x, y))$

Example: Graphs

What kinds of symbols do we need to reason about directed graphs?



Signature:

Function symbols:

- 1, 2, 3, 4 – arity 0 constants

Relation symbol:

- $\text{link}(x,y)$ – true if there is an edge from x to y (link has arity 2)

A first-order model M over our signature:

$$U = \{ 1, 2, 3, 4 \}$$

$$\text{FUN}(i) = i$$

$$\text{REL}(\text{link}) = \{ (1,2), (2,3), (3,1), (3,4) \}$$

Facts:

$$\text{link}(1,2)$$

$$\text{link}(2,3)$$

$$\neg (\forall x. \exists y. \text{link}(x, y))$$

Example: Graphs

What kinds of symbols do we need to reason about directed graphs?

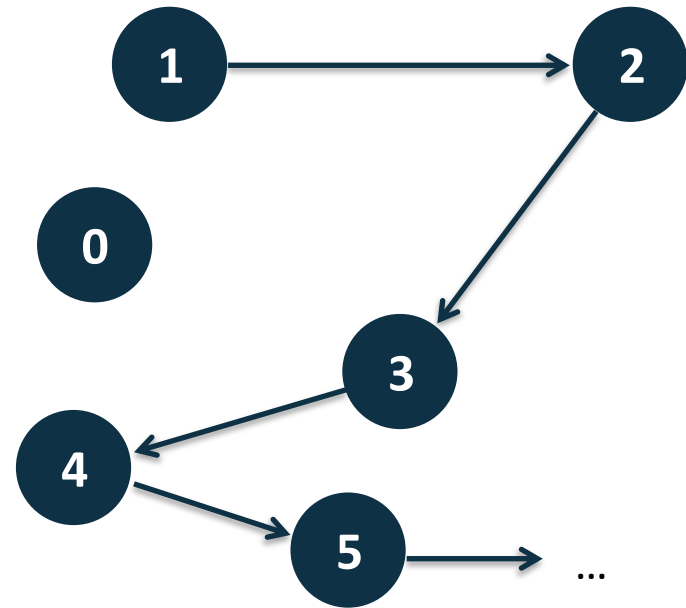
Signature:

Function symbols:

- 1, 2, 3, 4 – arity 0 constants

Relation symbol:

- $\text{link}(x,y)$ – true if there is an edge from x to y (link has arity 2)



A first-order model M over our signature:

$U = \text{Nat}$

$\text{FUN}(i) = i$

$\text{REL}(\text{link}) = \{ (1,2), (2,3), (3,4), (4,5), (5,6), \dots \}$

Facts:

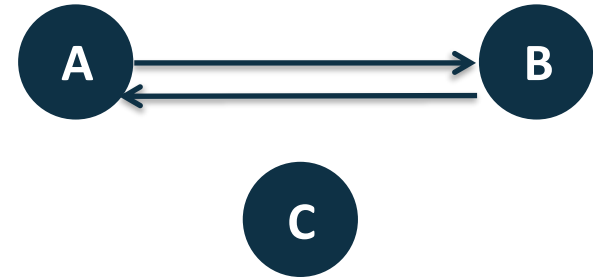
$\text{link}(1,2)$

$\text{link}(2,3)$

$\neg (\forall x. \exists y. \text{link}(x, y))$

Example: Graphs

What kinds of symbols do we need to reason about directed graphs?



Signature:

Function symbols:

- 1, 2, 3, 4 – arity 0 constants

Relation symbol:

- $\text{link}(x,y)$ – true if there is an edge from x to y (link has arity 2)

these were probably bad names for our constants
moral of the story: names don't tell us anything

A first-order model M over our signature:

$U = \{A, B, C\}$

$\text{FUN}(1) = A \quad \text{FUN}(2) = B \quad \text{FUN}(3) = A \quad \text{FUN}(4) = A$

$\text{REL}(\text{link}) = \{ (A, B), (B, A) \}$

Facts:

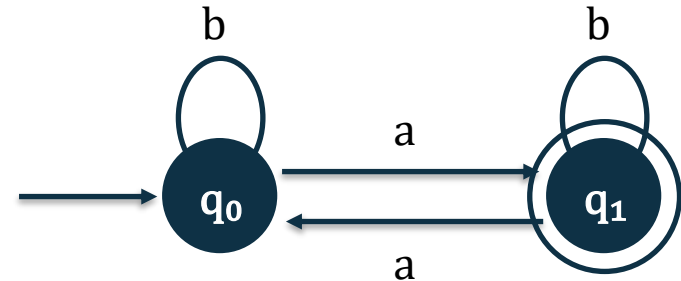
$\text{link}(1, 2)$

$\text{link}(2, 3)$

$\neg (\forall x. \exists y. \text{link}(x, y))$

Example: Automata

What kinds of symbols do we need to reason about the states of an automaton and how it transitions from one to the next?



