Classical First-Order Logic

Software Formal Verification

Maria João Frade

Departmento de Informática
Universidade do Minho

2009/2010

Introduction

First-order logic (FOL) is a richer language than propositional logic. Its lexicon contains not only the symbols $\wedge$, $\vee$, $\neg$, and $\rightarrow$ (and parentheses) from propositional logic, but also the symbols $\exists$ and $\forall$ for "there exists" and "for all", along with various symbols to represent variables, constants, functions, and relations.

There are two sorts of things involved in a first-order logic formula:

- *terms*, which denote the objects that we are talking about;
- *formulas*, which denote truth values.

Examples:

> *"Not all birds can fly."*
> *"Every child is younger than its mother."*
> *"Andy and Paul have the same maternal grandmother."*

Syntax

- *Variables:* $x, y, z, \ldots \in \mathcal{X}$ (represent arbitrary elements of an underlying set)
- *Constants:* $a, b, c, \ldots \in \mathcal{C}$ (represent specific elements of an underlying set)
- *Functions:* $f, g, h, \ldots \in \mathcal{F}$ (every function f as a fixed arity, $\mathrm{ar}(f)$)
- *Predicates:* $P, Q, R, \ldots \in \mathcal{P}$ (every predicate P as a fixed arity, $\mathrm{ar}(P)$)
- *Fixed logical symbols:* $\top$, $\bot$, $\wedge$, $\vee$, $\neg$, $\forall$, $\exists$
- *Fixed predicate symbol:* $=$ for "equals" ("first-order logic with equality")

Syntax

Terms

The set $\mathcal{T}$, of *terms* of FOL, is given by the abstract syntax

$$\mathcal{T} \ni t \ ::= \ x \mid c \mid f(t_1, \ldots, t_{\mathrm{ar}(f)})$$

Formulas

The set $\mathcal{L}$, of *formulas* of FOL, is given by the abstract syntax

$$\begin{array}{rcl} \mathcal{L} \ni \phi, \psi & ::= & \bot \mid \top \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \to \psi \mid t_1 = t_2 \\ & & \mid \forall x.\, \phi \mid \exists x.\, \phi \mid P(t_1, \ldots, t_{\mathrm{ar}(P)}) \end{array}$$

In absence of parentheses, we adopt the following convention about precedence. Ranging from the highest precedence to the lowest, we have respectively: $\neg$, $\wedge$, $\vee$ and $\to$. Finally we have that $\to$ binds more tightly than $\forall$ and $\exists$. Implication is right-associative.

Free and bound variables

- The *free variables* of a formula ϕ are those variables occurring in ϕ that are not quantified. $\mathrm{FV}(\phi)$ denotes the set of free variables occurring in ϕ.
- The *bound variables* of a formula ϕ are those variables occurring in ϕ that do have quantifiers. $\mathrm{BV}(\phi)$ denote the set of bound variables occurring in ϕ.

Note that variables can have both free and bound occurrences within the same formula. Let ϕ be $\exists x.\, R(x,y) \wedge \forall y.\, P(y,x)$, then

$$\mathrm{FV}(\phi) = \{y\} \quad \text{and} \quad \mathrm{BV}(\phi) = \{x, y\}.$$

- A formula ϕ is *closed* if it does not contain any free variables.
- If $\mathrm{FV}(\phi) = \{x_1, \ldots, x_n\}$, then
  - its *universal closure* is $\forall x_1 \ldots \forall x_n.\, \phi$
  - its *existential closure* is $\exists x_1 \ldots \exists x_n.\, \phi$

Substitution

Substitution

- We define $u[t/x]$ to be the term obtained by replacing each occurrence of variable x in u with t.
- We define $\phi[t/x]$ to be the formula obtained by replacing each free occurrence of variable x in ϕ with t.

Care must be taken, because substitutions can give rise to undesired effects.

Given a term t, a variable x and a formula ϕ, we say that *t is free for x in ϕ* if no free x in ϕ occurs in the scope of $\forall z$ or $\exists z$ for any variable z occurring in t.

