# First Order Predicate Logic

## Syntax and Informal Semantics

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

# Why Predicate Logic?

- Propositional logic is about "sentences" and their combination.

  *A "sentence" is a phrase that can be true or false.*

- Propositional logic cannot describe:
  1. "concrete objects" of a certain domain,
  2. functional relationships,
  3. statements about "for all" objects or about "for some" objects.
- Predicate logic is an extension of propositional logic, which (among other things!) allows to express these.

# Natural Language Formulations in Predicate Logic

- Alex is Tom's sister.

$$\text{sister}(\text{Alex}, \text{Tom})$$

- Tom has a sister in Linz.

$$\exists x : \text{sister}(x, \text{Tom}) \land \text{lives-in}(x, \text{Linz})$$

- Tom has two sisters.

$$\exists x, y : x \neq y \land \text{sister}(x, \text{Tom}) \land \text{sister}(y, \text{Tom})$$

- Tom has no brother.

$$\begin{array}{ll} \neg\exists x : \text{brother}(x, \text{Tom}) & \text{i.e. there does not exist a brother of Tom} \\ \forall x : \neg\text{brother}(x, \text{Tom}) & \text{i.e. everybody is not a brother of Tom} \end{array}$$

# Recall Syntax: Terms and Formulas

- In mathematics we want to speak about objects and their properties.
- The language of predicate logic provides terms and formulas, where
  - terms stand for objects (values) and
  - formulas stand for properties (which can be true or false).

$$
\begin{aligned}
\langle \mathit{expression} \rangle &::= \langle \mathit{term} \rangle \mid \langle \mathit{formula} \rangle \\
\langle \mathit{term} \rangle &::= \langle \mathit{variable} \rangle \mid \langle \mathit{constant} \rangle \mid \langle \mathit{fun\_sym} \rangle \; ( \; \langle \mathit{term} \rangle \;\; (, \langle \mathit{term} \rangle \,)* \;) \\
\langle \mathit{formula} \rangle &::= \top \mid \bot \mid \langle \mathit{atomic\_f} \rangle \mid \langle \mathit{connective\_f} \rangle \mid \langle \mathit{quantifier\_f} \rangle \\
\langle \mathit{atomic\_f} \rangle &::= \langle \mathit{pred\_sym} \rangle \; ( \; \langle \mathit{term} \rangle \;\; (, \langle \mathit{term} \rangle \,)* \;) \\
\langle \mathit{connective\_f} \rangle &::= \langle \mathit{conn1} \rangle \; \langle \mathit{formula} \rangle \mid \langle \mathit{formula} \rangle \; \langle \mathit{conn2} \rangle \; \langle \mathit{formula} \rangle \\
\langle \mathit{conn1} \rangle &::= \neg \\
\langle \mathit{conn2} \rangle &::= \wedge \mid \vee \mid \rightarrow \mid \leftrightarrow \\
\langle \mathit{quantifier\_f} \rangle &::= \langle \mathit{quantifier} \rangle \; \langle \mathit{variable} \rangle : \langle \mathit{formula} \rangle \\
\langle \mathit{quantifier} \rangle &::= \forall \mid \exists
\end{aligned}
$$

# Example

Tanja is female and every female is the daughter of her father.

$$isFemale(Tanja) \wedge \forall x : isFemale(x) \rightarrow isDaughter(x, fatherOf(x))$$

- "Names":
  - *Tanja* . . . constant
  - $x$ . . . variable
  - *isFemale* . . . predicate symbol
  - *fatherOf* . . . function symbol
- Terms:
  - $Tanja$, $x$, $fatherOf(x)$.
- Formulas:
  - $isFemale(Tanja)$
  - $isFemale(x)$
  - $isDaughter(x, fatherOf(x))$
  - $isFemale(x) \rightarrow isDaughter(x, fatherOf(x))$
  - $\forall x : isFemale(x) \rightarrow isDaughter(x, fatherOf(x))$
  - $isFemale(Tanja) \wedge \forall x : isFemale(x) \rightarrow isDaughter(x, fatherOf(x))$

Not always formulas are given in this "standard syntax".

