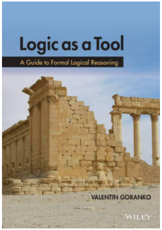
Logic as a Tool

Chapter 3: Understanding First-order Logic

3.2 Semantics of first-order logic

Valentin Goranko
Stockholm University



October 2016

Translation from first-order logic to natural language: examples in the structure of real numbers $\mathcal{R}$

$$\exists x(x < x \times y)$$

"*Some real number is less than its product with* y*.*"

$$\forall x(x < \mathbf{0} \rightarrow x^3 < \mathbf{0})$$

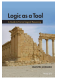
"*Every negative real number has a negative cube.*"

$$\forall x \forall y(xy > \mathbf{0} \rightarrow (x > \mathbf{0} \vee y > \mathbf{0})).$$

"*If the product of two real numbers is positive, then at least one of them is positive.*"

$$\forall x(x > \mathbf{0} \rightarrow \exists y(y^2 = x))$$

"*Every positive real number is a square of a real number.*"



Goranko

Translation from first-order logic to natural language: examples in the structure of humans $\mathcal{H}$

$$\mathbf{Elisabeth} = \mathsf{m}(\mathbf{Charles}) \to \exists x \mathsf{L}(x, \mathbf{Charles})$$

"*If Elisabeth is the mother of Charles then someone loves Charles.*"

$$\forall x(\exists y(y = \mathsf{m}(x)) \wedge \exists y(y = \mathsf{f}(x)))$$

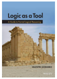
"*Everybody has a mother and a father.*"

$$\forall x \exists y L(x, y) \wedge \neg \exists x \forall y L(x, y)$$

"*Everyone loves someone and noone loves everyone.*"

$$\exists x \forall z(\neg \mathsf{L}(z, y) \to \mathsf{L}(x, z))$$

"*There is someone who loves everyone who does not love y.*"



Goranko

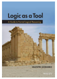
Semantics of first-order logic informally

The semantics of a first-order language $\mathcal{L}$ is a precise description of the meaning of terms and formulae in $\mathcal{L}$.

It is given by interpreting these into a given first-order structure $\mathcal{S}$ for which we want to use the language $\mathcal{L}$ to talk about.

Then, terms of formulae of $\mathcal{L}$ are translated into natural language expressions describing elements (for terms) or making statements (for formulae) in $\mathcal{S}$.

We will first discuss semantics of first-order languages informally.



Semantics of first-order languages formally: interpretations

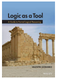
An interpretation of a first-order language $\mathcal{L}$ is any structure $\mathcal{S}$ for which $\mathcal{L}$ is a 'matching' language. For instance:

- the structure $\mathcal{N}$ is an interpretation of the language $\mathcal{L}_{\mathcal{N}}$.
  It is the intended, or standard interpretation of $\mathcal{L}_{\mathcal{N}}$.
- Likewise, the structure $\mathcal{H}$ is the standard interpretation of the language $\mathcal{L}_{\mathcal{H}}$.

There are many other, natural or 'unnatural' interpretations.

- For instance, we can interpret $\mathcal{L}_{\mathcal{N}}$ in other numerical structures extending $\mathcal{N}$, such as $\mathcal{Z}$, $\mathcal{Q}$, $\mathcal{R}$ by extending naturally the arithmetic predicates and operations.

- We can also interpret the non-logical symbols in $\mathcal{L}_{\mathcal{N}}$ arbitrarily in the set $\mathbb{N}$, or even in non-numerical domains, such as the set of humans $\mathbb{H}$.



Variable assignments and evaluations of terms

Given an interpretation $\mathcal{S}$ of a first-order language $\mathcal{L}$, a variable assignment in $\mathcal{S}$ is any mapping $v : VAR \to |\mathcal{S}|$ from the set of variables VAR to the domain of $\mathcal{S}$.

