Introduction to first-order logic:

First-order structures and languages.
Terms and formulae in first-order logic.
Interpretations, truth, validity, and satisfaction.

Valentin Goranko

DTU Informatics

September 2010

Propositional logic is too weak

Propositional logic only deals with fixed truth values.
It cannot capture the meaning and truth of statements like:

"$x + 2$ is greater than 5."

"There exists y such that $y^2 = 2$."

"For every real number x, if x is greater than 0, then there exists a real number y such that y is less than 0 and y^2 equals x."

"Everybody loves Raymond"

"Every man loves a woman"

First-order structures

A **first-order structure** consists of:

- A non-empty set, called a domain (of discourse) D;
- Distinguished predicates in D;
- Distinguished functions in D;
- Distinguished constants in D;

First-order structures: some examples

- $\mathcal{N}$: The set of natural numbers **N** with the unary *successor* function s, (where $s(x) = x + 1$), the binary functions $+$ (addition) and $\times$ (multiplication), the predicates $=$, $<$ and $>$, and the constant **0**.
- Likewise, but with the domains being the set of integers **Z**, rational numbers **Q**, or the reals **R** (possibly adding more functions) we obtain the structures $\mathcal{Z}$, $\mathcal{Q}$ and $\mathcal{R}$ respectively.
- $\mathcal{H}$: the domain is the set of all humans, with functions m ('*the mother of*'), f ('*the father of*'), the unary predicates M ('*man*'), W ('*woman*'), the binary predicates P ('*parent of*'), C ('*child of*'), L ('*loves*'), and constants (names), e.g. 'Adam', 'Eve', 'John', 'Mary' etc.
- $\mathcal{G}$: the domain is the set of all points and lines in the plane, with unary predicates **P** for '*point*', **L** for '*line*' and the binary predicate **I** for '*incidence*' between a point and a line.

Many-sorted first-order structures

Often the domain of discourse involves different sorts of objects, e.g., integers and reals; scalars and vectors; man and women; points, lines, triangles, circles; etc.

The notion of first-order structures can be extended naturally to many-sorted structures, with cross-sort functions and predicates.

Instead, we will use unary predicates to identify the different sorts within a universal domain.

First-order languages: vocabulary

1. Functional, predicate, and constant symbols, used as names for the distinguished functions, predicates and constants we consider in the structures.
   All these are referred to as non-logical symbols.
2. Individual variables: x, y, z, possibly with indices.
3. Logical symbols, including:
   3.1 the Propositional connectives: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ (or a sufficient subset of these);
   3.2 Equality $=$ (optional);
   3.3 Quantifiers:
   - ▷ the *universal quantifier* $\forall$ (*'all'*, *'for all'*, *'every'*, *'for every '*),
   - ▷ the *existential quantifier* $\exists$ (*'there exists'*, *'there is'*, *'some'*,*'for some'*, *'a'*).

   3.4 Auxiliary symbols, such as (,) etc.

First-order languages: terms

Inductive definition of the set of terms $TM(\mathcal{L})$ of a first-order language $\mathcal{L}$:

1. *Every constant symbol in $\mathcal{L}$ is a term.*
2. *Every individual variable in $\mathcal{L}$ is a term.*
3. *If $t_1, ..., t_n$ are terms and f is an n -ary functional symbol in $\mathcal{L}$, then $f(t_1, ..., t_n)$ is a term in $\mathcal{L}$.*

Construction/parsing tree of a term.

Examples of terms

1. In the language $\mathcal{L}_{\mathcal{N}}$: x, $s(x)$, $\mathbf{0}$, $s(\mathbf{0})$, $s(s(\mathbf{0}))$, etc.
   We denote the term $s(\ldots s(\mathbf{0})\ldots)$, where s occurs n times, by $\mathbf{n}$.

