22c181:
Formal Methods in Software Engineering

The University of Iowa

Spring 2008

Typed First-order Logic

Contents

Propositional Logic is insufficient

A ALL PERSONS ARE HAPPY

Propositional Logic is insufficient

A	**ALL PERSONS ARE HAPPY**
B	**PAT IS A PERSON**

Propositional Logic is insufficient

A	All persons are happy
B	Pat is a person
$?$	Pat is happy

Propositional logic lacks possibility to talk about individuals

In particular, need to model objects, attributes, associations, etc.

Propositional Logic is insufficient

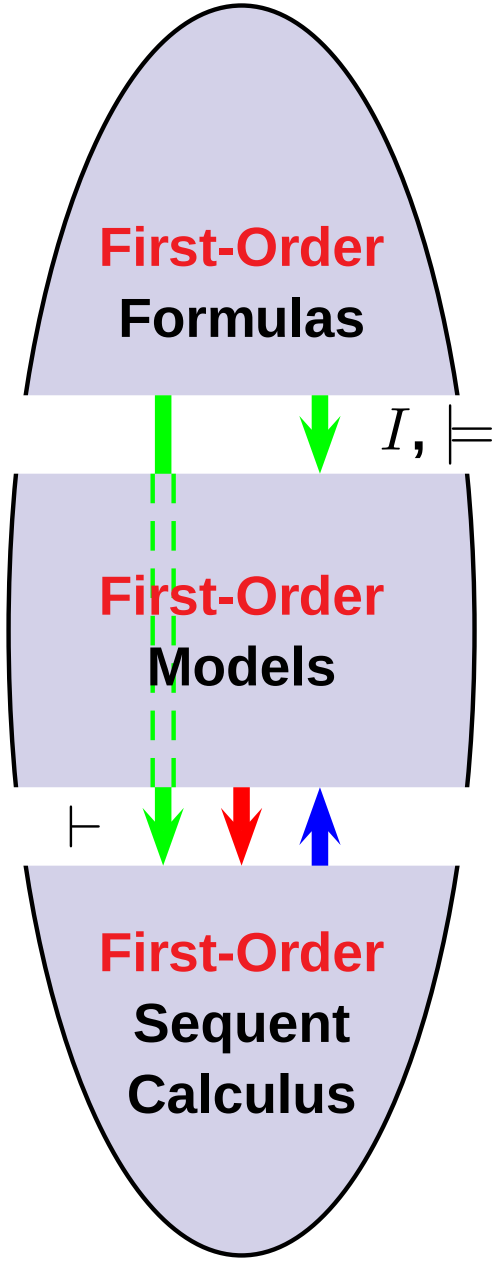
A	All persons are happy
B	Pat is a person
$?$	Pat is happy

Propositional logic lacks possibility to talk about individuals

In particular, need to model objects, attributes, associations, etc.