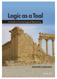
Due to the unique readability of terms, every variable assignment $v : VAR \to |\mathcal{S}|$ in a structure $\mathcal{S}$ can be uniquely extended to a mapping $v^{\mathcal{S}} : TM(\mathcal{L}) \to |\mathcal{S}|$, called term evaluation, such that for every n-tuple of terms $t_1, \ldots, t_n$ and an n-ary functional symbol f:

$$v^{\mathcal{S}}(f(t_1, \ldots, t_n)) = f^{\mathcal{S}}(v^{\mathcal{S}}(t_1), \ldots, v^{\mathcal{S}}(t_n))$$

where $f^{\mathcal{S}}$ is the interpretation of f in $\mathcal{S}$.

Intuitively, once a variable assignment v in the structure $\mathcal{S}$ is fixed, every term t in $TM(\mathcal{L})$ can be evaluated into an element of $\mathcal{S}$, which we denote by $v^{\mathcal{S}}(t)$ (or, just $v(t)$ when $\mathcal{S}$ is fixed) and call the value of the term t under the variable assignment v.

Important observation: *the value of a term only depends on the assignment of values to the variables occurring in that term.*



Goranko

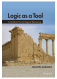
Evaluations of terms: examples

If v is a variable assignment in the structure $\mathcal{N}$ such that $v(x) = 3$ and $v(y) = 5$ then:

$$
\begin{aligned}
& v^{\mathcal{N}}(s(s(x) \times y)) \\
& = s^{\mathcal{N}}(v^{\mathcal{N}}(s(x) \times y)) \\
& = s^{\mathcal{N}}(v^{\mathcal{N}}(s(x)) \times^{\mathcal{N}} v^{\mathcal{N}}(y)) \\
& = s^{\mathcal{N}}(s^{\mathcal{N}}(v^{\mathcal{N}}(x)) \times^{\mathcal{N}} v^{\mathcal{N}}(y)) \\
& = s^{\mathcal{N}}(s^{\mathcal{N}}(3) \times^{\mathcal{N}} 5) \\
& = s^{\mathcal{N}}((3+1) \times^{\mathcal{N}} 5) \\
& = ((3+1) \times 5) + 1 \\
& = 21.
\end{aligned}
$$

Likewise, $v^{\mathcal{N}}(\mathbf{1} + (x \times s(s(\mathbf{2})))) = 13$.

If $v(x) =$'Mary'
then $v^{\mathcal{H}}(\mathbf{f}(\mathbf{m}(x))) =$ 'the father of the mother of Mary'.



Goranko

Truth of first-order formulae: the case of atomic formulae

We will define the notion of a formula A to be true in a structure $\mathcal{S}$ under a variable assignment v, denoted

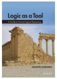
$$\mathcal{S}, v \models A,$$

compositionally on the structure of the formula A,
beginning with the case when A is an atomic formula.

Given an interpretation $\mathcal{S}$ of $\mathcal{L}$ and a variable assignment v in $\mathcal{S}$, we can compute the truth value of an atomic formula $p(t_1, \ldots, t_n)$ according to the interpretation of the predicate symbol $p^{\mathcal{S}}$ in $\mathcal{S}$, applied to the tuple of arguments $v^{\mathcal{S}}(t_1), \ldots, v^{\mathcal{S}}(t_n)$, i.e.

$\mathcal{S}, v \models p(t_1, \ldots, t_n)$ iff $p^{\mathcal{S}}$ holds (is true) for $v^{\mathcal{S}}(t_1), \ldots, v^{\mathcal{S}}(t_n)$.

Otherwise, we write $\mathcal{S}, v \not\models p(t_1, \ldots, t_n)$.