# Abstract Syntax vs. Concrete Syntax

- Abstract syntax: one particular standard form to describe expressions.
- Concrete syntax: "concrete way" to write/display expressions.
  - Notation: just another word for concrete syntax.

Abstract syntax must allow unique identification of "type of the expression" and its "subexpressions".

One expression in abstract syntax can have many different forms in concrete syntax.

The language of mathematics is very rich in notations (e.g. subscripts, superscripts, writing things one above the other, etc.).

Well-chosen notation should convey intuitive meaning.

# Syntax: Notations and Conventions

- Function/Predicate symbols are often written using infix/prefix/postfix/matchfix operators:

$$a < b \rightsquigarrow\, <(a, b) \qquad \int f \rightsquigarrow \int(f)$$

$$\frac{a}{b} \rightsquigarrow /(a, b) \qquad ]a, b[ \rightsquigarrow \mathrm{openInterval}(a, b)$$

$$f' \rightsquigarrow \mathrm{derivative}(f) \qquad f \rightarrow a \rightsquigarrow \mathrm{converges}(f, a)$$

- Variable arity (overloading, no details):

$$a + b \rightsquigarrow +(a, b) \qquad a + b + c \rightsquigarrow \begin{cases} +(a, b, c) \\ +(+(a, b), c) \\ +(a, +(b, c)) \end{cases}$$

(beyond syntax!)

# Syntax: Examples

$a$ is less than $b$

- Abstract syntax: $<(a, b)$
- Notation: $a < b$

The open interval between $a$ and $b$

- Abstract syntax: $\mathrm{openInterval}(a, b)$
- Notation: $]a, b[$, $(a, b)$

The remainder of $a$ divided by $b$

- Abstract syntax: $\mathrm{remainder}(a, b)$
- Notation: $\mathrm{mod}(a, b)$, $a \bmod b$, $a \pmod b$, $a\%b$

$f$ converges to $a$

- Abstract syntax: $\mathrm{converges}(f, a)$
- Notation: $f \to a$, $\lim f = a$, $f(n) \xrightarrow{n\to\infty} a$, $\lim_{n\to\infty} f(n) = a$

# Syntax: Conditions in Quantifiers

- Problem: quantifier shall only range over a "subdomain" of values.
  - Values shall be "filtered" by a condition $C$.
- Solution:

$$\forall x : C \rightarrow F \qquad\qquad \exists x : C \wedge F$$

- Notation:

$$\forall C : F \qquad\qquad \exists C : F$$

  - The quantified variable in $C$ must be recognized from the context.

## Example

$$\forall x \in \mathbb{N} : x \geq 0 \qquad \exists x > 0 : x - 1 = 0 \qquad \forall x \in \mathbb{N} : \exists x < y : y < x + 2$$

# Syntax: Free and Bound Variables

- Every occurrence of $x$ in $\forall x : F$ is called bound (by the $\forall$-quantifier).
- Every occurrence of $x$ in $\exists x : F$ is called bound (by the $\exists$-quantifier).
- An occurence of a variable is called free if it is not bound.

## Example

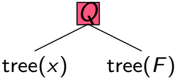
$$\begin{array}{rcl}
\text{converges}(f, a) \vee a = 0 & \rightsquigarrow & \text{no bound vars., } f, a \text{ free} \\
\forall f : \text{converges}(f, a) \wedge a = 0 & \rightsquigarrow & f \text{ is bound, } a \text{ is free} \\
\forall f : \forall a : \text{converges}(f, a) \leftrightarrow a = 0 & \rightsquigarrow & f, a \text{ are bound, no free vars.} \\
\forall f : \text{converges}(f, a) \rightarrow \exists a : a = 0 & \rightsquigarrow & f \text{ bound, } a \text{ free in one context} \\
 & & \text{and bound in another one.}
\end{array}$$

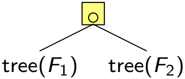
# Syntax Analysis

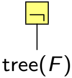
The construction of abstract syntax trees from their linear representation (proceeds easiest in a top-down fashion).

