Mathematical Foundations of computer science

Lecture 6: First-order predicate logic

Speaker: Toru Takisaka

April 11, 2024

Recap

Our first goal: formalize a "mathematician"

- Analysis target of Mathematician $\approx$ Formulas
  - Mathematical assertions $\rightarrow$ Formulas
  - Truth of assertions $\rightarrow$ Valuation function
- Mathematician $\approx$ those who write proofs of formulas
  - Tools: axioms, assumptions, inference rules
  - Proof $\rightarrow$ seq. of formulas from axioms/assump. to target, connected by inference rules
- Properties of Mathematician
  - Soundness theorem: proved formulas are "true"
  - Completeness theorem: "true" formulas can be proved

Recap

There were two important notions, $\Sigma \models \varphi$ and $\Sigma \vdash \varphi$.

- $\Sigma \models \varphi$ (Σ logically implies φ) talks about the nature of the truth of formulas, which Mathematician wants to analyze. It is a semantic notion (it is about "the meaning" of formulas).
- $\Sigma \vdash \varphi$ (φ has a proof from Σ) talks about the possibility of inferring φ from Σ, which Mathematician attempts. Recall inference is a syntactic operation (i.e., inference rules can be described at the grammatical level).

Excercise: Prove $\varphi \models \varphi \vee \psi$ and $\varphi \vdash \varphi \vee \psi$, and observe how different their proofs are.

Recap

Soundness/completeness theorems say these notions coincide.

Theorem (Soundness of proof structure)

$\Sigma \vdash \varphi$ *implies* $\Sigma \models \varphi$.

The theorem roughly claims "correctness" of Mathematician: a proved formula under assumptions Σ is true whenever Σ is.

Theorem ((First) completeness of proof structure)

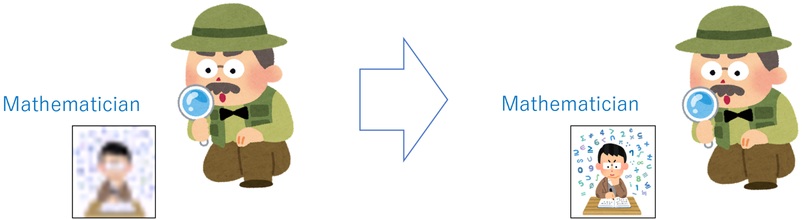
$\Sigma \models \varphi$ *implies* $\Sigma \vdash \varphi$.

The theorem roughly claims the "capability" of Mathematician: They can prove *any* φ from Σ whenever Σ logically implies φ.

Our next goal

Now we finished Part 1 of the course. What's next?

$\rightarrow$ formalize Mathematician with higher resolution...ours is too simple



Why too simple?

In Lecture 2, we said:

Formula = statement whose “correctness” can be argued

- “Dr. Takisaka is a professor”
- “Roses are blue”
- “1 + 1 = 3”

These formulas are *atomic*, i.e., they cannot be split into multiple formulas.

...but they are often too crude as the minimal parts of formulas.

In fact, they are made with subjects and predicates:

Why too simple?

In Lecture 2, we said:

Formula = statement whose "correctness" can be argued

- "Dr. Takisaka is a professor"
- "Roses are blue"
- "$1 + 1 = 3$"

These formulas are *atomic*, i.e., they cannot be split into multiple formulas.

...but they are often too crude as the minimal parts of formulas.

In fact, they are made with subjects and predicates:

The first step

As the first step, we “refine” the definition of formulas in the following way:

- Define *terms* (that formalize subjects),
- define *predicates*, and then
- define formulas by terms and predicates.

RECALL: when we define formulas, we do not consider “the meaning” of them yet; at first, they are just sequences of alphabets, and their truth etc. are given from outside.

Subjects = terms

Terms = objects whose properties are argued

- "n (which can be a number)", " φ (which can be a propositional formula)", ...
  $\rightarrow$ Variables... represent undetermined objects

- "Dr. Takisaka", "Roses", "0"...
  $\rightarrow$ Constant symbols... designate specific objects

- "$1+1$", "$x+y$", "The father of Dr. Takisaka",...
  $\rightarrow$ Terms made via function symbols

Arity = The number of inputs to a function symbol

- "$+$" is a *binary* (2-ary) function symbol
- "The father of" is a *unary* (1-ary) function symbol

Formal definition of terms

Fix a set F of function symbols. **Constant symbols are realized as 0-ary function symbols** (they accept the empty input).

