First Order Predicate Logic

Formal Semantics and Related Notions

Wolfgang Schreiner and Wolfgang Windsteiger
Wolfgang.(Schreiner|Windsteiger)@risc.jku.at

Research Institute for Symbolic Computation (RISC)
Johannes Kepler University (JKU), Linz, Austria
http://www.risc.jku.at

Formal Semantics

Up to now, our presentation of predicate logic formulas, their manipulation and proving, was mainly based on the form (syntax) of the formulas; this leaves many questions open.

- Equivalence of formulas:
  - What exactly does a formula *mean*, e.g., when do two syntactically different formulas express the same fact?
- Soundness and completeness of proving rules:
  - Proving rules allow by only considering the form of formulas to judge that some formula is a consequence of some other formulas.
  - But are the derived judgements really always true, i.e., are the rules really *sound*?
  - Furthermore, can all true judgements be derived, i.e., are the rules also *complete*?

We will answer these questions by underpinning our previous presentation with a formal definition of the meaning (semantics) of formulas.

Formal Semantics

The meaning of a predicate logic formula depends on the following entities.

- Domain D
  - A non-empty set, the universe about which the formula talks.

    $D = \mathbb{N}$.
- Interpretation I of all function and predicate symbols
  - Constants: For every constant c, $I(c)$ denotes an element of D, i.e., $I(c) \in D$.
  - Functions: For every function symbol f with arity $n > 0$, $I(f)$ denotes an n-ary function on D, i.e., $I(f) : D^n \to D$.
  - Predicates: For every predicate symbol p with arity $n > 0$, $I(p)$ denotes an n-ary predicate (relation) on D, i.e., $I(p) \subseteq D^n$.

$$I = [0 \mapsto \mathit{zero}, + \mapsto \mathit{add}, < \mapsto \mathit{less\text{-}than}, \ldots]$$

- Assignment $a : \mathit{Var} \to D$
  - A function that maps every variable x to a value $a(x)$ in this domain.

    $a = [x \mapsto 1, y \mapsto 0, z \mapsto 3, \ldots]$

The pair $M = (D, I)$ is also called a *structure*.

The Semantics of Terms

$$D, I, a \longrightarrow \boxed{[\![t]\!]} \longrightarrow d \in D$$

- Term semantics $[\![t]\!]^{D,I}_{a} \in D$
  - Given D, I, a, the semantics of term t is a value in D.
  - This value is defined by structural induction on t.

    $$t \ ::= \ x \mid c \mid f(t_1, \ldots, t_n)$$

- $[\![x]\!]^{D,I}_{a} := a(x)$
  - The semantics of a variable is the value given by the assignment.
- $[\![c]\!]^{D,I}_{a} := I(c)$
  - The semantics of a constant is the value given by the interpretation.
- $[\![f(t_1, \ldots, t_n)]\!]^{D,I}_{a} := I(f)([\![t_1]\!]^{D,I}_{a}, \ldots, [\![t_n]\!]^{D,I}_{a})$
  - The semantics of a function application is the result of the interpretation of the function symbol applied to the values of the argument terms.

The recursive definition of a function evaluating a term.

Example

$$D = \mathbb{N} = \{zero, one, two, three, \ldots\}$$
$$a = [x \mapsto one, y \mapsto two, \ldots]$$
$$I = [0 \mapsto zero, + \mapsto add, \ldots]$$

$$\begin{aligned}
[\![x + (y + 0)]\!]_a^{D,I} &= add([\![x]\!]_a^{D,I}, [\![y + 0]\!]_a^{D,I}) \\
&= add(a(x), [\![y + 0]\!]_a^{D,I}) \\
&= add(one, [\![y + 0]\!]_a^{D,I}) \\
&= add(one, add([\![y]\!]_a^{D,I}, [\![0]\!]_a^{D,I})) \\
&= add(one, add(a(y), I(0))) \\
&= add(one, add(two, zero)) \\
&= add(one, two) \\
&= three
\end{aligned}$$

The meaning of the term with the "usual" interpretation.

