Automated Reasoning for Software Engineering

L. Georgieva and A. Ireland

School of Mathematical and Computer Sciences

Heriot-Watt University

Lilia: Office G.50

Email: lilia@macs.hw.ac.uk

Part 1: First-Order Logic

- formalizes fundamental mathematical concepts
- expressive (Turing-complete)
- not too expressive (not axiomatizable: natural numbers, uncountable sets)
- rich structure of decidable fragments
- rich model and proof theory

First-order logic is also called (first-order) predicate logic.

1.1 Syntax

- non-logical symbols (domain-specific)
  terms, atomic formulas
- logical symbols (domain-independent)
  Boolean combinations, quantifiers

Signature

Usage: fixing the alphabet of non-logical symbols

$$\Sigma = (\Omega, \Pi),$$

where

- Ω a set of function symbols f with arity $n \geq 0$, written f/n,
- Π a set of predicate symbols p with arity $m \geq 0$, written p/m.

If $n = 0$ then f is also called a constant (symbol). If $m = 0$ then p is also called a propositional variable. We use letters P, Q, R, S, to denote propositional variables.

Refined concept for practical applications: many-sorted signatures (corresponds to simple type systems in programming languages);

Variables

Predicate logic admits the formulation of abstract, schematic assertions. (Object) variables are the technical tool for schematization.

We assume that

$$X$$

is a given countably infinite set of symbols which we use for (the denotation of) variables.

Terms

Terms over Σ (resp., Σ-terms) are formed according to these syntactic rules:

$$
\begin{array}{rcll}
s, t, u, v & ::= & x & , x \in X \quad \text{(variable)} \\
& | & f(s_1, ..., s_n) & , f/n \in \Omega \quad \text{(functional term)}
\end{array}
$$

By $T_\Sigma(X)$ we denote the set of Σ-terms (over X). A term not containing any variable is called a *ground term*. By T_Σ we denote the set of Σ-ground terms.

In other words, terms are formal expressions with well-balanced brackets which we may also view as marked, ordered trees. The markings are function symbols or variables. The nodes correspond to the *subterms* of the term. A node v that is marked with a function symbol f of arity n has exactly n subtrees representing the n immediate subterms of v.

Atoms

Atoms (also called atomic formulas) over Σ are formed according to this syntax:

$$
\begin{array}{rcll}
A, B & ::= & p(s_1, \dots, s_m) & ,\ p/m \in \Pi \\
\Big[& | & (s \approx t) & \text{(equation)} \quad \Big]
\end{array}
$$

Whenever we admit equations as atomic formulas we are in the realm of *first-order logic with equality*. Admitting equality does not really increase the expressiveness of first-order logic. But deductive systems where equality is treated specifically can be much more efficient.

Literals

$$
\begin{array}{lcll}
L & ::= & A & \text{(positive literal)} \\
 & | & \neg A & \text{(negative literal)}
\end{array}
$$

Clauses

$$
\begin{array}{lcll}
C, D & ::= & \bot & \text{(empty clause)} \\
 & \mid & L_1 \vee \ldots \vee L_k, \;\; k \geq 1 & \text{(non-empty clause)}
\end{array}
$$

First-Order Formulas

$F_\Sigma(X)$ is the set of first-order formulas over Σ defined as follows:

$$
\begin{array}{rcll}
F, G, H & ::= & \bot & \text{(falsum)} \\
& | & \top & \text{(verum)} \\
& | & A & \text{(atomic formula)} \\
& | & \neg F & \text{(negation)} \\
& | & (F \land G) & \text{(conjunction)} \\
& | & (F \lor G) & \text{(disjunction)} \\
& | & (F \implies G) & \text{(implication)} \\
& | & (F \equiv G) & \text{(equivalence)} \\
& | & \forall x F & \text{(universal quantification)} \\
& | & \exists x F & \text{(existential quantification)}
\end{array}
$$

Notational Conventions

- We omit brackets according to the following rules:
  - $\neg \quad >_p \quad \vee \quad >_p \quad \wedge \quad >_p \quad \Longrightarrow \quad >_p \quad \equiv$ (binding precedences)
  - $\vee$ and $\wedge$ are associative and commutative
  - $\Longrightarrow$ is right-associative
- $Qx_1, \ldots, x_n\, F$ abbreviates $Qx_1 \ldots Qx_n\, F$.
- infix-, prefix-, postfix-, or mixfix-notation with the usual operator precedences; examples:

$$
\begin{array}{ccc}
s + t * u & \text{for} & +(s, *(t, u)) \\
s * u \leq t + v & \text{for} & \leq (*(s, u), +(t, v)) \\
-s & \text{for} & -(s) \\
0 & \text{for} & 0()
\end{array}
$$