Signature:

Function symbols:

- start -- the start state (start has arity 0)
- $a(q)$ -- state transition on symbol a (a has arity 1)
- $b(q)$ -- state transition on symbol b (b has arity 1)

Relation symbol:

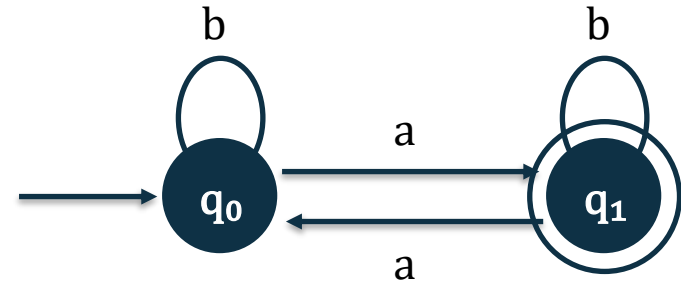
- $\text{final}(q)$ -- true if q is a final state (final has arity 1)

Some facts about our automaton:

$$\neg \text{final}(\text{start}) \quad \text{final}(a(\text{start})) \quad \forall x. b(x) = x \quad \text{start} = a(a(\text{start}))$$

Example: Automata

What kinds of symbols do we need to reason about the states of an automaton and how it transitions from one to the next?



Signature:

Function symbols:

- start -- the start state (start has arity 0)
- $a(q)$ -- state transition on symbol a (a has arity 1)
- $b(q)$ -- state transition on symbol b (b has arity 1)

Relation symbol:

- $\text{final}(q)$ -- true if q is a final state (final has arity 1)

A first-order model M:

$$U = \{ q_0, q_1 \}$$

$$\text{FUN}(\text{start}) = q_0 \quad \text{FUN}(a) = \text{fun } q \rightarrow \text{if } q = q_0 \text{ then } q_1 \text{ else } q_0 \quad \text{REL}(\text{final}) = \{q_1\}$$

$$\text{FUN}(b) = \text{fun } q \rightarrow q$$

Some Facts:

$$\begin{aligned} &\neg \text{final}(\text{start}) \\ &\quad \text{final}(a(\text{start})) \\ &\forall x. b(x) = x \\ &\text{start} = a(a(\text{start})) \end{aligned}$$

A Standard Signature for Linear Arithmetic

Consider the linear arithmetic signature $\Sigma_{LA} =$

- $+$ -- “addition” (arity 2)
- $-$ -- “negation” (arity 1)
- 0 -- “zero” (arity 0)
- 1 -- “one” (arity 0)
- $<$ -- “less than” (arity 2)

The standard model $\mathbb{Z}$

$U^{\mathbb{Z}} =$ **all integers**

$+^{\mathbb{Z}}$ addition, $-^{\mathbb{Z}}$ negation

$0^{\mathbb{Z}}$ zero, $1^{\mathbb{Z}}$ one

$<^{\mathbb{Z}}$ less-than

A two-element model N

$U^N = \{0, 1\}$

$+^N(a, b) = (a + b) \bmod 2$

$-^N(a) = a, \quad 0^N = 0, \quad 1^N = 1$

$<^N = \emptyset$

Once again, the signature fixes the **arities** of the symbols, not their **meaning**.

One model may satisfy some formula that the other does not.

Meaning of Terms

Recall our terms t :

$t ::= x \mid f(t_1, \dots, t_n) \quad \text{where } n = ar(f)$

A **valuation** $\rho : V \rightarrow U^M$ assigns each variable an element of the universe.

Given a valuation ρ and a first-order model M , we can assign a meaning to each term.

To do so, we define a “meaning function,” which is like an evaluator:

let $eval\ M\ \rho\ t = \dots$

But we define our meaning function using notation that looks like this:

$$\llbracket t \rrbracket(M, \rho) = \dots$$

Read that as saying “the meaning of t with respect to model M and valuation ρ is ...”

(think of “ $\llbracket \dots \rrbracket$ ” as the name of the evaluation function)

Meaning of Terms

Recall our terms t :

$$t ::= x \mid f(t_1, \dots, t_n) \quad \text{where } n = \text{ar}(f)$$

A **valuation** $\rho : V \rightarrow U^M$ assigns each variable an element of the universe.

Meaning of Terms:

$$\llbracket x \rrbracket(M, \rho) = \rho(x)$$

$$\llbracket f(t_1, \dots, t_n) \rrbracket(M, \rho) = f^M(\llbracket t_1 \rrbracket(M, \rho), \dots, \llbracket t_n \rrbracket(M, \rho)) \quad (\text{recall } f^M \text{ is the meaning of } f \text{ in model } M)$$

f is interpreted as the associated function from the model

argument terms are interpreted recursively

and you just apply the function to the interpreted arguments

Meaning of Terms

Recall our terms t :

$t ::= x \mid f(t_1, \dots, t_n)$ *where $n = ar(f)$*

A **valuation** $\rho : V \rightarrow U^M$ assigns each variable an element of the universe.

