FIRST-ORDER 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?

- A propositional formula F describes a "sentence" that can be "true" or "false":

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

  - Propositional variables $p \in \mathcal{P}$ with given truth values.
  - Propositional constants $\top$ and $\bot$ with fixed truth values.
  - Compound formulas constructed from the (logical) connectives $(\neg, \wedge, \vee, \rightarrow, \leftrightarrow)$ whose truth values are determined by corresponding truth tables.

Propositional logic is about the combination of truth values.

Why Not Only Propositional Logic?

For all numbers x and y it is the case that, if x is greater equal zero and y is greater equal zero, then x times y is zero or not less than x.

$$a \wedge b \rightarrow c \vee \neg d.$$

- This propositional formula ignores "for all numbers x and y".
- It uses propositional variables a, b, c, d to abstract from sentences:
  - a: "x is greater equal zero".
  - b: "y is greater equal zero".
  - c: "x times y is zero".
  - d: "x times y is less than x".
- The formula thus describes the "shape" of the sentence, but not its "content".

Propositional logic is not able to talk about concrete objects, their relationships, and the fact whether a sentence is true for all or just for just some objects of a domain.

The Syntax of First-Order Logic: Terms and Formulas

- First-order (predicate) logic has two kinds of syntactic phrases ("expressions"):
  - Terms denoting objects (values).
  - Formulas denoting properties of objects (i.e., the truth values "true" or "false").

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

$$F ::= \underline{p(t_1, \ldots, t_n)} \mid \top \mid \bot \mid (\neg F) \mid (F_1 \wedge F_2) \mid (F_1 \vee F_2) \mid (F_1 \rightarrow F_2) \mid (F_1 \leftrightarrow F_2)$$
$$\mid \underline{(\forall v \colon F)} \mid \underline{(\exists v \colon F)}$$

- The elements of the phrases:
  - $v \in \mathcal{V}$: a variable to which varying objects can be assigned.
  - $c \in \mathcal{C}$: a constant denoting a fixed object.
  - $f \in \mathcal{F}$: a function symbol of arity n denoting an n-ary function.
  - $p \in \mathcal{P}$: a predicate symbol of arity n denoting an n-ary predicate.
    - Functions return objects, while predicates return "true" or "false".
  - $\forall$ and $\exists$: a quantifier that binds a variable v within a formula F.
    - $\forall v \colon F$: "for <u>all</u> (possible objects assigned to) v, F is true".
    - $\exists v \colon F$: "there exists <u>some</u> (possible object assigned to) v, for which F is true".

Example

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

$$\big(\text{isFemale}(\text{Tanja}) \land (\forall x\colon (\text{isFemale}(x) \rightarrow \text{isDaughterOf}(x, \text{fatherOf}(x))))\big)$$

- "Names":
  - Tanja . . . a constant
  - x . . . a variable
  - isFemale, isDaughterOf . . . predicate symbols of arity 1/2 (return "true" or "false")
  - fatherOf . . . a function symbol of arity 1 (returns a person)
- Terms (denoting persons):
  - Tanja, x, fatherOf(x).
- (Sub)formulas (denoting "true" or "false"):
  - $\text{isFemale}(\text{Tanja})$
  - $\text{isFemale}(x)$
  - $\text{isDaughterOf}(x, \text{fatherOf}(x))$
  - $(\text{isFemale}(x) \rightarrow \text{isDaughterOf}(x, \text{fatherOf}(x)))$
  - $\big(\forall x\colon (\text{isFemale}(x) \rightarrow \text{isDaughterOf}(x, \text{fatherOf}(x)))\big)$

Formulas and Parentheses

We may reduce the number of parentheses by associating "binding powers" to operators:

- Binding powers:

$$(\neg) \gg (\wedge) \gg (\vee) \gg (\rightarrow) \gg (\leftrightarrow) \gg (\forall, \exists)$$