From now on we will assume that all substitutions satisfy this condition. That is when performing the $\phi[t/x]$ we are always assuming that t is free for x in ϕ.

Substitution

Convention

We write $\phi(x_1, \ldots, x_n)$ to denote a formula having free variables $x_1, \ldots, x_n$. We write $\phi(t_1, \ldots, t_n)$ to denote the formula obtained by replacing each free occurrence of x_i in ϕ with the term t_i. When using this notation, it should always be assumed that each t_i is free for x_i in ϕ. Also note that when writhing $\phi(x_1, ..., x_n)$ we do not mean that $x_1, ..., x_n$ are the only free variables of ϕ.

A *sentence* of first-order logic is a formula having no free variables.

- The presence of free variables distinguishes formulas from sentences.
- This distinction did not exist in propositional logic.

Semantics

Vocabulary

A *vocabulary* (or *signature*) is a set of function, relation, and constant symbols.

$\mathcal{C} \cup \mathcal{F} \cup \mathcal{P}$ is a vocabulary.

$\mathcal{V}$-structure

Let $\mathcal{V}$ be a vocabulary. A *$\mathcal{V}$-structure* consists of a nonempty underlying set U along with an interpretation of $\mathcal{V}$. An *interpretation* of $\mathcal{V}$ assigns:

- an element of U to each constant in $\mathcal{V}$,
- a function from U^n to U to each n-ary function in $\mathcal{V}$, and
- a subset of U^n to each n-ary relation in $\mathcal{V}$.

Model

We say that $\mathcal{M}$ is a *model* of $(\mathcal{C}, \mathcal{F}, \mathcal{P})$ iff $\mathcal{M}$ is a $(\mathcal{C} \cup \mathcal{F} \cup \mathcal{P})$-structure.

Semantics

An alternative definition:

Model

A *model* $\mathcal{M}$ of $(\mathcal{C}, \mathcal{F}, \mathcal{P})$ consists of the following set of data:

- a non-empty set U, the universe of concrete values;
- for each constant symbol $c \in \mathcal{C}$, a concrete element $\mathcal{M}(c) \in U$;
- for each $f \in \mathcal{F}$, a concrete function $\mathcal{M}(f) : U^{\mathrm{ar}(f)} \to U$;
- for each $P \in \mathcal{P}$, a subset $\mathcal{M}(P) \subseteq U^{\mathrm{ar}(P)}$.

The set U is the *interpretation domain* (or *interpretation universe*) of $\mathcal{M}$.

Semantically, one recognises the special role of equality by imposing on an interpretation function $\mathcal{M}(=)$ to be actual equality on the set U of $\mathcal{M}$. Thus, (a, b) is in the set $\mathcal{M}(=)$ iff a and b are the same elements in the set U.

Semantics

Assignment

An *assignment* or *environment* for a universe U of concrete values is a function $\alpha : \mathcal{X} \to U$.

We denote by $\alpha[x \mapsto a]$ the assignment which maps x to a and any other variable y to $\alpha(y)$.

Given a model $\mathcal{M}$ for $(\mathcal{C}, \mathcal{F}, \mathcal{P})$ with interpretation domain U, and given an assignment $\alpha : \mathcal{X} \to U$, we define an interpretation function for terms, $\alpha_{\mathcal{M}} : \mathcal{T} \to U$, as follows:

$$\begin{array}{lcl} \alpha_{\mathcal{M}}(x) & = & \alpha(x) \\ \alpha_{\mathcal{M}}(c) & = & \mathcal{M}(c) \\ \alpha_{\mathcal{M}}(f(t_1, \ldots, t_n)) & = & \mathcal{M}(f)(\alpha_{\mathcal{M}}(t_1), \ldots, \alpha_{\mathcal{M}}(t_n)) \end{array}$$