Meaning of Terms:

$\llbracket x \rrbracket(M, \rho) = \rho(x)$

$\llbracket f(t_1, \dots, t_n) \rrbracket(M, \rho) = f^M(\llbracket t_1 \rrbracket(M, \rho), \dots, \llbracket t_n \rrbracket(M, \rho))$ (recall f^M is the meaning of f in model M)

In OCaml:

```
let rec eval M p t =  
  match t with  
  | Var x -> p x  
  | App (f, ts) -> (M.FUN f) (List.map (eval M p) ts)
```

interpret variables using the valuation

interpret function symbols using the model M
and apply that function to values arising
from recursively evaluating the subterms

Meaning of Terms

$$\llbracket x \rrbracket (M, \rho) = \rho(x)$$

$$\llbracket f(t_1, \dots, t_n) \rrbracket (M, \rho) = f^M(\llbracket t_1 \rrbracket (M, \rho), \dots, \llbracket t_n \rrbracket (M, \rho))$$

An Example:

$\mathbb{Z}, \rho = \{x \mapsto 3, y \mapsto -5\}$: (model of the integers)

$$\llbracket x + (1 + y) \rrbracket (\mathbb{Z}, \rho)$$

$$= 3 +^{\mathbb{Z}} (1 +^{\mathbb{Z}} -5)$$

$$= -1$$

- use the substitution and the integer operations

$\mathbb{N}, \rho = \{x \mapsto 1, y \mapsto 1\}$:

(2-element model)

$$\llbracket x + (1 + y) \rrbracket (\mathbb{N}, \rho)$$

$$= 1 +^{\mathbb{N}} (1 +^{\mathbb{N}} 1)$$

$$= 1 +^{\mathbb{N}} 0 = 1$$

Recall:

$$U^{\mathbb{N}} = \{0, 1\}$$

$$+^{\mathbb{N}}(a, b) = (a + b) \bmod 2$$

$$-^{\mathbb{N}}(a) = a, \quad 0^{\mathbb{N}} = 0, \quad 1^{\mathbb{N}} = 1$$

$$<^{\mathbb{N}} = \emptyset$$

Satisfaction: $M, \rho \models F$

$M, \rho \models F$: “M satisfies F when the free variables of F are interpreted by ρ .”

evaluate the subterms and check
to see whether the tuple is in the relation r
from the model

$M, \rho \models r(t_1, \dots, t_n)$ iff $(\llbracket t_1 \rrbracket(M, \rho), \dots, \llbracket t_n \rrbracket(M, \rho)) \in r^M$

$M, \rho \models s = t$ iff $\llbracket s \rrbracket(M, \rho) = \llbracket t \rrbracket(M, \rho)$

$M, \rho \models \text{true}$

always

$M, \rho \models \text{false}$

never

evaluate the terms s and t and check
if they are the same value u from U

$M, \rho \models \neg F$

iff $M, \rho \not\models F$

$M, \rho \models F \wedge G$

iff $M, \rho \models F$ and $M, \rho \models G$

$M, \rho \models F \vee G$

iff $M, \rho \models F$ or $M, \rho \models G$

$M, \rho \models F \rightarrow G$

iff $M, \rho \not\models F$ or $M, \rho \models G$

is there some way to satisfy F by
extending the valuation at v?

$M, \rho \models \exists v. F$

iff for **some** $m \in U^M$, $M, \rho\{v \mapsto m\} \models F$

$M, \rho \models \forall v. F$

iff for **all** $m \in U^M$, $M, \rho\{v \mapsto m\} \models F$

is F satisfied for all extensions (at v) of the valuation?

Quantifiers

$\exists$ is a big “or”

$F[m_1] \vee F[m_2] \vee F[m_3] \vee \dots$

$\forall$ is a big “and”

$F[m_1] \wedge F[m_2] \wedge F[m_3] \wedge \dots$

The universe may be **infinite**, so this is a definition, not an algorithm.

Does there exist a model for this formula? Undecidable in general.

Propositional vs. First-Order

	Propositional	First-order
Vocabulary	atoms P	signature $\Sigma = \langle F, R, ar \rangle$
Structure	$M : P \rightarrow \{\text{true}, \text{false}\}$	universe U^M + meaning of every symbol
Judgement	$M \models F$	$M, \rho \models F$
# structures	2^n (finite)	infinitely many; universes may be infinite

Summary: First-Order Logic

Syntax:

- A signature $\Sigma = \langle F, R, ar \rangle$ names function and relation symbols.
- Terms denote objects; formulas add relations, $=$, $\forall$ and $\exists$.

Semantics:

- A structure gives a non-empty universe and a meaning to every symbol.
- A valuation ρ gives meaning to free variables.
- $M, \rho \models F$; quantifiers range over the entire universe.