  - $(x) \gg (y)$: "operator x binds stronger than operator y": $(F_1\ x\ F_2\ y\ F_3)$ is interpreted as $((F_1\ x\ F_2)\ y\ F_3)$, not as $(F_1\ x\ (F_2\ y\ F_3))$.
- Quantified formulas:
  - Without parentheses, the scope of a quantified formula $\forall v\colon F$ or $\exists v\colon F$ reaches to the end of the enclosing formula.
- Formula simplification:

$$\big(\mathrm{isFemale}(\mathrm{Tanja}) \wedge \big(\forall x\colon (\mathrm{isFemale}(x) \rightarrow \mathrm{isDaughterOf}(x, \mathrm{fatherOf}(x)))\big)\big)$$

$$\rightsquigarrow \mathrm{isFemale}(\mathrm{Tanja}) \wedge \forall x\colon \mathrm{isFemale}(x) \rightarrow \mathrm{isDaughterOf}(x, \mathrm{fatherOf}(x))$$

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

Example

For all numbers x and y it is the case that, if x is greater equal zero and y is greater equal zero, then x times y is zero or not less than x.

$$a \wedge b \rightarrow c \vee \neg d.$$

$$\rightsquigarrow$$

$$\begin{aligned}\forall x\colon \forall y\colon\ & \mathsf{greaterEqual}(x, \mathsf{zero}) \wedge \mathsf{greaterEqual}(y, \mathsf{zero}) \rightarrow \\ & \mathsf{equal}(\mathsf{times}(x, y), \mathsf{zero}) \vee \neg \mathsf{lessThan}(\mathsf{times}(x, y), x)\end{aligned}$$

First-order logic is able to talk about objects and their properties.

First-Order Logic and Natural Language

- "Alex is Tom's sister":

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

- "Tom has a sister in Linz":

$$\exists x\colon \text{isSisterOf}(x, \text{Tom}) \land \text{livesIn}(x, \text{Linz})$$

- "Tom has two sisters":

$$\exists x, y\colon x \neq y \land \text{isSisterOf}(x, \text{Tom}) \land \text{isSisterOf}(y, \text{Tom})$$

- "Tom has no brother":

$$\neg\exists x\colon \text{isBrotherOf}(x, \text{Tom}) \qquad \text{(there does not exist a brother of Tom)}$$

$$\forall x\colon \neg\text{isBrotherOf}(x, \text{Tom}) \qquad \text{(everybody is not a brother of Tom)}$$

Many natural language statements can be expressed in first-order logic.

Abstract Syntax versus Concrete Syntax

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

- Abstract syntax: a "standard form" of expressions.
  - Prefix notation: atomic formulas $p(t_1,\ldots,t_n)$ and function applications $f(t_1,\ldots,t_n)$.
  - Predicate/function symbol p/f appears before the subexpressions $t_1,\ldots,t_n$.
  - Unique identification of the "type of the expression" (p/f) and its "subexpressions".
- Concrete syntax: any particular "notation" to write expressions.
  - One expression in abstract syntax can have many different forms in concrete syntax.
  - Infix notation $(a+i$, $a[i])$, postfix notation (r^*), subscript notation (a_i), ...

For understanding their meaning, we need to be able to translate expressions from concrete syntax to abstract syntax.

Abstract Syntax versus Concrete Syntax

Concrete Syntax	Abstract Syntax	
a/b	$/(a,b)$	quotient(a,b)
$\frac{a}{b}$	$/(a,b)$	quotient(a,b)
$a \vert b$	$\vert(a,b)$	divides(a,b)
$a = b$	$=(a,b)$	equals(a,b)
$a < b$	$<(a,b)$	less(a,b)
$\sqrt{a}$	$\sqrt{}(a)$	sqrt(a)
$a[i]$	$[\,](a,i)$	index(a,i)
a_i	$[\,](a,i)$	index(a,i)
$[a,b]$	$[\,](a,b)$	interval(a,b)
f'	$'(f)$	derivative(f)
$\int f$	$\int(f)$	integral(f)
$f \to a$	$\to(f,a)$	converges(f,a)

Concrete: $\frac{a}{a+b} < 1 \rightsquigarrow$ abstract: $<(/(a,+(a,b)),1)$ or: less(quotient(a,sum(a,b)),one).