Semantics

Satisfaction relation

Given a model $\mathcal{M}$ for $(\mathcal{C}, \mathcal{F}, \mathcal{P})$ and given an assignment $\alpha : \mathcal{X} \to U$, we define the *satisfaction relation* $\mathcal{M} \models_\alpha \phi$ for each logical formula ϕ over $(\mathcal{C}, \mathcal{F}, \mathcal{P})$ as follows:

$$
\begin{array}{lll}
\mathcal{M} \models_\alpha \top & & \\
\mathcal{M} \not\models_\alpha \bot & & \\
\mathcal{M} \models_\alpha P(t_1, \ldots, t_n) & \text{iff} & (\alpha_{\mathcal{M}}(t_1), \ldots, \alpha_{\mathcal{M}}(t_n)) \in \mathcal{M}(P) \\
\mathcal{M} \models_\alpha \neg\phi & \text{iff} & \mathcal{M} \not\models_\alpha \phi \\
\mathcal{M} \models_\alpha \phi \wedge \psi & \text{iff} & \mathcal{M} \models_\alpha \phi \text{ and } \mathcal{M} \models_\alpha \psi \\
\mathcal{M} \models_\alpha \phi \vee \psi & \text{iff} & \mathcal{M} \models_\alpha \phi \text{ or } \mathcal{M} \models \psi \\
\mathcal{M} \models_\alpha \phi \to \psi & \text{iff} & \mathcal{M} \not\models_\alpha \phi \text{ or } \mathcal{M} \models_\alpha \psi \\
\mathcal{M} \models_\alpha \forall x.\, \phi & \text{iff} & \mathcal{M} \models_{\alpha[x \mapsto a]} \phi \text{ for all } a \in U \\
\mathcal{M} \models_\alpha \exists x.\, \phi & \text{iff} & \mathcal{M} \models_{\alpha[x \mapsto a]} \phi \text{ for some } a \in U
\end{array}
$$

If ϕ is a sentence we often drop α and write $\mathcal{M} \models \phi$.

Validity, satisfiability, and contradiction

If $\mathcal{M} \models \phi$ holds, then we say that $\mathcal{M}$ *models* ϕ, or that ϕ *holds* in $\mathcal{M}$, or simply, that ϕ *is true* in $\mathcal{M}$.

A sentence ϕ is

valid	iff	it holds in every model. We write $\models \phi$. A valid sentence is called a *tautology*.
satisfiable	iff	it holds in some model.
unsatisfiable	iff	there is no model in which ϕ is true. An unsatisfiable sentence is called a *contradiction*.

Validity, satisfiability, and contradiction

The definition of satisfiability can be extended to apply to all formulas of first-order logic (not just sentences).

The formula $\phi(x_1, \ldots, x_n)$ is *satisfiable* if and only if the sentence $\forall x_1 \ldots \forall x_n . \phi(x_1, \ldots, x_n)$ (its universal closure) is satisfiable.

Consequence and equivalence

- $\phi \models \psi$ iff for every model $\mathcal{M}$, if $\mathcal{M} \models \phi$ then $\mathcal{M} \models \psi$. We say ψ is a *consequence* of ϕ.
- $\phi \equiv \psi$ iff $\phi \models \psi$ and $\psi \models \phi$. We say ϕ and ψ are *equivalent*.
- Let $\Gamma = \{\phi_1, \phi_2, \phi_3, \ldots\}$ be a set of sentences.
  $\mathcal{M} \models \Gamma$ iff $\mathcal{M} \models \phi_i$ for each sentence ϕ_i in Γ. We say $\mathcal{M}$ *models* Γ.
  $\Gamma \models \psi$ iff $\mathcal{M} \models \Gamma$ implies $\mathcal{M} \models \psi$ for every model $\mathcal{M}$. We say ψ is a *consequence* of Γ.

Proposition

- $\phi \models \psi$ iff $\models \phi \rightarrow \psi$
- $\Gamma \models \psi$ and Γ finite iff $\models \bigwedge \Gamma \rightarrow \psi$

Consistency

Let $\Gamma = \{\phi_1, \phi_2, \phi_3, \dots\}$ be a set of sentences.

- Γ is *consistent* or *satisfiable* iff there is a model for Γ.
- We say that Γ is *inconsistent* iff it is not consistent and denote this by $\Gamma \models \bot$.

