

First Order Predicate Logic

Syntax

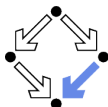
Wolfgang Schreiner and Wolfgang Windsteiger

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

Research Institute for Symbolic Computation (RISC)

Johannes Kepler University (JKU), Linz, Austria

<http://www.risc.jku.at>



Why not only Propositional Logic?

- ▶ Propositional logic is about “sentences” and their combination.

A (logic) formula F is a “sentence” that can be true or false.

$$F ::= \top \mid \perp \mid p \\ \mid (\neg F) \mid (F_1 \wedge F_2) \mid (F_1 \vee F_2) \mid (F_1 \rightarrow F_2) \mid (F_1 \leftrightarrow F_2)$$

- ▶ *Atomic* formulas \top , \perp , p with fixed/given truth values.
- ▶ *Compound* formulas constructed from (logical) connectives ($\neg, \wedge, \vee, \rightarrow, \leftrightarrow$); their truth values are computed from the truth values of their subformulas according to the “truth table” of the connective.
- ▶ 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.



The Syntax of Predicate Logic: Terms and Formulas

We want to speak about **objects** and their **properties**.

- ▶ For this predicate logic provides two kinds of “expressions”:
 - ▶ **Terms** denoting **objects** (values).
 - ▶ **Formulas** denoting **properties** of objects (truth values).

$$t ::= \underline{v} \mid \underline{c} \mid \underline{f(t_1, \dots, t_n)}$$

$$F ::= \top \mid \perp \mid \underline{p(t_1, \dots, t_n)}$$

$$\mid (\neg F) \mid (F_1 \wedge F_2) \mid (F_1 \vee F_2) \mid (F_1 \rightarrow F_2) \mid (F_1 \leftrightarrow F_2)$$

$$\mid \underline{(\forall v : F)} \mid \underline{(\exists v : F)}$$

- ▶ v : a *variable* denoting a varying object.
- ▶ c : a *constant* denoting a fixed object.
- ▶ f : a *function symbol* denoting a function that maps the objects denoted by terms t_1, \dots, t_n to another object.
- ▶ p : a *predicate symbol* denoting a predicate that maps the objects denoted by terms t_1, \dots, t_n to a truth value.
- ▶ \forall, \exists : a *quantifier* “binding” a variable v within a formula F .
 - ▶ $\forall v : F$: “for all (values of) v , F is true”.
 - ▶ $\exists v : F$: “there exists some (value of) v for which F is true”.



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



Formulas and Parentheses

We use the following convention to reduce the number of parentheses:

- ▶ Decreasing binding power:
 - ▶ \neg binds stronger than \wedge .
 - ▶ \wedge binds stronger than \vee .
 - ▶ \vee binds stronger than \rightarrow .
 - ▶ \rightarrow binds stronger than \leftrightarrow .
 - ▶ \leftrightarrow binds stronger than \forall, \exists .
- ▶ Consequence for quantified formula $\forall v : F$ and $\exists v : F$:
 - ▶ Without parentheses, the scope of a quantified formula reaches to the end of the enclosing formula.

Example

The previous formula can be also written as

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

If in doubt, use parentheses (respectively ask!).



Predicate Logic and Natural Language

- ▶ Alex is Tom's sister.

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

- ▶ Tom has a sister in Linz.

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

- ▶ Tom has two sisters.

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

- ▶ Tom has no brother.

$\neg \exists x : \text{brother}(x, \text{Tom})$ i.e. there does not exist a brother of Tom

$\forall x : \neg \text{brother}(x, \text{Tom})$ i.e. everybody is not a brother of Tom

Many natural language statements can be expressed in predicate logic.



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 occurrence of a variable is called **free** if it is not bound.
- ▶ A formula with no free variables is called **closed**.

Example

$\text{converges}(u, x) \vee x = 0 \rightsquigarrow$ no bound vars., u, x free

$\forall x : \text{converges}(u, x) \wedge x = 0 \rightsquigarrow$ x is bound, u is free

$\forall u : \forall x : \text{converges}(u, x) \leftrightarrow x = 0 \rightsquigarrow$ u, x are bound, formula closed

$\forall u : \text{converges}(u, x) \rightarrow \exists x : x = 0 \rightsquigarrow$ u bound, x free in one context
and bound in another one.

As we will see later, if a formula is not closed, its truth value depends on the values given to its free variables.



The Free Variables of a Formula

$fv(t)$ and $fv(F)$ compute the set of free vars of term t and formula F .

$$fv(\top) = \emptyset$$

$$fv(\perp) = \emptyset$$

$$fv(p(t_1, \dots, t_n)) = fv(t_1) \cup \dots \cup fv(t_n)$$

$$fv(\neg F) = fv(F)$$

$$fv(F_1 \wedge F_2) = fv(F_1) \cup fv(F_2)$$

$$fv(F_1 \vee F_2) = fv(F_1) \cup fv(F_2)$$

$$fv(F_1 \rightarrow F_2) = fv(F_1) \cup fv(F_2)$$

$$fv(F_1 \leftrightarrow F_2) = fv(F_1) \cup fv(F_2)$$

$$fv(\forall v : F) = \underline{fv(F) \setminus \{v\}}$$

$$fv(\exists v : F) = \underline{fv(F) \setminus \{v\}}$$

$$fv(v) = \{v\}$$

$$fv(c) = \emptyset$$

$$fv(f(t_1, \dots, t_n)) = fv(t_1) \cup \dots \cup fv(t_n)$$

Example

$$\begin{aligned} fv(\forall x : p(x) \rightarrow \exists y : q(x, y)) &= fv(p(x) \rightarrow \exists y : q(x, y)) \setminus \{x\} \\ &= (fv(p(x)) \cup fv(\exists y : q(x, y))) \setminus \{x\} \\ &= (\{x\} \cup (fv(q(x, y)) \setminus \{y\})) \setminus \{x\} \\ &= (\{x\} \cup (\{x, y\} \setminus \{y\})) \setminus \{x\} \\ &= (\{x\} \cup \{x\}) \setminus \{x\} = \{x\} \setminus \{x\} = \emptyset \end{aligned}$$

Quantifiers remove free variables from a formula.