Abstract Syntax versus Concrete Syntax

- The concrete syntax not always determines the abstract syntax uniquely:

Concrete Syntax	Abstract Syntax	
$a+b+c$	$+(a,b,c)$	$\mathsf{sum3}(a,b,c)$
	$+(a,+(b,c))$	$\mathsf{sum}(a,\mathsf{sum}(b,c))$
	$+(+(a,b),c)$	$\mathsf{sum}(\mathsf{sum}(a,b),c)$

- Translation of natural language to abstract syntax:

Concrete Syntax	Abstract Syntax
the sum of all values from a to b	$\mathsf{summation}(a,b)$
the remainder of a divided by b	$\mathsf{remainder}(a,b)$
a is a divisor of b	$\mathsf{divides}(a,b)$
f converges to a	$\mathsf{converges}(f,a)$

Conditions and Quantifiers

- Statements with constrained domain:
  Every natural number is greater equal zero.
  There exists a natural number whose predecessor is zero.
- Corresponding formulas with filtering condition:

$$\begin{array}{lcl} \forall x \in \mathbb{N} \colon x \geq 0 & & \forall x \colon x \in \mathbb{N} \to x \geq 0 \\ & \rightsquigarrow & \\ \exists x \in \mathbb{N} \colon x - 1 = 0 & & \exists x \colon x \in \mathbb{N} \land x - 1 = 0 \end{array}$$

- General pattern:

$$\begin{array}{lcl} \forall C \colon F & & \forall x \colon C \to F \\ & \rightsquigarrow & \\ \exists C \colon F & & \exists x \colon C \land F \end{array}$$

- Quantified variable must be deduced from context:

$$\forall x \in \mathbb{N} \colon \exists x < y \colon y < x + 2 \quad \rightsquigarrow \quad \forall x \colon x \in \mathbb{N} \to \exists y \colon x < y \land y < x + 2$$

Free and Bound Variables

- Non-closed formula:

$$\text{equal}(x, \text{zero})$$

  - Truth value depends on value we assign to x: "true" for $x = \text{zero}$, "false", otherwise.
  - Variable x is free in the formula.
  - If some of its variables are free, a formula is non-closed.
- Closed formulas:

$$\forall x\colon \text{equal}(x, \text{zero})$$

$$\exists x\colon \text{equal}(x, \text{zero})$$

  - Truth values do not depend on x: first formula is "false", second one is "true".
  - Variable x is bound in both formulas (by the quantifier $\forall$ respectively $\exists$).
  - If all of its variables are bound, a formula is closed.

The truth value of a formula only depends on the values assigned to the formula's free variables; the truth value is independent of the values of the bound variables.

The Free Variables of a Formula

The computation of the free variables proceeds "inside-out":

$$\underbrace{\forall x\colon \underbrace{\underbrace{p(x,w)}_{\text{free: } x,w} \to \underbrace{\exists y\colon \underbrace{q(x,y,z)}_{\text{free: } x,y,z}}_{\text{free: } x,z}}_{\text{free: } x,w,z}}_{\text{free: } w,z}$$

This computation can be formally described.

The Free Variables of a Formula

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

$$\begin{aligned}
\mathsf{fv}(p(t_1,\ldots,t_n)) &= \mathsf{fv}(t_1)\cup\ldots\cup\mathsf{fv}(t_n)\\
\mathsf{fv}(\top) &= \emptyset\\
\mathsf{fv}(\bot) &= \emptyset\\
\mathsf{fv}(\neg F) &= \mathsf{fv}(F)\\
\mathsf{fv}(F_1 \wedge F_2) &= \mathsf{fv}(F_1)\cup\mathsf{fv}(F_2)\\
\mathsf{fv}(F_1 \vee F_2) &= \mathsf{fv}(F_1)\cup\mathsf{fv}(F_2)\\
\mathsf{fv}(F_1 \rightarrow F_2) &= \mathsf{fv}(F_1)\cup\mathsf{fv}(F_2)\\
\mathsf{fv}(F_1 \leftrightarrow F_2) &= \mathsf{fv}(F_1)\cup\mathsf{fv}(F_2)\\
\mathsf{fv}(\forall v\colon F) &= \underline{\mathsf{fv}(F)\setminus\{v\}}\\
\mathsf{fv}(\exists v\colon F) &= \underline{\mathsf{fv}(F)\setminus\{v\}}
\end{aligned}$$