Example: Peano Arithmetic

$$\Sigma_{PA} = (\Omega_{PA}, \Pi_{PA})$$
$$\Omega_{PA} = \{0/0, +/2, */2, s/1\}$$
$$\Pi_{PA} = \{\leq /2, < /2\}$$
$$+, *, <, \leq \text{ infix; } * >_p + >_p < >_p \leq$$

Exampes of formulas over this signature are:

$$\forall x, y(x \leq y \equiv \exists z(x + z \approx y))$$
$$\exists x \forall y(x + y \approx y)$$
$$\forall x, y(x * s(y) \approx x * y + x)$$
$$\forall x, y(s(x) \approx s(y) \implies x \approx y)$$
$$\forall x \exists y\ (x < y \land \neg \exists z(x < z \land z < y))$$

Remarks About the Example

We observe that the symbols $\leq$, $<$, 0, s are redundant as they can be defined in first-order logic with equality just with the help of $+$. The first formula defines $\leq$, while the second defines zero. The last formula, respectively, defines s.

Eliminating the existential quantifiers by Skolemization (cf. below) reintroduces the "redundant" symbols.

Consequently there is a *trade-off* between the complexity of the quantification structure and the complexity of the signature.

Bound and Free Variables

In QxF, $Q \in \{\exists, \forall\}$, we call F the scope of the quantifier Qx. An *occurrence* of a variable x is called bound, if it is inside the scope of a quantifier Qx. Any other occurrence of a variable is called free.

Formulas without free variables are also called closed formulas or sentential forms.

Formulas without variables are called ground.

Example

$$\forall y \quad (\overbrace{\forall x \quad \overbrace{p(x)) \quad \Longrightarrow \quad q(x, y)}^{scope}}^{scope})$$

The occurrence of y is bound, as is the first occurrence of x. The second occurrence of x is a free occurrence.

Substitutions

Substitution is a fundamental operation on terms and formulas that occurs in all inference systems for first-order logic. In the presence of quantification it is surprisingly complex.

By $F[s/x]$ we denote the result of substituting all *free occurrences* of x in F by the term s.

Formally we define $F[s/x]$ by structural induction over the syntactic structure of F by the equations depicted on the next page.

Substitution of a Term for a Free Variable

$$
\begin{aligned}
x[s/x] &= s \\
x'[s/x] &= x' \ ; \quad \text{if } x' \neq x \\
f(s_1, \ldots, s_n)[s/x] &= f(s_1[s/x], \ldots, s_n[s/x]) \\
\bot[s/x] &= \bot \\
\top[s/x] &= \top \\
p(s_1, \ldots, s_n)[s/x] &= p(s_1[s/x], \ldots, s_n[s/x]) \\
(u \approx v)[s/x] &= (u[s/x] \approx v[s/x]) \\
\neg F[s/x] &= \neg(F[s/x]) \\
(F \rho G)[s/x] &= (F[s/x] \rho G[s/x]) \ ; \quad \text{for each binary connective } \rho \\
(QyF)[s/x] &= Qz((F[z/y])[s/x]) \ ; \ \text{with } z \text{ a ``fresh'' variable}
\end{aligned}
$$

Why Substitution is Complicated

We need to make sure that the (free) variables in s are not *captured* upon placing s into the scope of a quantifier, hence the renaming of the bound variable y into a "fresh", that is, previously unused, variable z.

Why this definition of substitution is well-defined will be discussed below.

General Substitutions

In general, substitutions are mappings

$$\boldsymbol{\sigma} : X \longrightarrow T_\Sigma(X)$$

such that the domain of $\boldsymbol{\sigma}$, that is, the set

$$dom(\boldsymbol{\sigma}) = \{x \in X \mid \boldsymbol{\sigma}(x) \neq x\},$$

is finite. The set of variables introduced by $\boldsymbol{\sigma}$, that is, the set of variables occurring in one of the terms $\boldsymbol{\sigma}(x)$, with $x \in dom(\boldsymbol{\sigma})$, is denoted by $codom(\boldsymbol{\sigma})$.