Proposition

- $\{\phi, \neg\phi\} \models \bot$
- If $\Gamma \models \bot$ and $\Gamma \subseteq \Delta$, then $\Delta \models \bot$.
- $\Gamma \models \phi$ iff $\Gamma, \neg\phi \models \bot$

Substitution

- Formula ψ is a *subformula* of formula ϕ if it occurs syntactically within ϕ.
- Formula ψ is a *strict subformula* of ϕ if ψ is a subformula of ϕ and $\psi \neq \phi$

Substitution theorem

Suppose $\phi \equiv \psi$. Let θ be a formula that contains ϕ as a subformula. Let θ' be the formula obtained by safe replacing (i.e., avoiding the capture of free variables of ϕ) some occurrence of ϕ in θ with ψ. Then $\theta \equiv \theta'$.

Adquate sets of connectives for FOL

As in propositional logic, there is some redundancy among the connectives and quantifiers.

Note that in classical first-order logic

$$\forall x.\, \phi \quad \equiv \quad \neg \exists x.\, \neg \phi$$

$$\exists x.\, \phi \quad \equiv \quad \neg \forall x.\, \neg \phi$$

Decidability

Given formulas ϕ and ψ as input, we may ask:

Decision problems

Validity problem:	"Is ϕ valid ?"
Satisfiability problem:	"Is ϕ satisfiable ?"
Consequence problem:	"Is ψ a consequence of ϕ ?"
Equivalence problem:	"Are ϕ and ψ equivalent ?"

These are, in some sense, variations of the same problem.

ϕ is valid	iff	$\neg\phi$ is unsatisfiable
$\phi \models \psi$	iff	$\neg(\phi \rightarrow \psi)$ is unsatisfiable
$\phi \equiv \psi$	iff	$\phi \models \psi$ and $\psi \models \phi$
ϕ is satisfiable	iff	$\neg\phi$ is not valid

Decidability

A *solution* to a decision problem is a program that takes problem instances as input and always terminates, producing a correct "yes" or "no" output.

- A decision problem is *decidable* if it has a solution.
- A decision problem is *undecidable* if it is not decidable.

In PL we could, in theory, compute a truth table to determine whether or not a formula is satisfiable. In FOL, we would have to check every model to do this.

Theorem (Church & Turing)

- The decision problem of validity in first-order logic is undecidable: no program exists which, given any ϕ, decides whether $\models \phi$.
- The decision problem of satisfiability in first-order logic is undecidable: no program exists which, given any ϕ, decides whether ϕ is satisfiable.

Decidability

However, there is a procedure that halts and says "yes" if ϕ is valid.

A decision problem is *semi-decidable* if exists a procedure that, given an input,

- halts and answers "yes" iff "yes" is the correct answer,
- halts and answers "no" if "no" is the correct answer, or
- does not halt if "no" is the correct answer

Unlike a decidable problem, the procedure is only guaranteed to halt if the correct answer is "yes".

The decision problem of validity in first-order logic is semi-decidable.

Decidability

Methods for the Validity problem in fist-order logic:

- Semantic Tableaux
- Resolution for first-order logic
- SLD-resolution
- ...

Although first-order validity is undecidable, there are special simple fragments of FOL where it is decidable, e.g.

- *Monadic predicate logic* (i.e. only unary predicates and no function symbols) is decidable.
- *The Bernays-Schönfinkel class* of formulas (i.e. formulas that can be written with all quantiers appearing at the beginning of the formula with existentials before universals and that do not contain any function symbols) is decidable.

Proof system

- As with any logic, the semantics of first-order logic yield rules for deducing the truth of one sentence from that of another.
- The proof system we present for FOL is Natural Deduction in sequent style.
- The sequent $\Gamma \vdash \phi$ indicates that ϕ can be formally derived from the set of assumptions Γ.
- Basically, we just have to add rules for quantifiers and the equality.

Other proof systems for FOL: *Hilbert system* and *sequent calculus*.