Quantifiers bind variables.

$$\mathsf{fv}(v) = \{v\} \qquad \mathsf{fv}(c) = \emptyset$$

$$\mathsf{fv}(f(t_1,\ldots,t_n)) = \mathsf{fv}(t_1)\cup\ldots\cup\mathsf{fv}(t_n)$$

Example

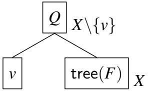
$$\begin{aligned}
\mathsf{fv}(q(x,y,z)) &= \{x,y,z\}\\
\mathsf{fv}(\exists y\colon q(x,y,z)) &= \mathsf{fv}(q(x,y,z))\setminus\{y\}\\
&= \{x,y,z\}\setminus\{y\} = \{x,z\}\\
\mathsf{fv}(p(x,w)) &= \{x,w\}\\
\mathsf{fv}(p(x,w)\rightarrow\exists y\colon q(x,y,z)) &= \mathsf{fv}(p(x,w))\cup\mathsf{fv}(\exists y\colon q(x,y,z))\\
&= \{x,w\}\cup\{x,z\} = \{x,w,z\}\\
\mathsf{fv}(\forall x\colon p(x,w)\rightarrow\exists y\colon q(x,y,z)) &= \mathsf{fv}(p(x,w)\rightarrow\exists y\colon q(x,y,z))\setminus\{x\}\\
&= \{x,w,z\}\setminus\{x\} = \{w,z\}
\end{aligned}$$

Syntax Analysis

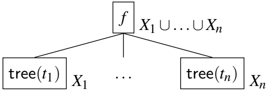
Generate from a formula's concrete syntax (a linear text with multiple interpretations) its abstract syntax tree (a data structure with only a single interpretation).

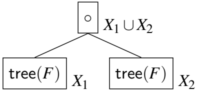
- Syntax analyisis of formula proceeds in top-down fashion by analyzing the formula's
  - quantified formulas (constructed by quantifiers from variables and sub-formulas),
  - propositional formulas (constructed by logical connectives from sub-formulas),
  - atomic formulas (constructed by predicate symbols from terms),
  - terms (variables or constants or constructed by function symbols from sub-terms).
- Determines the roles of names as variables, constants, function/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.
- Determines the free variables of every formula and term.

Syntax Analysis: Formal Definition

$\mathsf{tree}(Q\,v : F) =$ 

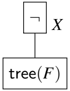
$Q \in \{\forall, \exists\}$

$\mathsf{tree}(f(t_1, \ldots, t_n)) =$ 

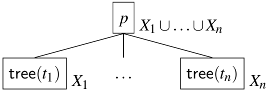
$\mathsf{tree}(F_1 \circ F_2) =$ 

$\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$

$\mathsf{tree}(c) = \boxed{c}_{\{\}}$ $\quad \mathsf{tree}(v) = \boxed{v}_{\{v\}}$

$\mathsf{tree}(\neg F) =$ 

$\mathsf{tree}(\top) = \boxed{\top}_{\{\}}$ $\quad \mathsf{tree}(\bot) = \boxed{\bot}_{\{\}}$

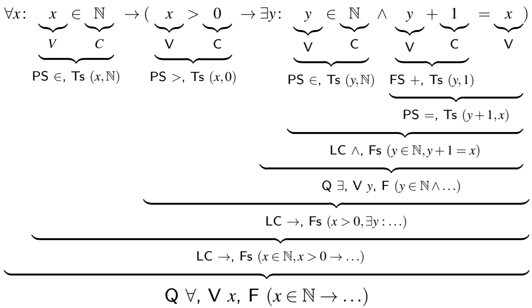
$\mathsf{tree}(p(t_1, \ldots, t_n)) =$ 

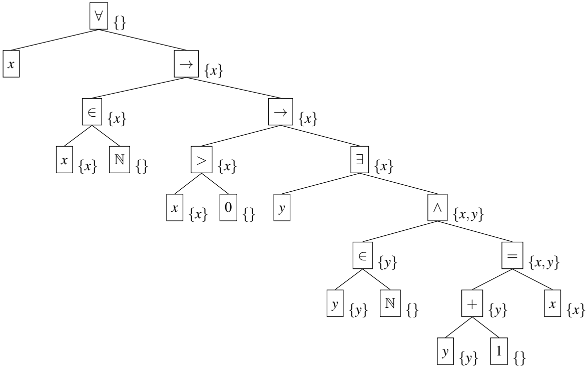
Syntax Analysis: Example

$$\forall x \in \mathbb{N} \colon x > 0 \to \exists y \in \mathbb{N} \colon y + 1 = x$$