Definition (terms)

Terms are defined by induction:

- Every variable x, y, z, etc., is a term.
- Every constant symbol c is a term.
- If f is a function symbol in F of arity k and t_1, ..., t_k are terms, then the expression $f(t_1, \ldots, t_k)$ is also a term.

For instance, imagine that F has a unary function symbol g and binary function symbol f. Then the following are terms:

$$x,\ y,\ z,\ g(y),\ f(x, y),\ f(z, z),\ g(x),\ f(g(x), f(z, z)),\ f(x, f(g(y), z)).$$

Unique readability of terms

Similar to formulas in propositional logic, it is important to check the unique readability of terms; it can be done in a similar way as formulas.

Predicates, languages

Predicates are just symbols with their arities.

- "is a professor" is a unary predicate
- "is blue" is a unary predicate
- "$=$" is a binary predicate

Sets of functions and predicate symbols constitute a *language* to describe formulas.

Definition (language)

A **language** L is the sequence

$$f_0^{n_0}, \ldots, f_t^{n_t}, P_0^{m_0}, \ldots, P_k^{m_k},$$

where $f_j^{n_j}$ is a **function symbol** of arity n_j , $j = 1, \ldots, t$,
and $P_i^{m_i}$ is a **predicate symbol** of arity m_i, $i = 1, \ldots, k$.

* function and predicate symbols are also called operation and relation symbols, respectively.

Definition of formulas

Definition (first-order predicate formula over language L)

Base case (defines atomic, or equivalently base, formulas):

- Given terms $t_1, \ldots, t_k$ and a predicate $P \in L$ of relation k, the expression $P(t_1, \ldots, t_k)$ is a formula.
- The expression $(t_1 = t_2)$ is a formula, t_1 and t_2 are terms.

Inductive step:

- If ϕ and ψ are formulas, then so are $(\phi \,\&\, \psi)$, $\neg\phi$, and $(\phi \ \vee \ \psi)$.
- If ϕ is a formula and x is a variable, then $\forall x \phi$ and $\exists x \phi$ are formulas.

As in the propositional logic case, $(\phi \to \psi)$ stands for $(\neg\phi \vee \psi)$.

Similar to propositional logic, one can check predicate formulas are uniquely readable.

Examples

- "Dr. Takisaka is a professor" could be formalized as a formula

$$\mathit{IsProf}(\text{Dr. Takisaka}),$$

where, *IsProf* is a unary predicate symbol and "Dr. Takisaka" is a constant symbol.

- "Any professor in China is over 20years old" could be formalized as a formula

$$\forall x. \Big((\mathit{IsProf}(x) \wedge \mathit{IsInChina}(x)) \to \mathit{Over}20(x) \Big),$$

where, *IsProf*, *IsInChina*, and *Over*20 are unary predicate symbols.

Examples

- "for any n, either n or $n+1$ is an odd number" could be formalized as a formula

$$\forall n. \Big(IsOdd(n) \vee IsOdd(S(n)) \Big),$$

where, S is a unary function symbol, and *IsOdd* is a unary predicate symbol.

  - S...*successor* function symbol, which represents "the next number"

- The *twin prime conjecture* could be formalized as a formula

$$\exists x. \forall y. \Big(prime(y) \wedge prime(y+2) \to y \leq x \Big),$$

where, *prime* is a unary predicate symbol, and $\leq$ is a binary predicate symbol.

We have defined *the syntax* of formulas, i.e., the grammatical rules how formulas are constructed (once again, without talking about its "meaning"). Now we will formalize "their meaning", i.e., *the semantics* of formulas.

In propositional logic, semantics is given by a truth assignment. In predicate logic, we need to specify:

- the domain, which specifies the objects we talks about;
- functions that specify the meaning of function symbols;
- predicates that specify the meaning of predicate symbols.

These components constitute an *(algebraic) structure*.

Domain

The *domain* A (a.k.a. the *universe*) of a structure is simply any nonempty set.