Substitutions are often written as $[s_1/x_1, \ldots, s_n/x_n]$, with x_i pairwise distinct, and then denote the mapping

$$[s_1/x_1, \ldots, s_n/x_n](y) = \begin{cases} s_i, & \text{if } y = x_i \\ y, & \text{otherwise} \end{cases}$$

We also write $x\boldsymbol{\sigma}$ for $\boldsymbol{\sigma}(x)$.

Application of a Substitution

"Homomorphic" extension of $\boldsymbol{\sigma}$ to terms and formulas:

$$
\begin{aligned}
f(s_1, \dots, s_n)\boldsymbol{\sigma} &= f(s_1\boldsymbol{\sigma}, \dots, s_n\boldsymbol{\sigma}) \\
\bot\boldsymbol{\sigma} &= \bot \\
\top\boldsymbol{\sigma} &= \top \\
p(s_1, \dots, s_n)\boldsymbol{\sigma} &= p(s_1\boldsymbol{\sigma}, \dots, s_n\boldsymbol{\sigma}) \\
(u \approx v)\boldsymbol{\sigma} &= (u\boldsymbol{\sigma} \approx v\boldsymbol{\sigma}) \\
\neg F\boldsymbol{\sigma} &= \neg(F\boldsymbol{\sigma}) \\
(F\boldsymbol{\rho}G)\boldsymbol{\sigma} &= (F\boldsymbol{\sigma}\ \boldsymbol{\rho}\ G\boldsymbol{\sigma})\ ; \quad \text{for each binary connective } \boldsymbol{\rho} \\
(Qx\, F)\boldsymbol{\sigma} &= Qz\,(F\,\boldsymbol{\sigma}[x \mapsto z])\ ; \quad \text{with } z \text{ a fresh variable}
\end{aligned}
$$

E: Convince yourself that for the special case $\boldsymbol{\sigma} = [t/x]$ the new definition coincides with our previous definition (modulo the choice of fresh names for the bound variables).

1.2. Semantics

To give semantics to a logical system means to define a notion of truth for the formulas. The concept of truth that we will now define for first-order logic goes back to Tarski.

In classical logic (dating back to Aristoteles) there are “only” two truth values “true” and “false” which we shall denote, respectively, by 1 and 0.

There are multi-valued logics having more than two truth values.

Structures

A Σ-algebra (also called Σ-interpretation or Σ-structure) is a triple

$$\mathcal{A} = (U,\ (f_{\mathcal{A}} : U^n \longrightarrow U)_{f/n \in \Omega},\ (p_{\mathcal{A}} \subseteq U^m)_{p/m \in \Pi})$$

where $U \neq \emptyset$ is a set, called the universe of $\mathcal{A}$.

Normally, by abuse of notation, we will have $\mathcal{A}$ denote both the algebra and its universe.

By Σ-Alg we denote the class of all Σ-algebras.

Assignments

A variable has no intrinsic meaning. The meaning of a variable has to be defined externally (explicitly or implicitly in a given context) by an assignment.

A (variable) assignment, also called a valuation (over a given Σ-algebra $\mathcal{A}$), is a map $\beta : X \longrightarrow \mathcal{A}$.

Variable assignments are the semantic counterparts of substitutions.

Ex: "Standard" Interpretation N for Peano Arithmetic

$$
\begin{aligned}
U_N &= \{0, 1, 2, \ldots\} \\
0_N &= 0 \\
s_N &: n \mapsto n + 1 \\
+_N &: (n, m) \mapsto n + m \\
*_N &: (n, m) \mapsto n * m \\
\leq_N &= \{(n, m) \mid n \text{ less than or equal to } m\} \\
<_N &= \{(n, m) \mid n \text{ less than } m\}
\end{aligned}
$$

Note that N is just one out of many possible Σ_{PA}-interpretations.