Example

$$D = \mathcal{P}(\mathbb{N}) = \{\emptyset, \{zero\}, \{one\}, \{two\}, \ldots, \{zero, one\}, \ldots\}$$
$$a = [x \mapsto \{one\}, y \mapsto \{two\}, \ldots]$$
$$I = [0 \mapsto \emptyset, + \mapsto union, \ldots]$$

$$\begin{aligned}
[\![x + (y + 0)]\!]_a^{D,I} &= union([\![x]\!]_a^{D,I}, [\![y + 0]\!]_a^{D,I}) \\
&= union(a(x), [\![y + 0]\!]_a^{D,I}) \\
&= union(\{one\}, [\![y + 0]\!]_a^{D,I}) \\
&= union(\{one\}, union([\![y]\!]_a^{D,I}, [\![0]\!]_a^{D,I})) \\
&= union(\{one\}, union(a(y), I(0))) \\
&= union(\{one\}, union(\{two\}, emptyset)) \\
&= union(\{one\}, \{two\}) \\
&= \{one, two\}
\end{aligned}$$

The meaning of the term with another interpretation.

The Semantics of Formulas

$$D, I, a \longrightarrow \boxed{[\![F]\!]} \longrightarrow \mathit{true}, \mathit{false}$$

- Formula semantics $[\![F]\!]^{D,I}_{a} \in \{\mathit{true}, \mathit{false}\}$
  - Given D, I, a, the semantics of term T is a truth value.
  - This value is defined by structural induction on F.

$$\begin{aligned} F \ :=\ & p(t_1, \ldots, t_n) \mid \top \mid \bot \\ & \mid \neg F \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid F_1 \rightarrow F_2 \mid F_1 \leftrightarrow F_2 \\ & \mid \forall x : F \mid \exists x : F \mid \ldots \end{aligned}$$

- $[\![p(t_1, \ldots, t_n)]\!]^{D,I}_{a} := I(p)([\![t_1]\!]^{D,I}_{a}, \ldots, [\![t_n]\!]^{D,I}_{a})$
  - The semantics of a atomic formula is the result of the interpretation of the predicate symbol applied to the values of the argument terms.
- $[\![\top]\!]^{D,I}_{a} := \mathit{true}, [\![\bot]\!]^{D,I}_{a} := \mathit{false}$

And now for the non-atomic formulas.

The Semantics of Propositional Formulas

- $[\![\neg F]\!]^{D,I}_a := \begin{cases} \mathit{true} & \text{if } [\![F]\!]^{D,I}_a = \mathit{false} \\ \mathit{false} & \text{else} \end{cases}$
- $[\![F_1 \wedge F_2]\!]^{D,I}_a := \begin{cases} \mathit{true} & \text{if } [\![F_1]\!]^{D,I}_a = [\![F_2]\!]^{D,I}_a = \mathit{true} \\ \mathit{false} & \text{else} \end{cases}$
- $[\![F_1 \vee F_2]\!]^{D,I}_a := \begin{cases} \mathit{false} & \text{if } [\![F_1]\!]^{D,I}_a = [\![F_2]\!]^{D,I}_a = \mathit{false} \\ \mathit{true} & \text{else} \end{cases}$
- $[\![F_1 \rightarrow F_2]\!]^{D,I}_a := \begin{cases} \mathit{false} & \text{if } [\![F_1]\!]^{D,I}_a = \mathit{true} \text{ and } [\![F_2]\!]^{D,I}_a = \mathit{false} \\ \mathit{true} & \text{else} \end{cases}$
- $[\![F_1 \leftrightarrow F_2]\!]^{D,I}_a := \begin{cases} \mathit{true} & \text{if } [\![F_1]\!]^{D,I}_a = [\![F_2]\!]^{D,I}_a \\ \mathit{false} & \text{else} \end{cases}$

The semantics coincides here with that of propositional logic.