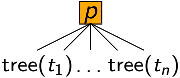
- Analyze quantified formulas (constructed from variables and other formulas by applications of quantifiers).
- Analyze propositional formulas (constructed from other formulas by applications of connectives).
- Analyze atomic formulas (constructed from terms by applications of predicate symbols).
- Analyze terms (variables or constants or constructed from other terms by applications of function symbols).
- This analysis determines the roles of names as variables, constants, function symbols, and predicate symbols.
  - Names like $x, y, z, \ldots$ are often used for variables.
  - Names like $a, b, c, \ldots$ are often used for constants.
  - Names like $f, g, h, \ldots$ are often used for function symbols.
  - Names like $p, q, r, \ldots$ are often used for predicate symbols.
- Determine the free variables of every formula.

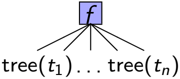
# Syntax Analysis

- For $Q \in \{\forall, \exists\}$: $\text{tree}(Qx : F) =$ 

- For $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$: $\text{tree}(F_1 \circ F_2) =$ 

- Formula $\neg F$: $\text{tree}(\neg F) =$ 

- Formula $p(t_1, \ldots, t_n)$: $\text{tree}(p(t_1, \ldots, t_n)) =$ 

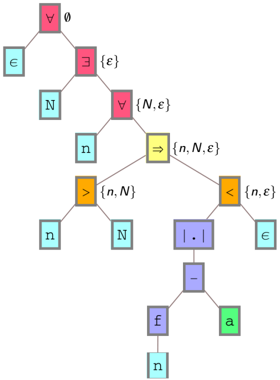
- Term $f(t_1, \ldots, t_n)$: $\text{tree}(f(t_1, \ldots, t_n)) =$ 

- Constant $c$: $\text{tree}(c) = \boxed{c}$
- Variable $x$: $\text{tree}(x) = \boxed{x}$

# Syntax Analysis

$$\forall \varepsilon : \exists N : \forall n : (n > N \rightarrow |f(n) - a| < \varepsilon)$$

- Quantifiers: $\varepsilon, N, n$ must be variables.
- Left and right of $\rightarrow$ must be formulas.
- $n > N$ must be an atomic formula (infix notation, predicate symbol "$>$" applied to variables $n$ and $N$.
- $|f(n) - a| < \varepsilon$: must be an atomic formula (infix notation, predicate symbol "$<$" applied to term $|f(n) - a|$ and variable $\varepsilon$).
- $|f(n) - a|$: function symbol "$|.|$" applied to $f(n) - a$.
- $f(n) - a$: function symbol "$-$" applied to $f(n)$ and $a$.
- $f(n)$: function symbol $f$ applied to variable $n$.

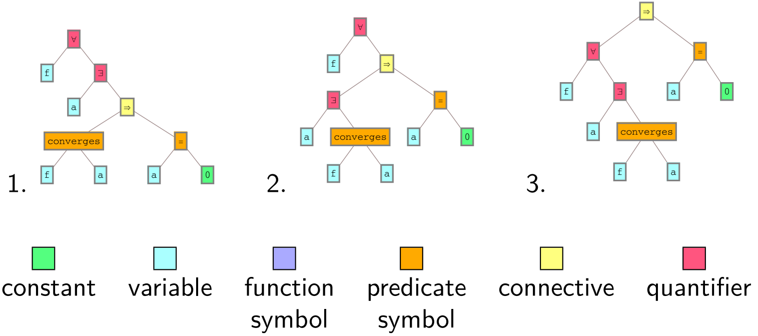


# Syntax Analysis

A syntax analysis can yield multiple results.

$$\forall f : \exists a : \text{converges}(f, a) \rightarrow a = 0$$

1. $\forall f : \exists a : (\text{converges}(f, a) \rightarrow a = 0)$
2. $\forall f : ((\exists a : \text{converges}(f, a)) \rightarrow a = 0)$
3. $(\forall f : \exists a : \text{converges}(f, a)) \rightarrow a = 0$




Typically syntax trees without free variables are intended (choice 1).

# Semantics

A predicate logic expression gets a meaning through a configuration, i.e. the specification of

1. a non-empty domain,
2. an interpretation that gives
   - for every constant an element of that domain,
   - for every function symbol with arity $n$ some concrete $n$-ary function on the domain, and
   - for every predicate symbol with arity $n$ some concrete $n$-ary relation on the domain, and
3. an assignment for the free variables in the expression.