Goranko

Truth of atomic formulae: examples

If the binary predicate $\mathbf{L}$ is interpreted in $\mathcal{N}$ as $<$, and the variable assignment v is such that $v(x) = 3$ and $v(y) = 5$, we find that:

$\mathcal{N}, v \models \mathbf{L}(\mathbf{1} + (x \times s(s(\mathbf{2}))), s(s(x) \times y))$

iff $\mathbf{L}^{\mathcal{N}}((\mathbf{1} + (x \times s(s(\mathbf{2}))))^{\mathcal{N}}, (s(s(x) \times y))^{\mathcal{N}})$

iff $13 < 21$, which is true.

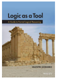
Likewise, $\mathcal{N}, v \models \mathbf{8} \times (x + s(s(y))) = (s(x) + y) \times (x + s(y))$

iff $(\mathbf{8} \times (x + s(s(y))))^{\mathcal{N}} = ((s(x) + y) \times (x + s(y)))^{\mathcal{N}}$

iff $80 = 81$, which is false.

Likewise, in $\mathcal{L}_{\mathcal{H}}$ with the standard interpretation:

- $x = \mathrm{m}(\mathbf{Mary})$ is true iff
  the value assigned to x is the mother of Mary.
- $\mathrm{L}(\mathrm{f}(\mathbf{John}), \mathrm{m}(\mathbf{Mary}))$ is true iff
  the father of John loves the mother of Mary.

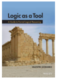


Goranko

Truth of first-order formulae
the propositional cases

The truth values propagate over the propositional connectives according to their truth tables, as in propositional logic:

- $\mathcal{S}, v \models \neg A$ iff $\mathcal{S}, v \not\models A$.
- $\mathcal{S}, v \models (A \wedge B)$ iff $\mathcal{S}, v \models A$ and $\mathcal{S}, v \models B$;
- $\mathcal{S}, v \models (A \vee B)$ iff $\mathcal{S}, v \models A$ or $\mathcal{S}, v \models B$;
- $\mathcal{S}, v \models (A \to B)$ iff $\mathcal{S}, v \not\models A$ or $\mathcal{S}, v \models B$;
- and likewise for $(A \leftrightarrow B)$.



Truth of first-order formulae: the quantifier cases

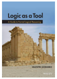
Notation: if x is a variable, v is a variable assignment in a structure $\mathcal{S}$, and $a \in \mathcal{S}$ then $v[x := a]$ is the assignment obtained from v by re-defining $v(x)$ to be a.

The truth of formulae $\forall x A(x)$ and $\exists x A(x)$ is computed according to the meaning of the quantifiers and the truth A:

$\mathcal{S}, v \models \exists x A(x)$ if $\mathcal{S}, v[x := a] \models A(x)$ *for some object* $a \in \mathcal{S}$.

Likewise,
$\mathcal{S}, v \models \forall x A(x)$ if $\mathcal{S}, v[x := a] \models A(x)$ *for every* object $a \in \mathcal{S}$.

If $\mathcal{S}, v \models A$ we also say that the formula A is satisfied by the assignment v in the structure $\mathcal{S}$.



Goranko

Computing the truth of first-order formulae

The truth of a formula in a given structure under given assignment only depends on the assignment of values to the variables occurring in that formula.

That is, if v_1, v_2 are variable assignments in $\mathcal{S}$ such that

$$v_1 \mid_{VAR(A)} = v_2 \mid_{VAR(A)}$$

where $VAR(A)$ is the set of variables in A, then

$$\mathcal{S}, v_1 \models A \text{ iff } \mathcal{S}, v_2 \models A.$$

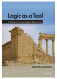
NB: the truth definitions of the quantifiers require taking into account possibly *infinitely many* variable assignments.