Values over N for Sample Terms and Formulas

Under the assignment $\beta : x \mapsto 1, y \mapsto 3$ we obtain

$$
\begin{array}{lcl}
N(\beta)(s(x) + s(0)) & = & 3 \\
N(\beta)(x + y \approx s(y)) & = & 1 \\
N(\beta)(\forall x, y(x + y \approx y + x)) & = & 1 \\
N(\beta)(\forall z\; z \leq y) & = & 0 \\
N(\beta)(\forall x \exists y\; x < y) & = & 1
\end{array}
$$

Part 2: Higher Order Logic and Sequent Calculus

- In order to formally reason about mathematical objects, or programs we need a formal language PVS uses higher order logic.
- Constructions in higher order logic used in PVS:
  - $\neg$ not
  - $\wedge$ and
  - $\vee$ or
  - $\longrightarrow$ if ... then
  - $\leftrightarrow$ if and only if
  - $\forall x : XP(x)$
  - $\exists x : xP(x)$
  - $=$ is equal to
  - $p(t_1, \ldots, t_n)$, $t_1, \ldots, t_n$ are in relationship with each other: $(t_1, \ldots, t_n)$ are called atoms;

Examples

- The atoms $p(t_1, \ldots, t_n)$ can have the form:
  - $a < b$
  - $1 < 1 + 1$
  - even(4), odd(5);
- Examples of formulae are:
  - $\forall x, y : \mathrm{Nat} \leftrightarrow x + 1 < y + 1$
  - $\forall x, y : \mathrm{Nat} \leftrightarrow x < y \longrightarrow x < y + 1$
  - $\forall \mathrm{prime}(p) : \mathrm{Nat} \leftrightarrow \neg \exists x : \mathrm{Nat} 1 < x \text{ and } x < p \wedge \mathrm{divides}(x, p)$
  - $\forall x, y : \mathrm{Real} \ \mathrm{square}(x + y) = \mathrm{square}(x) + \mathrm{square}(y) + 2 * x * y$

Predicate versus higher order logic: what is an order

- Predicates that speak of domain objects are of first-order.
- Predicates that speak of objects of at most ith order are themselves $i + 1$th order.
- Functions that take and return domain objects are of first-order.
- Functions that take and return objects of at most i-th order are themselves $i + 1$-th order.

Example: The induction principle is second order.
$\forall P : Nat \longrightarrow \mathrm{Bool} P0 \wedge \forall n : Nat(P(n) \longrightarrow P(n+1)) \longrightarrow \mathrm{Nat} : P(n)$

Higher-order logic

- When reasoning about physical objects the following principles are considered valid:
  - law of excluded middle: $A \vee \neg A$
  - law of double negation: $\neg\neg A \implies A$

  Example: Either there are errors in the code, or there are no errors in the code.

Intuitionistic or constructive logic

- Mathematical objects.
- The law of excluded middle is not observed.
- To prove $\exists x : Xp(x)$ in Intuitionistic logic means to find a witness t for which $p(t)$ holds.

Sequent calculus for first-order logic

The most important types of deduction systems are:

- natural deduction
  - models the natural style of reasoning;
  - principle of forward reasoning: deriving conclusions, deriving conclusions from the conclusions, etc.
- sequent calculus;
  - conclusions and premises are treated in the same way.
  - the proof consists o judgments rather than conclusions;
- PVS is based on a sequent calculus for higher order classical logic.
- COQ is based on higher order intuitionistic logic with inductive types.

Sequent Calculus for Classical Logic

Definition: A multiset is a set that can distinguish how often en element occurs in it. Alternatively: a list that cannot see the order of its elements.

Examples

1. $A \vee B \; A \wedge B \; A \wedge B$
2. $A \vee B \; A \wedge B \; C \implies D$
3. $A \wedge B \; A \vee B \; A \wedge B$

The first and the last multiset are equal.

Sequents

A sequent is an object of the form:

$$\Gamma \Vdash \Delta$$

where:

- Both Γ and Δ are multisets of formulae.

Meaning: Whenever all of the Γ are true then at least one Δ is true.

Propositional rules

- Axiom:

$$\overline{\Gamma, A \Vdash \Delta, A}$$

- The cut rule:

$$\frac{\Gamma, A \Vdash B \qquad \Gamma \Vdash \Delta, A}{\Gamma \Vdash \Delta}$$

Structural rules

- Weakening (left):

$$\frac{\Gamma \Vdash \Delta}{\Gamma, A \Vdash \Delta}$$

- Weakening (right):

$$\frac{\Gamma \Vdash \Delta}{\Gamma \Vdash \Delta, A}$$

Structural rules (Cntd)