   More examples of terms in $\mathcal{L}_{\mathcal{N}}$:

   - $+(\mathbf{2}, \mathbf{2})$, which in a more familiar notation is written as $\mathbf{2} + \mathbf{2}$
   - $\mathbf{3} \times y$ (written in the usual notation)
   - $(x^2 + x) - \mathbf{5}$, where x^2 is an abbreviation of $x \times x$
   - $x_1 + s((y_2 + \mathbf{3}) \times s(z))$, etc.

2. In the 'human' language $\mathcal{L}_{\mathcal{H}}$:

   - x
   - **Mary**
   - m(**John**) ('*the mother of John*')
   - f(m(y)) ('*the father of the mother of x*'), etc.

First-order languages: atomic formulae

If $t_1, ..., t_n$ are terms in a language $\mathcal{L}$ and p is an n-ary predicate symbol in $\mathcal{L}$, then $p(t_1, ..., t_n)$ is an atomic formula in $\mathcal{L}$.

Examples:

1. In $\mathcal{L}_{\mathcal{N}}$:
   - $< (\mathbf{1}, \mathbf{2})$, or in traditional notation: $\mathbf{1} < \mathbf{2}$;
   - $x = \mathbf{2}$,
   - $\mathbf{5} < (x + \mathbf{4})$,
   - $\mathbf{2} + s(x_1) = s(s(x_2))$,
   - $(x^2 + x) - \mathbf{5} > 0$,
   - $x \times (y + z) = x \times y + x \times z$, etc.
2. In $\mathcal{L}_{\mathcal{H}}$:
   - $x = \mathrm{m}(\mathbf{Mary})$ ('x is the mother of Mary').
   - $\mathrm{L}(\mathrm{f}(y), y)$ ('The father of y loves y'), etc.

First-order languages: formulae

Inductive definition of the set of formulae $FOR(\mathcal{L})$:

1. *Every atomic formula in $\mathcal{L}$ is a formula in $\mathcal{L}$.*
2. *If A is a formula in $\mathcal{L}$ then $\neg A$ is a formula in $\mathcal{L}$.*
3. *If A, B are formulae in $\mathcal{L}$ then $(A \vee B), (A \wedge B), (A \rightarrow B), (A \leftrightarrow B)$ are formulae in $\mathcal{L}$.*
4. *If A is a formula in $\mathcal{L}$ and x is a variable, then $\forall x A$ and $\exists x A$ are formulae in $\mathcal{L}$.*

Construction/parsing tree of a formula, subformulae, main connectives: like in propositional logic.

Examples of formulae

1. In $\mathcal{L}_{\mathcal{Z}}$:
   - $(\mathbf{5} < x \wedge x^2 + x - \mathbf{2} = 0)$,
   - $\exists x(\mathbf{5} < x \wedge x^2 + x - \mathbf{2} = 0)$,
   - $\forall x(\mathbf{5} < x \wedge x^2 + x - \mathbf{2} = 0)$,
   - $(\exists y(x = y^2) \rightarrow (\neg x < 0))$,
   - $\forall x((\exists y(x = y^2) \rightarrow (\neg x < 0))$, etc.
2. In $\mathcal{L}_{\mathcal{H}}$:
   - $\mathbf{John} = \mathrm{f}(\mathbf{Mary}) \rightarrow \exists x \mathrm{L}(x, \mathbf{Mary})$;
   - $\exists x \forall z(\neg \mathrm{L}(z, y) \rightarrow \mathrm{L}(x, z))$,
   - $\forall y((x = \mathrm{m}(y)) \rightarrow (\mathrm{C}(y, x) \wedge \exists z \mathrm{L}(x, z)))$.

Some conventions

Priority order on the logical connectives:

- the unary connectives: negation and quantifiers have the strongest binding power, i.e. the highest priority,
- then come the conjunction and disjunction,
- then the implication, and
- the biconditional has the lowest priority.