Truth of first-order formulae: examples

Consider the structure $\mathcal{N}$ and a variable assignment v such that $v(x) = 0$, $v(y) = 1$, $v(z) = 2$. Then:

- $\mathcal{N}, v \models \neg(x > y)$.
- However: $\mathcal{N}, v \models \exists x(x > y)$, since $\mathcal{N}, v[x := 2] \models x > y$.
- In fact, the above holds for any value assignment of y, and therefore $\mathcal{N}, v \models \forall y \exists x(x > y)$.
- On the other hand, $\mathcal{N}, v \models \exists x(x < y)$,
  but $\mathcal{N}, v \not\models \forall y \exists x(x < y)$. Why?
- What about $\mathcal{N}, v \models \exists x(x > y \wedge z > x)$? This is false.
- However, for the same variable assignment in the structure of rationals, $\mathcal{Q}, v \models \exists x(x > y \wedge z > x)$.

  Does this hold for every variable assignment in $\mathcal{Q}$?



Evaluation games

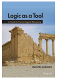
Two-player games, between Verifier and Falsifier.

The game is played in rounds, starting with an initial configuration:
$\langle$structure $\mathcal{S}$, variable assignment v, formula A $\rangle$

The objective of Verifier: to defend the claim that $\mathcal{S}, v \models A$,

The objective of Falsifier: to attack and refute that claim.

At each round, the current configuration $(\mathcal{S}, w, C)$ determines the player to move and the permissible moves, depending on the main connective of the formula C.



Evaluation games: the rules

- If the formula C is atomic, the game ends.

  If $\mathcal{S}, w \models C$ then Verifier wins, otherwise Falsifier wins.

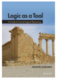
- If $C = \neg B$ then Verifier and Falsifier *swap their roles* and the game continues with the configuration $(\mathcal{S}, w, B)$.

  Swapping roles means: Verifier wins the game $(\mathcal{S}, w, \neg B)$ iff Falsifier wins the game $(\mathcal{S}, w, B)$; Falsifier wins the game $(\mathcal{S}, w, \neg B)$ iff Verifier wins the game $(\mathcal{S}, w, B)$.

  Intuition: verifying $\neg B$ is equivalent to falsifying B.

- If $C = C_1 \wedge C_2$ then Falsifier chooses $i \in \{1, 2\}$ and the game continues with the configuration $(\mathcal{S}, w, C_i)$.

  Intuition: for Verifier to defend the truth of $C_1 \wedge C_2$ he should be able to defend the truth of *any* of the two conjuncts, so, it is up to Falsifier to question the truth of either of them.



Goranko

- If $C = C_1 \vee C_2$ then Verifier chooses $i \in \{1, 2\}$ and the game continues with the configuration $(\mathcal{S}, w, C_i)$.

  Intuition: for Verifier to defend the truth of $C_1 \vee C_2$ it suffices to be able to defend the truth of at least one of the disjuncts.

- If $C = C_1 \to C_2$ then Verifier chooses $i \in \{1, 2\}$ and, depending on that choice, the game continues respectively with the configuration $(\mathcal{S}, w, \neg C_1)$ or $(\mathcal{S}, w, C_2)$.

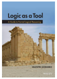
  Intuition: $C_1 \to C_2 \equiv \neg C_1 \vee C_2$.

- If $C = \exists x B$ then Verifier chooses an element $a \in \mathcal{S}$ and the game continues with the configuration $(\mathcal{S}, w[x := a], B)$.

  Intuition: verifying that $\mathcal{S}, w \models \exists x B$ amounts to verifying that $\mathcal{S}, w[x := a] \models B$ for some suitable element $a \in \mathcal{S}$.

- If $C = \forall x B$ then Falsifier chooses an element $a \in \mathcal{S}$ and the game continues with the configuration $(\mathcal{S}, w[x := a], B)$.

  Intuition: falsifying $\mathcal{S}, w \models \forall x B$ amounts to falsifying $\mathcal{S}, w[x := a] \models B$ for some suitable element $a \in \mathcal{S}$.



Evaluation games: winning strategies and truth of formulae

Any evaluation game always ends in a finite number of steps.