The Semantics of Quantified Formulas

- $[\![\forall x : F]\!]^{D,I}_a := \begin{cases} \mathit{true} & \text{if } [\![F]\!]^{D,I}_{a[x \mapsto d]} = \mathit{true} \text{ for all } d \in D \\ \mathit{false} & \text{else} \end{cases}$
  - Formula is true, if body F is true for every value of the domain assigned to x.
- $[\![\exists x : F]\!]^{D,I}_a := \begin{cases} \mathit{true} & \text{if } [\![F]\!]^{D,I}_{a[x \mapsto d]} = \mathit{true} \text{ for some } d \in D \\ \mathit{false} & \text{else} \end{cases}$
  - Formula is true, if body F is true for at least one value of the domain assigned to x.

$$a[x \mapsto d](y) = \begin{cases} d & \mathit{if}\ x = y \\ a(y) & \mathit{else} \end{cases}$$

The core of the semantics.

Example

$$D = \mathbb{N}_3 = \{zero, one, two\}$$

$$a = [x \mapsto one, y \mapsto two, z \mapsto two, \ldots],\ I = [0 \mapsto zero, + \mapsto add, \ldots]$$

$$[\![\forall x : \exists y : x + y = z]\!]^{D,I}_{a} = true$$

- $[\![\exists y : x + y = z]\!]^{D,I}_{a[x \mapsto zero]} = true$
  - $[\![x + y = z]\!]^{D,I}_{a[x \mapsto zero, y \mapsto zero]} = false$
  - $[\![x + y = z]\!]^{D,I}_{a[x \mapsto zero, y \mapsto one]} = false$
  - $[\![x + y = z]\!]^{D,I}_{a[x \mapsto zero, y \mapsto two]} = \underline{true}$
- $[\![\exists y : x + y = z]\!]^{D,I}_{a[x \mapsto one]} = true$
  - $[\![x + y = z]\!]^{D,I}_{a[x \mapsto one, y \mapsto zero]} = false$
  - $[\![x + y = z]\!]^{D,I}_{a[x \mapsto one, y \mapsto one]} = \underline{true}$
  - $[\![x + y = z]\!]^{D,I}_{a[x \mapsto one, y \mapsto two]} = false$
- $[\![\exists y : x + y = z]\!]^{D,I}_{a[x \mapsto two]} = true$
  - $[\![x + y = z]\!]^{D,I}_{a[x \mapsto two, y \mapsto zero]} = \underline{true}$
  - $[\![x + y = z]\!]^{D,I}_{a[x \mapsto two, y \mapsto one]} = false$
  - $[\![x + y = z]\!]^{D,I}_{a[x \mapsto two, y \mapsto two]} = false$

The systematic investigation of respectively search for assignments.

Semantic Notions

Let F denote formulas, M structures, a assignments.

- F is satisfiable, if $[\![F]\!]^M_a = true$ for some M and a.

  $$p(0,x) \text{ is satisfiable; } q(x) \wedge \neg q(x) \text{ is not.}$$

- M is a model of F (short: $M \models F$), if $[\![F]\!]^M_a = true$ for all a.

  $$(\mathbb{N}, [0 \mapsto \mathit{zero}, p \mapsto \mathit{less\text{-}equal}]) \models p(0,x)$$

- F is valid (short: $\models F$), if $M \models F$ for all M.

  $$\models p(x) \wedge (p(x) \rightarrow q(x)) \rightarrow q(x)$$

  - F is satisfiable, if $\neg F$ is not valid.
  - F is valid, if $\neg F$ is not satisfiable.

- F is a logical consequence of formula set Γ (short: $\Gamma \models F$), if for all M and a, the following is true:

  If $[\![G]\!]^M_a = true$ *for every* G *in* Γ, *then also* $[\![F]\!]^M_a = true$.

  $$p(x), p(x) \rightarrow q(x) \models q(x)$$

- F_1 is a logical consequence of formula F_2, if $\{F_2\} \models F_1$.

Logical Equivalence

We are now going to address the first question stated in the beginning.