Natural deduction rules for $\mathcal{L}$ (in sequent style)

$$\frac{}{\Gamma \vdash \top}\ \text{true} \qquad \frac{\phi \in \Gamma}{\Gamma \vdash \phi}\ \text{assumption} \qquad \frac{\Gamma \vdash \phi \quad \Gamma \subset \Gamma'}{\Gamma' \vdash \phi}\ \text{monotonicity}$$

$$\frac{\Gamma \vdash \phi \quad \Gamma \vdash \psi}{\Gamma \vdash \phi \wedge \psi}\ \wedge_I \qquad \frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \psi}\ \wedge_{E1} \qquad \frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \psi}\ \wedge_{E2}$$

$$\frac{\Gamma \vdash \phi}{\Gamma \vdash \phi \vee \psi}\ \vee_{I1} \qquad \frac{\Gamma \vdash \psi}{\Gamma \vdash \phi \vee \psi}\ \vee_{I2} \qquad \frac{\Gamma \vdash \phi \vee \psi \quad \Gamma, \phi \vdash \theta \quad \Gamma, \psi \vdash \theta}{\Gamma \vdash \theta}\ \vee_E$$

$$\frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \to \psi}\ \to_I \qquad \frac{\Gamma \vdash \phi \quad \Gamma \vdash \phi \to \psi}{\Gamma \vdash \psi}\ \to_E$$

$$\frac{\Gamma, \phi \vdash \bot}{\Gamma \vdash \neg\phi}\ \neg_I \qquad \frac{\Gamma \vdash \phi \quad \Gamma \vdash \neg\phi}{\Gamma \vdash \bot}\ \neg_E$$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash \phi}\ \bot_E \qquad \frac{\Gamma \vdash \neg\neg\phi}{\Gamma \vdash \phi}\ \neg\neg_E$$

Natural deduction rules for $\mathcal{L}$ (in sequent style)

Proof rules for equality and quantifiers.

$$\frac{}{\Gamma \vdash t = t}\ =_I \qquad \frac{\Gamma \vdash \phi(t_1) \quad \Gamma \vdash t_1 = t_2}{\Gamma \vdash \phi(t_2)}\ =_E$$

$$\frac{\Gamma \vdash \phi(y)}{\Gamma \vdash \forall x.\, \phi(x)}\ \forall_I\ \textbf{(a)} \qquad \frac{\Gamma \vdash \forall x.\, \phi(x)}{\Gamma \vdash \phi(t)}\ \forall_E$$

$$\frac{\Gamma \vdash \phi(t)}{\Gamma \vdash \exists x.\, \phi(x)}\ \exists_I \qquad \frac{\Gamma \vdash \exists x.\, \phi(x) \quad \Gamma, \phi(y) \vdash \theta}{\Gamma \vdash \theta}\ \exists_E\ \textbf{(b)}$$

(a) y must not occur free in Γ or $\phi(x)$.

(b) y must not occur free in Γ, $\phi(x)$ or θ.

Formal proof

Deduction is purely syntactical.

A *formal proof* is a finite sequence of statements of the form "$\Gamma \vdash \phi$" each of which follows from the previous statements by one of the basic rules. We say that ψ can be *derived* from Γ if there is a formal proof concluding with the statement $\Gamma \vdash \psi$.

Example: $t_1 = t_2 \vdash t_2 = t_1$

	Statements	Justification
1.	$t_1 = t_2 \vdash t_1 = t_2$	assumption
2.	$t_1 = t_2 \vdash t_1 = t_1$	$=_I$ 1
3.	$t_1 = t_2 \vdash t_2 = t_1$	$=_E$ 2, 1

Recall that in a proof-assistant the proof is usually developed backwards.