- When you claim "Any professor is over 20years old", you may be talking about
  - professors in UESTC ($A =$ people in UESTC)
  - professors in China ($A =$ people in China)
  - professors in the world ($A =$ people in the world)
- When you claim "for any n, either n or $n + 1$ is an odd number", you may be talking about
  - the property of natural numbers ($A = \mathbb{N}$)
  - the property of real numbers ($A = \mathbb{R}$)

Relations

Let A be a set (finite or infinite). Define

$$A^k = \{(a_1, \ldots, a_k) \mid a_1, \ldots, a_k \in A\}.$$

This is the set of all k-tuples on A.
For $k = 0$, we have $A^0 = \{()\}$.

If A has cardinality $n \geq 1$, then the cardinality of the set A^k is n^k.

A **relation** of arity k on A is any subset of A^k. On n element set A there are exactly 2^{n^k} relations of arity k.

Relations as Boolean valued functions

Relations R of arity k on A are k-variable Boolean-valued functions on variables $(x_1, \ldots, x_k)$. The variables range over A.

Namely, for all $x_1, \ldots, x_k \in A$, we have this:
$R(x_1, \ldots, x_k) =$ **true** if $(x_1, \ldots, x_k) \in R$, and
$R(x_1, \ldots, x_k) =$ **false** if $(x_1, \ldots, x_k) \notin R$.

Example: Edge relation E on graph $G = (V, E)$ defines the Boolean valued function:

$E(x, y) =$ **true** if $(x, y) \in E$,
$E(x, y) =$ **false** otherwise.

Equivalence relations

A binary relation R on A is a subset of A^2.

The relation R is an **equivalence relation** if for all $x, y, z \in A$:

- $(x, x) \in R$ (reflexivity).
- If $(x, y) \in R$ then $(y, x) \in R$ (symmetry).
- If $(x, y) \in R$ and $(y, z) \in R$ then $(x, z) \in R$ (transitivity).

For an $x \in A$, the **equivalence class of** x, written $[x]$, is:

$$[x] = \{y \in A \mid (x, y) \in R\}.$$

Binary relations: partial orders

The relation R on A is a **partial order** on A if for all $x, y, z \in A$:

- $(x, x) \in R$ (reflexivity).
- If $(x, y) \in R$ and $(y, x) \in R$ then $x = y$ (antisymmetry).
- If $(x, y) \in R$ and $(y, z) \in R$ then $(x, z) \in R$ (transitivity).

We often write $x \leq y$ to indicate $(x, y) \in R$.
So, we use the infix notation by replacing R with the symbol $\leq$.

Operations

A k-ary (or k-place) **operation** on a set A is a mapping

$$f : A^k \to A.$$

We postulate that the value $f(a_1, \ldots, a_k)$ is always defined. In this sense, all our operations are *total*.

When $k = 0$, the domain of $f : A^0 \to A$ is the set $\{()\}$. So, such a function is just an element in A that we call a *constant*.

Algebraic structures

Definition

An **algebraic structure (or simply structure)** $\mathcal{A}$ is a tuple:

$$(A; P_0^{m_0}, \ldots, P_k^{m_k}, f_0^{n_0}, \ldots, f_t^{n_t}),$$

where:

- A is a non-empty set called the **domain** of the structure,
- Each $P_i^{m_i}$, $i = 1, \ldots, k$, is a relation of arity m_i on A, and
- Each $f_j^{n_j}$, $j = 1, \ldots, t$, is an operation of arity n_j on A.

Examples of algebraic structures

1. Directed graphs $(V; E)$.
2. Simple graphs $(V; E)$.
3. $(\mathbb{N}; S, \leq)$, where $\mathbb{N} = \{0, 1, 2, \ldots\}$, $S(x) = x + 1$, and $\leq$ is the less or equal to relation on $\mathbb{N}$.
4. The Presburger arithmetic $(\mathbb{N}; S, +, 0, \leq)$.
5. The arithmetic $(\mathbb{N}; S, +, \times, 0, \leq)$.
6. Consider the structure $(P(X); \cup, \cap, \neg)$, where the domain $P(X)$ is the set of all subsets of X, $\cap$ is the intersection operation, $\neg$ is the complementation operation, and $\cup$ is the union operation. These are called *Boolean algebras*.