- Definition: two formulas F_1 and F_2 are logically equivalent (short: $F_1 \Leftrightarrow F_2$), if $F_1 \models F_2$ and $F_2 \models F_1$.
- Lemma: if $F \Leftrightarrow F'$ and $G \Leftrightarrow G'$, then

$$\neg F \Leftrightarrow \neg F'$$
$$F \wedge G \Leftrightarrow F' \wedge G'$$
$$F \vee G \Leftrightarrow F' \vee G'$$
$$F \rightarrow G \Leftrightarrow F' \rightarrow G'$$
$$F \leftrightarrow G \Leftrightarrow F' \leftrightarrow G'$$
$$\forall x : F \Leftrightarrow \forall x : F'$$
$$\exists x : F \Leftrightarrow \exists x : F'$$

Logically equivalent formulas can be substituted in any context without affecting the logical equivalence of the result (since $F \Leftrightarrow G$ iff $F \leftrightarrow G$ is valid, this justifies the proof rule A-$\leftrightarrow$).

Expressiveness of First-Order Logic

- Variables denote elements of the domain, thus no quantification is possible over functions and predicates of the domain.

  This would require second-order predicate logic.

- Nevertheless we express in first-order logic statements such as

  $\forall A, B, f \in A \to B : f$ is bijective $\to \exists g \in B \to A : \forall x \in B : f(g(x)) = x$

- This is possible because formulas are usually interpreted over the domain of sets, i.e., all variables denote sets:

$$
\begin{aligned}
A \to B := \{S \subseteq A \times B \mid\; & \\
& (\forall a \in A : \exists b \in B : (a, b) \in S) \wedge \\
& (\forall a, a', b : (a, b) \in S \wedge (a', b) \in S \to a = a')\}
\end{aligned}
$$

- Terms like $f(g(x))$ involve a hidden binary function "apply"

$$f(g(x)) \leadsto apply(f, apply(g, x))$$

  which denotes "function application":

$$apply(f, x) := \textbf{the } y : (x, y) \in f$$

First-order predicate logic over the domain of sets is the "working horse" of mathematics; virtually all of mathematics is formulated in this framework.

Soundness and Completeness of First-Order Logic

Now we turn our attention to the second question.

Completeness Theorem (Kurt Gödel, 1929): First order predicate logic has a proof calculus for which the following holds:

- Soundness: if by the rules of the calculus a conclusion F can be derived from a set of assumptions Γ ($\Gamma \vdash F$), then F is a logical consequence of Γ ($\Gamma \models F$).
- Completeness: if F is a logical consequence of Γ ($\Gamma \models F$), then by the rules of the calculus F can be derived from Γ ($\Gamma \vdash F$).

No logic that is stronger (more expressive) than first order predicate logic has a proof calculus that also enjoys both soundness and completeness.

Undecidability of First-Order Logic

The existence of a complete proof calculus does not mean that the truth of every formula is algorithmically decidable.

- Undecidability (Church/Turing, 1936/1937): there does not exist any algorithm that for given formula set Γ and formula F always terminates and says whether $\Gamma \models F$ holds or not.
- Semidecidability: but there exists an algorithm, that for given Γ and F, if $\Gamma \models F$, detects this fact in a finite amount of time.

  This algorithm searches for a proof of $\Gamma \vdash F$ in a complete proof calculus; if such a proof exists, it will eventually detect it; however, if no such proof exists, the search runs forever.

Automatic proof search is not able to detect that a formula is not true.

Limits of First-Order Logic

Not every structure can be completely described by a finite set of formulas.

- Incompleteness Theorem (Kurt Gödel, 1931): it is in no sound logic possible to prove all true arithmetic statements (i.e., all statements about natural numbers with addition and multiplication).
  - To adequately characterize $\mathbb{N}$, the (infinite) axiom scheme of mathematical induction has to be added.
- Corollary: in every sound formal system that is sufficiently rich there are statements that can neither be proved nor disproved.

In practice, complete reasoners for first-order logic are often supported by (complete or incomplete) reasoners for special theories.