Abstract Syntax vs. Concrete Syntax

Terms and formulas are not always given in the syntax presented so far.

- ▶ **Abstract syntax:** one particular “standard form” of expressions.
 - ▶ Up to now we presented this standard form.
 - ▶ Abstract syntax must allow unique identification of “type of the expression” and its “subexpressions”.
- ▶ **Concrete syntax:** any concrete “notation” to write expressions.
 - ▶ 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.

For understanding their meaning, it is important to translate expressions from concrete syntax to their standard form.



Syntax: Notations and Conventions

- ▶ Function/predicate symbols are often written using **infix/prefix/postfix/matchfix operators**:

$$\begin{array}{ll} a < b \rightsquigarrow <(a, b) & \int f \rightsquigarrow \int(f) \\ \frac{a}{b} \rightsquigarrow /(a, b) &]a, b[\rightsquigarrow \text{openInterval}(a, b) \\ f' \rightsquigarrow \text{derivative}(f) & f \rightarrow a \rightsquigarrow \text{converges}(f, a) \end{array}$$

- ▶ **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: $\text{openInterval}(a, b)$
- ▶ Notation: $]a, b[, (a, b)$

The remainder of a divided by b

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

f converges to a

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



Syntax: Conditions in Quantifiers

- ▶ **Problem:** quantifier shall only range over a “subdomain” of values.
 - ▶ Only values satisfying a “filter condition” C shall be considered.

- ▶ **Solution:**

$$\forall x : C \rightarrow F$$

$$\exists x : C \wedge F$$

- ▶ **Notation:**

$$\forall C : F$$

$$\exists C : F$$

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

Example

- ▶ “All natural numbers are greater equal zero.”

$$\forall x \in \mathbb{N} : x \geq 0$$

$$\forall x : x \in \mathbb{N} \rightarrow x \geq 0$$

- ▶ “There exists some natural number whose predecessor is zero.”

$$\exists x \in \mathbb{N} : x - 1 = 0$$

$$\exists x : x \in \mathbb{N} \wedge x - 1 = 0$$

- ▶ “For every $x \in \mathbb{N}$ there exists a number between x and $x + 2$.”

$$\forall x \in \mathbb{N} : \exists x < y : y < x + 2$$

Standard form?



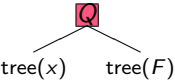
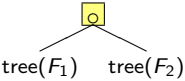

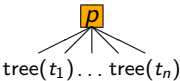
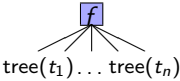


Syntax Analysis

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, \dots are often used for variables.
 - ▶ Names like a, b, c, \dots are often used for constants.
 - ▶ Names like f, g, h, \dots are often used for function symbols.
 - ▶ Names like p, q, r, \dots are often used for predicate symbols.
- ▶ Determine the **free variables** of every formula.



Syntax Analysis

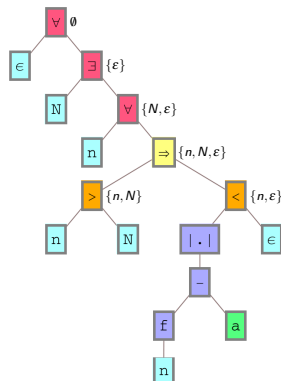
- ▶ For $Q \in \{\forall, \exists\}$: $\text{tree}(Qx : F) =$ 
- ▶ For $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$: $\text{tree}(F_1 \circ F_2) =$ 
- ▶ Formula $\neg F$: $\text{tree}(\neg F) =$ 
- ▶ Formula $p(t_1, \dots, t_n)$: $\text{tree}(p(t_1, \dots, t_n)) =$ 
- ▶ Term $f(t_1, \dots, t_n)$: $\text{tree}(f(t_1, \dots, t_n)) =$ 
- ▶ Constant c : $\text{tree}(c) =$ 
- ▶ Variable x : $\text{tree}(x) =$ 



Syntax Analysis

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

- ▶ Quantifiers: ε, 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 ε).
- ▶ $|f(n) - a|$: **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 .

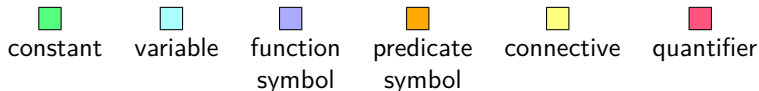
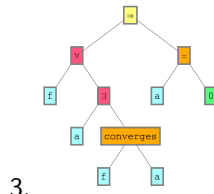
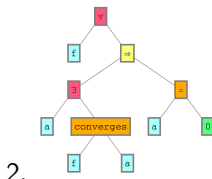
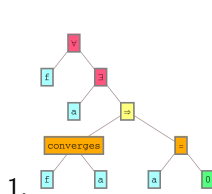


Syntax Analysis

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).



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

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

- ▶ 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

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



Further Quantifiers

Common mathematical language uses more quantifiers:

- ▶ $\sum_{i=l}^h t$: binds i . Meaning: $t[l/i] + \dots + t[h/i]$.
- ▶ $\prod_{i=l}^h t$: 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.
- ▶ $\lim_{x \rightarrow v} t$: 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 .
- ▶ ...