# Semantics of Terms and Formulas

- Meaning of a term is an object in the domain.
  - Meaning of a variable is given by the assignment.
  - Meaning of a constant is given by the interpretation.
  - Meaning of $f(t_1, \ldots, t_n)$ is determined by applying the interpretation of $f$ to the meaning of the $t_i$.
- Meaning of a formula is true or false.
  - Meaning of $\top$ is true, meaning of $\bot$ is false.
  - Meaning of $t_1 = t_2$ is true, iff the meanings of $t_1$ and $t_2$ are identical.
  - Meaning of $p(t_1, \ldots, t_n)$ is determined by applying the interpretation of $p$ to the meaning of the $t_i$.
  - Meaning of logical connectives is determined by applying the truth tables to the meaning of the constituent subformulas.
  - Meaning of $\forall x : F$ is true iff the meaning of $F$ is true for all possible assignments for the free variable $x$.
  - Meaning of $\exists x : F$ is true iff the meaning of $F$ is true for at least one assignment for the free variable $x$.

# Semantics: Examples

- $\forall n : R(n,n)$
  - Domain: natural numbers.
  - $R$ is interpreted as the divisibility relation on natural numbers.
  - Every natural number is divisible by itself. $\rightsquigarrow$ true
- $\forall n : R(n,n)$
  - Domain: real numbers.
  - $R$ is interpreted as the less-than relation on real numbers.
  - Every real number is less than itself. $\rightsquigarrow$ false
- $\exists x : R(a,x) \wedge R(x,b)$
  - Domain: real numbers.
  - $R$ is interpreted as the less-than relation on real numbers.
  - There is a real number $x$ such that $a < x$ and $x < b$. $\rightsquigarrow$ ???
  - Assignment $[a \mapsto 5, b \mapsto 6]$: There is an assignment for $x$ such that $5 < x$ and $x < 6$. $\rightsquigarrow$ true, e.g. $[x \mapsto 5.5]$
  - Assignment $[a \mapsto 7, b \mapsto 6]$: There is an assignment for $x$ such that $7 < x$ and $x < 6$. $\rightsquigarrow$ false, why?

# Nested Quantifiers

When quantifiers of different type are nested, the order matters.

## Example

Domain: natural numbers.

$$\forall x : \exists y : x < y \quad \rightsquigarrow \quad \textit{true}$$

(Why? For the assignment $[x \mapsto \bar{x}]$ for $x$ take $[y \mapsto \bar{x} + 1]$ as the assignment for $y$. The meaning of $x < y$ is then $\bar{x} < \bar{x} + 1$, which is true no matter what $\bar{x}$ is.)

$$\exists y : \forall x : x < y \quad \rightsquigarrow \quad \textit{false}$$

(Why? Assume it was true, i.e. there is an assignment $[y \mapsto \bar{y}]$ for $y$ such that $x < y$ is true for all assignments for $x$. But take $[x \mapsto \bar{y}]$ as the assignment for $x$. The meaning of $x < y$ is then $\bar{y} < \bar{y}$, which is false, hence the original assumption must not be made, thus the meaning of the formula must be false.)

# Semantics Convention

- The meaning of "$=$", logical connectives, and quantifiers is defined by above rules.
- The meaning of all other symbols depends on the interpretation which can be chosen as desired and must be given explicitly.
  - It is in principle possible to express "$a$ divides the sum of $b$ and $c$" by

    $$a \subseteq (b * c)$$

    using the interpretation

    $$[\subseteq \mapsto \text{ the divisibility relation}, \ * \mapsto \text{ the addition function}].$$
  - Convention: if the interpretation is not given explicitly, then a "standard interpretation" is assumed.