- Contraction (left):

$$\frac{\Gamma, A, A \Vdash \Delta}{\Gamma, A \Vdash \Delta}$$

- Contraction (right):

$$\frac{\Gamma \Vdash \Delta, A, A}{\Gamma \Vdash \Delta, A}$$

Rules for the constants

- ($\top$ left):

$$\frac{\Gamma \Vdash \Delta}{\Gamma, \top \Vdash \Delta}$$

- ($\bot$-left):

$$\frac{}{\Gamma, \bot \Vdash \Delta}$$

- ($\top$ right):

$$\frac{}{\Gamma \Vdash \Delta, \top}$$

- ($\bot$-right):

$$\frac{\Gamma \Vdash \Delta}{\Gamma \Vdash \Delta, \bot}$$

Rules for negation

- Negation (left):

$$\frac{\Gamma \Vdash \Delta, A}{\Gamma, \neg A \Vdash \Delta}$$

- Negation (right):

$$\frac{\Gamma, A \Vdash \Delta}{\Gamma \Vdash \Delta, \neg A}$$

Rules for Conjunction and Disjunction

- $(\wedge\text{left})$:

$$\frac{\Gamma, A, B \Vdash \Delta}{\Gamma, A \wedge B \Vdash \Delta}$$

- $(\vee\text{-left})$:

$$\frac{\Gamma, A \Vdash \Delta \quad \Gamma, B \Vdash \Delta}{\Gamma, A \vee B \Vdash \Delta}$$

- $(\wedge\text{right})$:

$$\frac{\Gamma \Vdash \Delta, A \quad \Gamma \Vdash \Delta, B}{\Gamma \Vdash \Delta, A \wedge B}$$

- $(\vee\text{-right})$:

$$\frac{\Gamma \Vdash \Delta, A, B}{\Gamma \Vdash \Delta, A \vee B}$$

Premises and conclusions are treated in the same way.

Rules for $\longrightarrow$ and $\longleftrightarrow$

- ($\longrightarrow$-left):

$$\frac{\Gamma \Vdash \Delta, A \quad \Gamma, B \Vdash \Delta}{\Gamma, A \longrightarrow B \Vdash \Delta}$$

- ($\longrightarrow$-right):

$$\frac{\Gamma, A \Vdash \Delta, B}{\Gamma \Vdash \Delta, A \longrightarrow B}$$

- ($\longleftrightarrow$-left):

$$\frac{\Gamma, A \Vdash B, A \quad B \longrightarrow A, \Vdash \Delta}{\Gamma, A \longleftrightarrow B \Vdash \Delta}$$

- ($\longleftrightarrow$-right):

$$\frac{\Gamma \Vdash \Delta, A \longrightarrow B \quad \Gamma \Vdash \Delta, B \longrightarrow A}{\Gamma \Vdash \Delta, A \longleftrightarrow B}$$

Rules for the quantifiers

- ($\forall$-left):

$$\frac{\Gamma, P[x := t] \Vdash \Delta}{\Gamma, \forall x : X P(x) \Vdash \Delta}$$

- ($\exists$-left):

$$\frac{\Gamma, P[x := y] \Vdash \Delta}{\Gamma, \exists x : X P(x) \Vdash \Delta}$$

- ($\forall$-right):

$$\frac{\Gamma \Vdash \Delta, P[x := y]}{\Gamma \Vdash \Delta, \forall x : X P(x)}$$

- ($\exists$-right):

$$\frac{\Gamma \Vdash \Delta, P[x := t]}{\Gamma \Vdash \Delta, \exists x : X P(x)}$$

The t is an arbitrary term of type X and X is not free in Γ, Δ

Rules for equality

- Reflection:

$$\frac{}{\Gamma \Vdash \Delta, t = t}$$

- Replication:

$$\frac{t_1 = t_2, \Gamma[t_2] \Vdash \Delta[t_2]}{\Gamma[t_1] \Vdash \Delta[t_1]}$$

Rules for IF

- PVS has an IF operator:
- The operator is defined as $(A \wedge B) \vee \neg A \wedge C$
- IF-left

$$\frac{\Gamma, A, B \Vdash \Delta \quad \Gamma, \neg A, C \Vdash \Delta}{\Gamma, IF(A, B, C) \Vdash \Delta}$$

- IF-right

$$\frac{\Gamma, A \Vdash \Delta, B \quad \Gamma, \neg A \Vdash \Delta, C}{\Gamma \Vdash \Delta IF(A, B, C)}$$