Fundamentos Lógicos y Algebraicos

The Language of First-Order Logic

Departamento de Sistemas Informáticos y Computación (DSIC)
Universidad Politécnica de Valencia (UPV)

Salvador Lucas
http://slucas.webs.upv.es/

Máster en Ingeniería y Tecnología de Sistemas Software (MITSS)

First-Order Logic (also called *Predicate Logic*), see, e.g., [Mendelson97], is an appropriate (simple, familiar) framework to

1. *specify* programs (syntax)
2. *describe* computations (semantics)
3. *specify* properties about programs, data, etc.
4. *reason* about computations and program properties (verification)

A *signature* is a set of *symbols* together with a mapping *ar* which indicates the number of *arguments* of each symbol (i.e., its *arity*)

We often write $h_{/k}$ to denote that *h is a k-ary symbol*, i.e.,$ar(h) = k$

The application of symbols to arguments can be *prefix* (by default), *infix* (e.g., $\_+\_$), *postfix* (e.g., $\_!$), or *mixfix* (e.g., *if_then_else_*)

Symbols of arith k are called k-ary; some of them have 'nicknames'

- Zero-argument symbols (e.g., 0, []) are called *constants*
- Symbols of arity 1 are called *monadic* (e.g., $isOdd_{/1}$, $qsort_{/1}$, $\_::Int$)
- Symbols of arity 2 are called *binary* (e.g., $max_{/2}$, $\_\geq\_$, $\_+\_$, $\_:\_$, $\_::\_$)
- Symbols of arity 3 are called *ternary* (e.g., *if_then_else_*)

There are signatures of:

- *Function symbols*, returning some object out from its arguments (e.g., 0, *qsort*, *max*). We often denote such signatures as $\mathcal{F}$
- *Predicate symbols*, representing properties that *hold* on the arguments (e.g., isOdd, _::Int, $\_\geq\_$). We often denote such signatures as Π

Let $\mathcal{X}$ be a countable set of *variables* $x, y, z, \ldots$, and $\mathcal{F}$ be a *signature of function symbols*.

Terms over $\mathcal{F}$ and $\mathcal{X}$ are defined by *induction*:

- (T-Base1) Variable symbols x are terms
- (T-Base2) Constant symbols a (i.e., $ar(a) = 0$) are terms
- (T-Induction) if f is a k-ary *function* symbol (i.e., $k = ar(f)$), $k > 0$, and $t_1, \ldots, t_k$ are *terms*, then $f(t_1, \ldots, t_k)$ is a term.

The set of *terms* is denoted as $\mathcal{T}(\mathcal{F}, \mathcal{X})$.

Example (Practical use of terms)

- Arithmetic expressions: $x + (y + z)$, $x \times y + x \times z$, $x + 0$.
- Data structures: a list *[2,0,1]* is represented as *2:0:1:[]*.
- Function calls: 2+1 represents a call to the *addition* operator.
- Assignments: $n := 3$ is a term: ':=' is a *binary* operator.
- Conditional statements: if n > 0 then n := n-1 else n := n+1 is a term: 'if_then_else_' is a ternary symbol, > is a *binary* operator, ...

Let $\mathcal{F}$ be a signature of *functions* and Π be a signature of *predicates*

If $P_{/n} \in \Pi$ and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ then $P(t_1, \ldots, t_n)$ is an atom

Example (Use in classical logic)

- isPhilosopher(socrates) encodes "*Socrates is a philosopher*"
- teacherOf(socrates,plato) encodes "*Socrates is the teacher of Plato*"

Example (Use in mathematics)

- $2 + 2 = 4$
- $x + 0 = x$

Example (Use in computer science)

- $\langle [(m, 0), (n, 1)] \mid m := 2 \rangle \to \langle [(m, 2), (n, 1)] \mid skip \rangle$
- $0 :: Int$

Example (Relational databases)

Consider the relational database using predicate *teach* [Reiter78, p. 59]:

teach	*Teacher*	*Student*
	alice	peter
	alice	quentin

It can be modeled as a set of atoms Teach:

$$\{teach(alice, peter), teach(alice, quentin)\}$$

Given signatures $\mathcal{F}$ and Π as above, a first-order *formula* F (often called *well-formed formula*, or *wff*) is built as follows:

- (F-Base1) Atoms are formulas.
- (F-Induction1) if F and F' are formulas, then $F \wedge F'$ and $\neg F$ (and also $F \vee F'$, $F \Rightarrow F'$,...) are formulas.
- (F-Induction2) if x is a variable and F is a formula, then $(\forall x)\ F$ and $(\exists x)\ F$ are formulas.