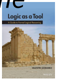
The game should be won by the player who has a winning strategy for it: a rule that, for every possible configuration from which that player is to move, assigns such a move, that he is guaranteed to eventually wins the game, no matter how the other player plays.

One of the players is sure to have a winning strategy.

Theorem

For every configuration $(\mathcal{S}, v, A)$*:*

1. $\mathcal{S}, v \models A$ *iff Verifier has a winning strategy for the evaluation game* $(\mathcal{S}, v, A)$*.*
2. $\mathcal{S}, v \not\models A$ *iff Falsifier has a winning strategy for the evaluation game* $(\mathcal{S}, v, A)$*.*



Evaluation games: Example 1

Consider the game $(\mathcal{N}, v, \forall y \exists x(x > y + z))$,

where v is such that $v(x) = 0$, $v(y) = 1$, $v(z) = 2$.

The first move of the game is by Falsifier.
He has to choose an integer n.

Then the game continues from configuration
$(\mathcal{N}, v[y := n], \exists x(x > y + z))$.

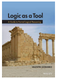
Now, Verifier has to choose an integer m so that to win the game
$(\mathcal{N}, v[y := n][x := m], (x > y + z))$. Suffices to choose $m > n + 2$.

Thus, he has a winning strategy for the game
$(\mathcal{N}, v[y := n], \exists x(x > y + z))$, for any $n \in \mathcal{N}$.

Hence, he has a winning strategy for the game $(\mathcal{N}, v, \forall y \exists x(x > y + z))$.

Therefore, $\mathcal{N}, v \models \forall y \exists x(x > y + z)$.

Such strategy for Verifier wins for *any* assignment of value to z,
thus showing that $\mathcal{N}, v \models \forall z \forall y \exists x(x > y + z)$.



Evaluation games: Example 2

Consider the game $(\mathcal{N}, v, \forall x(y < x \vee x < z))$.

The first move is by Falsifier, who has to choose an integer n.

Then the game continues from configuration
$(\mathcal{N}, v[x := n], (y < x \vee x < z))$,
in which Verifier must choose one of $y < x$ and $x < z$.

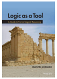
The Verifier has the following strategy:

- if Falsifier has chosen $n > 1$ then Verifier chooses the disjunct $y < x$ and wins the game $(\mathcal{N}, v[x := n], y < x)$;

- if Falsifier has chosen $n \leq 1$ then Verifier chooses the disjunct $x < z$ and wins the game $(\mathcal{N}, v[x := n], x < z)$.

Thus, Verifier has a winning strategy for the game $(\mathcal{N}, v, \forall x(y < x \vee x < z))$.

Therefore, $\mathcal{N}, v \models \forall x(y < x \vee x < z)$.



Goranko

Evaluation games: Example 3

Consider the game $(\mathcal{N}, v, \forall x(x < z \to \exists y(y < x)))$.

We claim that Falsifier has a winning strategy for that game.

The first move is by Falsifier. Let him choose 0.

Then the game continues from configuration
$(\mathcal{N}, v[x := 0], (x < z \to \exists y(y < x)))$.

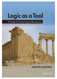
Now, Verifier is to choose a component of the implication.

- If Verifier chooses the antecedent, the game continues from configuration $(\mathcal{N}, v[x := 0], \neg(x < z))$ which is won by Falsifier, because $(\mathcal{N}, v[x := 0], x < z)$ is won by Verifier, since $0 < 2$.

- If Verifier chooses the consequent, the game continues from configuration $(\mathcal{N}, v[x := 0], \exists y(y < x))$. Now, Verifier is to choose a value for y. But, whatever $n \in \mathcal{N}$ Verifier chooses, he loses the game $(\mathcal{N}, v[x := 0][y := n], y < x)$ because $n < 0$ is false.

Thus, Verifier has no winning move. So, Falsifier is sure to win.

Therefore, $\mathcal{N}, v \not\models \forall x(x < z \to \exists y(y < x))$.



Goranko