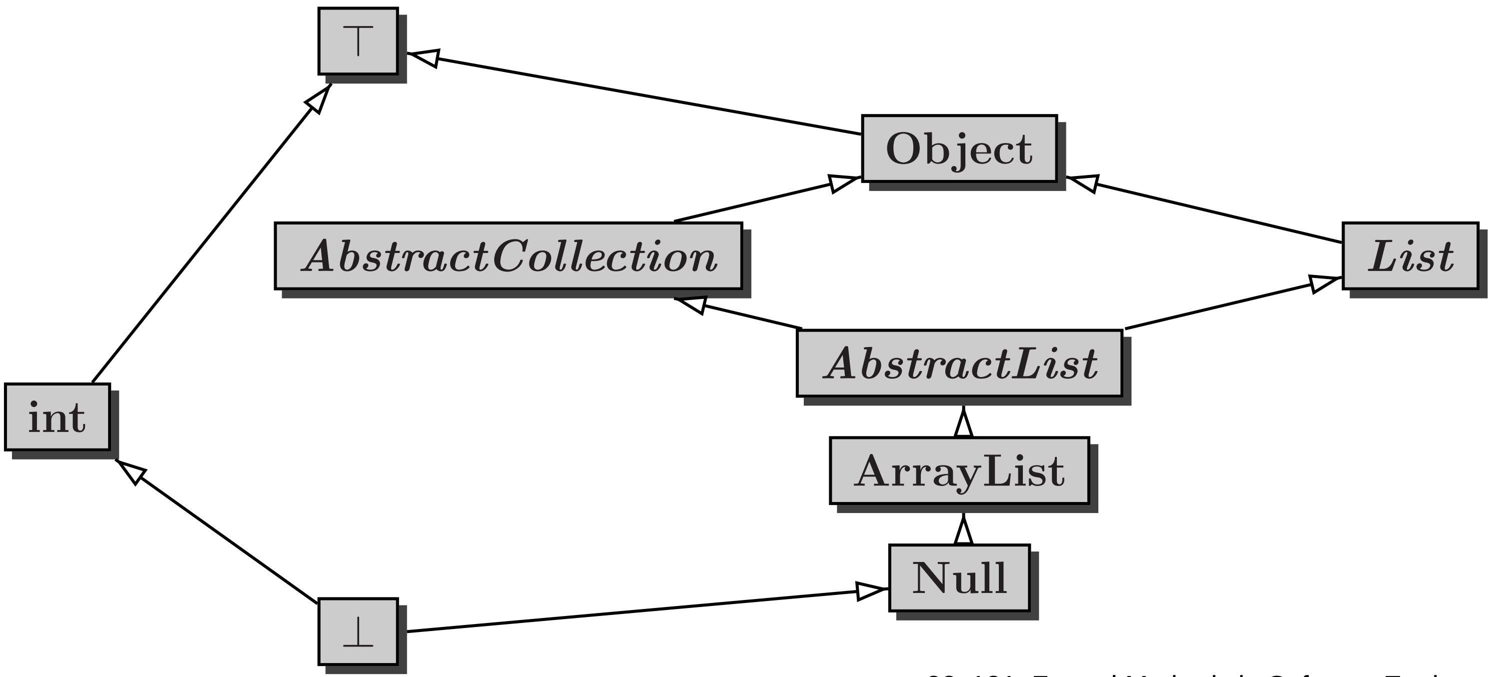
$\Rightarrow$ First-Order Logic (FOL) with Types

First-Order Logic



OO Type Hierarchy

- Finite set $\mathcal{T}$ of **static types**, **subtype** relation $\sqsubseteq$,
- **Dynamic types** $\mathcal{T}_d \subseteq \mathcal{T}$, where $\top \in \mathcal{T}_d$
- **Abstract types** $\mathcal{T}_a \subseteq \mathcal{T}$, where $\bot \in \mathcal{T}_a$
- $\mathcal{T}_d \cap \mathcal{T}_a = \emptyset$, $\quad \mathcal{T}_d \cup \mathcal{T}_a = \mathcal{T}$, $\quad \bot \sqsubseteq z \sqsubseteq \top$ for all $z \in \mathcal{T}$



Signature of Typed First-Order Logic

Given type hierarchy $(\mathcal{T}, \mathcal{T}_d, \mathcal{T}_a, \sqsubseteq)$**,** **let** $\mathcal{T}_q := \mathcal{T} \setminus \{\perp\}$

Signature $\Sigma = (\mathbf{V}, \mathbf{P}, \mathbf{F}, \alpha)$

Signature of Typed First-Order Logic

Given type hierarchy $(\mathcal{T}, \mathcal{T}_d, \mathcal{T}_a, \sqsubseteq)$**,** **let** $\mathcal{T}_q := \mathcal{T} \setminus \{\perp\}$

Signature $\Sigma = (\mathbf{V}, \mathbf{P}, \mathbf{F}, \alpha)$

Variable Symbols $\mathbf{V} = \{x_i \mid i \in \mathbb{N}\}$

Predicate Symbols $\mathbf{P} = \{p_i \mid i \in \mathbb{N}\}$

Function Symbols $\mathbf{F} = \{f_i \mid i \in \mathbb{N}\}$

Signature of Typed First-Order Logic

Given type hierarchy $(\mathcal{T}, \mathcal{T}_d, \mathcal{T}_a, \sqsubseteq)$**,** **let** $\mathcal{T}_q := \mathcal{T} \setminus \{\bot\}$