# Semantics: Consequence and Equivalence

- $F$ is a (logical) consequence of $\Gamma$ iff
  $F$ is true in every configuration, in which all $G \in \Gamma$ are true.
  - $F_2$ is a logical consequence of $F_1$ means $F_2$ is a consequence of $\{F_1\}$.
  - $F_2$ "follows from" $F_1$ regardless of the configuration. $F_1$ "implies" $F_2$.
- $F_1$ is (logically) equivalent to $F_2$ (write "$F_1 \Leftrightarrow F_2$") iff
  $F_1$ is a consequence of $F_2$ and $F_2$ is a consequence of $F_1$.
  - $F_1$ and $F_2$ have the same meaning, regardless of the configuration.
  - Every formula can always be substituted by an equivalent one.
- $F$ is valid iff $F$ is true in every configuration.
  - $F$ is a "fact", $F$ is a logical consequence of $\emptyset$.
  - ($F_1 \Leftrightarrow F_2$) iff ($F_1 \leftrightarrow F_2$ is valid).
  - $F_2$ is a logical consequence of $F_1$ iff ($F_1 \rightarrow F_2$ is valid).

# Equivalent Formulas

In addition to equivalences for connectives (see propositional logic):

$$
\begin{array}{lcl}
\neg(\forall x : F) & \Leftrightarrow & \exists x : \neg F \quad \text{(De-Morgan)} \\
\neg(\exists x : F) & \Leftrightarrow & \forall x : \neg F \quad \text{(De-Morgan)} \\
\forall x : (F_1 \wedge F_2) & \Leftrightarrow & (\forall x : F_1) \wedge (\forall x : F_2) \\
\exists x : (F_1 \vee F_2) & \Leftrightarrow & (\exists x : F_1) \vee (\exists x : F_2) \\
\forall x : (F_1 \vee F_2) & \Leftrightarrow & F_1 \vee (\forall x : F_2), \text{ if } x \text{ does not occur free in } F_1 \\
\exists x : (F_1 \wedge F_2) & \Leftrightarrow & F_1 \wedge (\exists x : F_2), \text{ if } x \text{ does not occur free in } F_1
\end{array}
$$

For a finite domain $\{v_1, \ldots, v_n\}$:

$$
\begin{array}{lcl}
\forall x : F & \Leftrightarrow & F[v_1/x] \wedge \ldots \wedge F[v_n/x] \\
\exists x : F & \Leftrightarrow & F[v_1/x] \vee \ldots \vee F[v_n/x]
\end{array}
$$

$E[t/x]$: the expression $E$ with every free occurrence of $x$ substituted by the term $t$. ($\rightsquigarrow$ $E$ has the same meaning for $x$ as $E[t/x]$ has for $t$.)

# Language Extensions

- Locally bound variables: **let** $x = t$ **in** $E$
  - $E$ can be a term or a formula, **let** ... **in** ... is term or a formula, respectively.
  - Binds the variable $x$.
  - Meaning: $E[t/x]$.
  - Alternative notation: $E$ **where** $x = t$ or $E|_{x=t}$.
  - If $F$ is a formula, then

$$\textbf{let } x = t \textbf{ in } F \Leftrightarrow \exists x : x = t \wedge F.$$

- Conditional: **if** $C$ **then** $E_1$ **else** $E_2$
  - $E_i$ can be both terms or both formulas, **if** $C$ **then** $E_1$ **else** $E_2$ is term or a formula, respectively.
  - Meaning: if $C$ means true, then the meaning of $E_1$, otherwise the meaning of $E_2$.
  - If $E_1$ and $E_2$ are formulas, then

$$\textbf{if } C \textbf{ then } E_1 \textbf{ else } E_2 \Leftrightarrow (C \rightarrow E_1) \wedge (\neg C \rightarrow E_2).$$

# Further Quantifiers

Common mathematical language uses more quantifiers:

- $\sum_{i=l}^{h} t$: binds $i$. Meaning: $t[l/i] + \cdots + t[h/i]$.
- $\prod_{i=l}^{h} t$: binds $i$. Meaning: $t[l/i] \cdots t[h/i]$.
- $\{x \in A \mid P\}$: binds $x$. Meaning: The set of all $x$ in $A$ such that $P$ is true.
- $\{t \mid x \in A \wedge P\}$: binds $x$. Meaning: The set of all $t$ when $x$ is in $A$ and $P$ is true.
- $\lim_{x \to v} t$: binds $x$. Meaning: The limit of $t$ when $x$ goes to $v$.
- $\max_{x \in A} t$: binds $x$. Meaning: The maximum of $t$ when $x$ runs through $A$.
- $\min_{x \in A} t$: binds $x$. Meaning: The minimum of $t$ when $x$ runs through $A$.
- ...