The set of first-order formulas is denoted as $Form(\mathcal{F}, \Pi, \mathcal{X})$

Example (Use in mathematics)

- $(\forall x)\ (x + 0 = x \wedge x \times 0 = 0)$

Example (Use in classical logic)

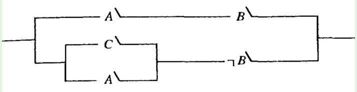
- $(\forall x)\ (\mathrm{isPhilosopher}(x) \Rightarrow \mathrm{isClever}(x))$, i.e., *all philosophers are clever*
- $(\forall x)(\forall y)\ (\mathrm{isPhilosopher}(x) \wedge \mathrm{teacherOf}(x, y) \Rightarrow \mathrm{isPhilosopher}(y))$, i.e., *somebody who is taught by a philosopher is a philosopher as well*

Example (Use in computer science)

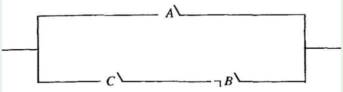
- $(\forall m)(\forall n)\,(m{::}Int \wedge n{::}Int \Rightarrow m + n{::}Int)$, i.e., *whenever m and n are of type Int, the expression $m+n$ is of type Int*

Example (Use in computer engineering (Shannon 1938 MsC Thesis))

Claude E. Shannon pioneered the use of logic in electric engineering, treating circuits as *boolean expressions*



$(A \wedge B) \vee ((C \vee A) \wedge \neg B)$



$A \vee (C \wedge \neg B)$

From [Mendelson97, Figures 1.2 & 1.3]

The rightmost (smaller/cheaper) circuit is *equivalent* to the leftmost one.

Such an equivalence can be *proved* to *simplify* the first into the second.

Definition (Sentence)

A *sentence* is a well-formed formula whose variables are *all quantified*.

Example (Sentence / No sentence)

- The formulas $0 = 1$ and $(\forall x)\; x + 0 = x$ are *sentences*.
- The formula $x \times 0 = 0$ is *not*.

Definition (Theory)

A *set of sentences* Th is a *theory*.

A *literal* L is either an *atom* A or the *negation* $\neg A$ of an atom A.

A *clause* C is a disjunction $L_1 \vee \cdots \vee L_n$ of literals, often represented as a *set* $C = \{L_1, \ldots, L_n\}$. Implicitly, variables are all *universally* quantified.

As we will see, *sentences* can be seen as *sets of clauses* $\{C_1, \ldots, C_m\}$, where, for all $1 \leq i \leq m$, C_i is a set of *literals*.

Example (sentences in clausal form)

- {{¬ isPhilosopher(x),isClever(x)}}
- {{¬isPhilosopher(x),¬teacherOf(x,y), isPhilosopher(y)}}
- $\{\{x + 0 \ = \ x\}, \{x \times 0 = 0\}\}$
- $\{\{\neg m :: Int, \neg n :: Int, m + n :: Int\}\}$

Innermost commas are *disjunction* $\vee$; *outermost* commas are *conjunction* $\wedge$

Chin-Liang Chang and Richard Char-Tung Lee.
Symbolic Logic and Mechanical Theorem Proving.
Academic Press, 1973.

Elliot Mendelson.
Introduction to Mathematical Logic. Fourth edition.
Chapman & Hall, 1997.

Teresa Hortalá González, Narciso Martí Oliet, Miguel Palomino Tarjuelo, Mario Rodríguez Artalejo y Rafael del Vado Vírseda.
Lógica Matemática para Informáticos.
Pearson, 2008.

Martin Davis and Hilary Putnam.
A Computing Procedure for Quantification Theory, *Journal of the ACM* 7:201-215, 1960.

Raymond Reiter.
On closed world data bases.
In Hervé Gallaire and Jack Minker, editors, *Logic and Data Bases, Symposium on Logic and Data Bases, Centre d'études et de recherches de Toulouse, France, 1977*, Advances in Data Base Theory, pages 55–76, New York, 1977. Plemum Press.