Example:

$$\forall x(\exists y(x = y^2) \to (\neg(x < 0) \vee (x = \mathbf{0})))$$

can be simplified to

$$\forall x(\exists y\ x = y^2 \to \neg x < \mathbf{0} \vee x = \mathbf{0}).$$

On the other hand, for easier readability, extra parentheses can be optionally put around subformulae.

First-order instances of propositional formulae

Definition: Any uniform substitution of first-order formulae for the propositional variables in a propositional formula A produces a first-order formula, called a first-order instance of A.

Example:
Take the propositional formula

$$A = (p \wedge \neg q) \rightarrow (q \vee p).$$

The uniform substitution of $(\mathbf{5} < x)$ for p and $\exists y(x = y^2)$ for q in A results in the first-order instance

$$((\mathbf{5} < x) \wedge \neg \exists y(x = y^2)) \rightarrow (\exists y(x = y^2) \vee (\mathbf{5} < x)).$$

Unique readability of terms and formulae

Let $\mathcal{L}$ be an arbitrarily fixed first-order language.

Every occurrence of a functional symbol in a term from $TM(\mathcal{L})$ is the beginning of a unique subterm.

Therefore:

The set of terms $TM(\mathcal{L})$ has the unique readability property.

Every occurrence of a predicate symbol, $\neg, \exists$, or $\forall$ in a formula A from $FOR(\mathcal{L})$ is the beginning of a unique subformula of A.

Therefore:

The set of formulae $FOR(\mathcal{L})$ has the unique readability property.

Semantics of first-order logic informally

The semantics of a first-order language $\mathcal{L}$ is a precise description of the meaning of terms of 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, and later will define it formally.

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.*"

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

$$\textbf{Elisabeth} = \mathrm{m}(\textbf{Charles}) \rightarrow \exists x \mathrm{L}(x, \textbf{Charles})$$

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

$$\exists x \forall z (\neg \mathrm{L}(z, y) \rightarrow \mathrm{L}(x, z))$$

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

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

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

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

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

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

There is a real number greater than 2 and less than 3."

$$\exists x(x > \mathbf{2} \wedge x < \mathbf{3}).$$

There is an integer greater than 2 and less than 3."

$$\exists x(I(x) \wedge x > \mathbf{2} \wedge x < \mathbf{3}).$$

where $I(x)$ is interpreted as '*x is an integer.*

There is no real number the square of which equals -1."

It actually says "*It is not true that there is a real number the square of which equals* -1."

How about

$$\exists x(\neg x^2 = -1)?$$

No! The correct translation is

$$\neg\exists x(x^2 = -1).$$

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

Translate to first-order logic "*Every man loves a woman.*"

$$\forall x \exists y \mathsf{L}(x, y)?$$

No! This means 'Everybody loves somebody.'.

We must restrict the quantification of x to men, and of y respectively to women.

For that purpose we transform the sentence to:

"For every human, if he is a man, then there is a human who is a woman and the man loves that woman."

Now the translation into $\mathcal{L}_{\mathcal{H}}$ is immediate:

$$\forall x(\mathsf{M}(x) \to \exists y(\mathsf{W}(y) \wedge \mathsf{L}(x, y))).$$

Now, translate "*Every mother has a child whom she loves.*"

$$\forall x(\exists y(x = \mathsf{m}(y)) \to \exists z(\mathsf{C}(z, x) \wedge \mathsf{L}(x, z))).$$

Restricted quantification

To quantify only over those elements of the domain that satisfy a given (definable) property P, we use restricted quantification.

- For existential restricted quantification we use the template:

$$\exists x(P(x) \wedge \ldots)$$

- For universal restricted quantification we use the template:

$$\forall x(P(x) \rightarrow \ldots)$$

For instance:

$$\exists x(x > \mathbf{0} \wedge x^2 + x < \mathbf{5})$$

interpreted in $\mathcal{R}$, says that there exists a real number x which is positive and which satisfies $x^2 + x < 5$.