$$\rightsquigarrow \quad \forall x \colon x \in \mathbb{N} \to (x > 0 \to \exists y \colon y \in \mathbb{N} \land y + 1 = x)$$

$$\rightsquigarrow \quad (\forall x \colon ((x \in \mathbb{N}) \to ((x > 0) \to (\exists y \colon ((y \in \mathbb{N}) \land (y + 1 = x))))))$$

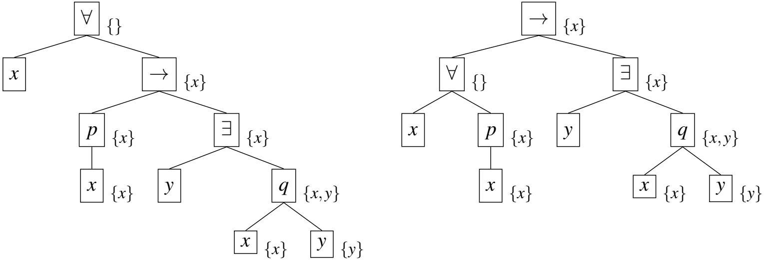




Q . . . quantifier, V . . . variable, F(s) . . . formula(s), LC . . . logical connective,
T(s) . . . term(s), PS . . . predicate symbol, FS . . . function symbol

Syntax Analysis: Pitfalls

$$\forall x\colon p(x) \rightarrow \exists y\colon q(x,y)$$



By the precedence rules, the formula has to be parenthesized as $\forall x\colon (p(x) \rightarrow \exists y\colon q(x,y))$, not as $(\forall x\colon p(x)) \rightarrow (\exists y\colon q(x,y))$; therefore the left syntax tree is the correct one.

Further Constructs: Language Extensions

- Local definition: $(\textbf{let } v = t \textbf{ in } E)$ (also: $(E \textbf{ where } v = t)$ or $(E|_{v=t})$)
  - E can be a formula or a term, phrase is correspondingly a formula or a term.
  - Phrase means $E[t/v]$ (every free occurrence of v in E is replaced by t); thus v is bound.
  - Formula $(\textbf{let } v = t \textbf{ in } F)$ is equivalent to:

$$\exists v\colon (v = t \wedge F)$$

- Conditional expression: $(\textbf{if } F \textbf{ then } E_1 \textbf{ else } E_2)$
  - E_1, E_2 can be both formulas or both terms, phrase is correspondingly formula or term.
  - Phrase means E_1, if F is true, and E_2, otherwise.
  - Formula $(\textbf{if } F \textbf{ then } F_1 \textbf{ else } F_2)$ is equivalent to:

$$(F \rightarrow F_1) \wedge (\neg F \rightarrow F_2)$$

Not strictly necessary but often convenient in practice.

Further Constructs: Mathematical Quantifiers

- $\sum\limits_{i=a}^{b} t$ binds variable i; its meaning is the sum $t[a/i] + \cdots + t[b/i]$.
- $\prod\limits_{i=a}^{b} t$ binds variable i; its meaning is the product $t[a/i] * \cdots * t[b/i]$.
- $\{x \in S \mid F\}$ binds x; it denotes the set of all x from set S for which F is true.
- $\{t \mid x \in S \wedge F\}$ binds x; it denotes the set of all t where x is from S and F is true.
- $\lim\limits_{x \to v} t$ binds variable x; its meaning is the limit of term t when x goes to value v.
- $\max\limits_{x \in S} t$ binds x; it denotes the maximum of all values of t where x is from S.
- $\min\limits_{x \in S} t$ binds x; it denotes the minimum of all values of t where x is from S.
- ...

Mathematics provides a great variety of variable binding constructs (i.e., quantifiers).