Formal proof

Example: $P(t), (\forall x.\, P(x) \to \neg Q(x)) \vdash \neg Q(t)$

	Statements	Justification
1.	$P(t), \forall x.\, P(x) \to \neg Q(x) \vdash P(t)$	assumption
2.	$P(t), \forall x.\, P(x) \to \neg Q(x) \vdash \forall x.\, P(x) \to \neg Q(x)$	assumption
3.	$P(t), \forall x.\, P(x) \to \neg Q(x) \vdash P(t) \to \neg Q(t)$	$\forall_E$ 2
4.	$P(t), \forall x.\, P(x) \to \neg Q(x) \vdash \neg Q(t)$	$\to_E$ 1, 3

Example: $\forall x.\, P(x) \vdash \exists x.\, P(x)$

	Statements	Justification
1.	$\forall x.\, P(x) \vdash \forall x.\, P(x)$	assumption
2.	$\forall x.\, P(x) \vdash P(t)$	$\forall_E$ 1
3.	$\forall x.\, P(x) \vdash \exists x.\, P(x)$	$\exists_I$ 2

Formal proof

Example: $\exists x.\, \neg\psi(x) \vdash \neg\forall x.\, \psi(x)$

	Statements	Justification
1.	$\exists x.\, \neg\psi(x), \forall x.\, \psi(x) \vdash \exists x.\, \neg\psi(x)$	assumption
2.	$\exists x.\, \neg\psi(x), \forall x.\, \psi(x), \neg\psi(x_0) \vdash \forall x.\, \psi(x)$	assumption
3.	$\exists x.\, \neg\psi(x), \forall x.\, \psi(x), \neg\psi(x_0) \vdash \neg\psi(x_0)$	assumption
4.	$\exists x.\, \neg\psi(x), \forall x.\, \psi(x), \neg\psi(x_0) \vdash \psi(x_0)$	$\forall_E$ 2
5.	$\exists x.\, \neg\psi(x), \forall x.\, \psi(x), \neg\psi(x_0) \vdash \bot$	$\neg_E$ 4, 3
6.	$\exists x.\, \neg\psi(x), \forall x.\, \psi(x) \vdash \bot$	$\exists_E$ 1, 5
7.	$\exists x.\, \neg\psi(x) \vdash \neg\forall x.\, \psi(x)$	$\neg_I$ 6

Soundness, completeness and compactness

Soundness

If $\Gamma \vdash \phi$, then $\Gamma \models \phi$.

Therefore, if $\vdash \psi$, then ψ is a tautology; and if $\vdash \neg\psi$, then ψ is a contradiction.

Completeness

If $\Gamma \models \phi$, then $\Gamma \vdash \phi$.

Compactness

A (possible infinite) set of sentences Γ is satisfiable if and only if every finite subset of Γ is satisfiable.

Exercises

Prove that the following sequents hold

- $\vdash \neg\forall x.\psi(x) \rightarrow \exists x.\neg\psi(x)$ (classical)
- $(\forall x.\phi(x)) \vee (\forall x.\psi(x)) \vdash \forall x.\phi(x) \vee \psi(x)$
- $\exists x.\exists y.\phi(x,y) \vdash \exists y.\exists x.\phi(x,y)$

Proof checking mathematical statements

- Mathematics is usually presented in an informal but precise way.

> In situation Γ we have ψ.
> Proof. p. QED

- In Logic, Γ, ψ become formal objects and proofs can be formalized as a derivation (following some precisely given set of rules).

> $\Gamma \vdash_L \psi$
> Proof. p. QED

Proof-assistants

A *proof-assistant* is the combination of a *proof-checker* with a *proof-development system* to help on the formalization process and the interactive development of proofs.

In a proof-assistant, after formalizing the primitive notions of the theory (under study), the user develops the proofs interactively by means of (proof) *tactics*, and when a proof is finished a *"proof-term"* is created.

Machine assisted theorem proving:

- helps to deal with large problems;
- prevents us from overseeing details;
- does the bookkeeping of the proofs.

Proof-assistants

There are many proof-assistants for many different logics: fist-order logic, higher-order logic, modal logic, ...

We can mention as examples:

- Coq - http://coq.inria.fr/
- Isabelle - http://isabelle.in.tum.de/
- HOL - http://www.cl.cam.ac.uk/research/hvg/HOL
- Agda - http://wiki.portal.chalmers.se/agda/
- PVS - http://pvs.csl.sri.com/
- ...

The Coq proof-assistant

Demo

http://coq.inria.fr/