Likewise,

$$\forall x(x > \mathbf{0} \rightarrow x^2 + x < \mathbf{5})$$

interpreted in $\mathcal{R}$ says that all real numbers x which are positive satisfy $x^2 + x < 5$.

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.*

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'.

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

Truth of atomic formulae: examples

If the binary predicate **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.

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 $\mathcal{S}, v \models B$;
- $\mathcal{S}, v \models (A \rightarrow B)$ 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

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 *there exists an object* $a \in \mathcal{S}$ such that $\mathcal{S}, v[x := a] \models A(x)$, where the assignment $v[x := a]$ is obtained from v by re-defining $v(x)$ to be a.

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

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

Scope of a quantifier. Free and bound variables

Two different uses of variables in first-order formulae:

1. Free variables: used to denote *unknown or unspecified objects*, as in $(x > \mathbf{5}) \vee (x^2 + x - \mathbf{2} = \mathbf{0})$.
2. Bound variables: used *to quantify*, as in $\exists x(x^2 + x - \mathbf{2} = \mathbf{0})$ and $\forall x(x > \mathbf{5} \rightarrow x^2 + x - \mathbf{2} > \mathbf{0})$.

Scope of (an occurrence of) a quantifier in a formula A: the *unique* subformula QxB beginning with that occurrence of the quantifier.

An occurrence of a variable x in a formula A is bound if it is in the scope of some occurrence of a quantifier Qx in A. Otherwise, that occurrence of x is free. A variable is free (bound) in a formula, if it has a free (bound) occurrence in it. For instance, in the formula

$$A = (x > \mathbf{5}) \rightarrow \forall y(y < \mathbf{5} \rightarrow (y < x \wedge \exists x(x < \mathbf{3}))).$$

the first two occurrences of x are free, while all other occurrences of variables are bound. Thus, the only free variable in A is x, while both x and y are bound in A.

Truth of a formula does not depend on its bound variables

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

That is, if v_1, v_2 are variable assignments in $\mathcal{S}$ such that $v_1 \mid_{FV(A)} = v_2 \mid_{FV(A)}$, where $FV(A)$ is the set of free variables in A, then

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

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: $\mathcal{N}, v \models \exists x(x > y)$.
- 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 $\mathcal{Q}$?

Truth of sentences in structures. Models and countermodels.

Recall that a sentence is a formula with no free variables.

The truth of a sentence in a given structure does not depend on the variable assignment.

Therefore, for a structure $\mathcal{S}$ and sentence A we can simply write $\mathcal{S} \models A$ if $\mathcal{S}, v \models A$ for any/every variable assignment v.

We then say that $\mathcal{S}$ is a model of A and that A is true in $\mathcal{S}$, or that A is satisfied by $\mathcal{S}$.

Otherwise we write $\mathcal{S} \not\models A$ and say that $\mathcal{S}$ is a counter-model for A.

For instance: $\mathcal{N}$ is a model of the sentences
$\forall x \exists y (x < y)$ and $\forall x \forall y (x + y = y + x)$,
but is a counter-model of the sentence $\forall x \exists y (y < x)$.

Truth of first-order sentences: more examples

The sentence $\forall x(x = x)$ is true for any x in any domain of discourse, because of the meaning of the equality symbol $=$.

The sentence $\exists x(3x = 1)$ is true in the structure of rational numbers, but false in the structure of integers.

In the structure of real numbers $\mathcal{R}$:

- $\exists x(x = x^2)$ is true, take $x = 0$.
- $\forall x(x < \mathbf{0} \to x^3 < \mathbf{0})$ is true.
- $\forall x \forall y(xy > \mathbf{0} \to (x > \mathbf{0} \lor y > \mathbf{0}))$ is false: take e.g., $x = y = -1$.
- $\forall x(x > \mathbf{0} \to \exists y(y^2 = x))$ is true.
- $\exists x \forall y(xy < \mathbf{0} \to y = \mathbf{0})$ is true or false?