Signature $\Sigma = (\mathbf{V}, \mathbf{P}, \mathbf{F}, \alpha)$

Variable Symbols $\mathbf{V} = \{x_i \mid i \in \mathbb{N}\}$

Predicate Symbols $\mathbf{P} = \{p_i \mid i \in \mathbb{N}\}$

Function Symbols $\mathbf{F} = \{f_i \mid i \in \mathbb{N}\}$

Typing function α **for all symbols:**

- $\alpha(x) \in \mathcal{T}_q$ **for all** $x \in \mathbf{V}$
  We write $x{:}z$ **instead of** $\alpha(x) = z$ **(in Java: "`z t;`")**

- $\alpha(p) \in \mathcal{T}_q^*$ **for all** $p \in \mathbf{P}$
  We write $p{:}\, z_1, \ldots, z_r$ **intead of** $\alpha(p) = (z_1, \ldots, z_r)$

- $\alpha(f) \in \mathcal{T}_q^* \times \mathcal{T}_q$ **for all** $f \in \mathbf{F}$
  We write $f{:}\, z_1, \ldots, z_r \to z$ **instead of** $\alpha(f) = ((z_1, \ldots, z_r), z)$

$r = 0$ **ok, No overloading of variables, functions, predicates!**

Special Signature Symbols

An **Equality** symbol $\doteq$ in $\mathbf{P}$, with typing $\doteq : \top, \top$

A **type predicate** symbol $\sqsubseteq_z$ in $\mathbf{P}$ for each $z \in \mathcal{T}_q$.
with typing $\sqsubseteq_z : \top$

Type cast symbol (z) in $\mathbf{F}$ for each $z \in \mathcal{T}_q$,
with typing $(z) : \top, z$

First-Order Signature Example

Sticks and stones may break your bones, but flowers will never hurt

First-Order Signature Example

Sticks and stones may break your bones, but flowers will never hurt

Types $\mathcal{T}_d = \{\mathbf{Stick}, \mathbf{Stone}, \mathbf{Flower}\}, \quad \mathcal{T}_a = \{\mathbf{Weapon}, \mathbf{Any}\}$

$\mathbf{Stick}, \mathbf{Stone} \sqsubseteq \mathbf{Weapon} \sqsubseteq \mathbf{Any}, \ \mathbf{Flower} \sqsubseteq \mathbf{Any}$

Predicates $\mathbf{P} = \{\mathbf{hurts} : \ \mathbf{Any}\}$

Functions $\mathbf{F} = \{\mathbf{stick} : \rightarrow \mathbf{Stick}, \ \mathbf{stone} : \rightarrow \mathbf{Stone}, \mathbf{r} : \rightarrow \mathbf{Flower}\}$

Function with empty argument list: constant

First-Order Signature Example

Sticks and stones may break your bones, but flowers will never hurt

Types $\mathcal{T}_d = \{\mathbf{Stick}, \mathbf{Stone}, \mathbf{Flower}\}, \quad \mathcal{T}_a = \{\mathbf{Weapon}, \mathbf{Any}\}$

$\mathbf{Stick}, \mathbf{Stone} \sqsubseteq \mathbf{Weapon} \sqsubseteq \mathbf{Any}, \; \mathbf{Flower} \sqsubseteq \mathbf{Any}$

Predicates $\mathbf{P} = \{\mathbf{hurts} : \; \mathbf{Any}\}$

Functions $\mathbf{F} = \{\mathbf{stick} : \to \mathbf{Stick}, \; \mathbf{stone} : \to \mathbf{Stone}, \mathbf{r} : \to \mathbf{Flower}\}$

Function with empty argument list: constant

cf. KeY book p28

Terms of First-Order Logic

Given signature $(\mathbf{V}, \mathbf{P}, \mathbf{F}, \alpha)$

Terms: **Set** $Term_z$ **of terms of type** z**, one for each static type** $z \in \mathcal{T}$

- x **is term of type** z **for each variable** $x : z$
- $f(t_1, \ldots, t_r)$ **is term of type** z **for each function symbol** $f : z_1, \ldots, z_r \to z$ **and terms** t_i **of type** $z_i' \sqsubseteq z_i$ **for** $1 \leq i \leq r$

  If f **is constant (**$r = 0$**) we write** f **instead of** $f()$

Terms of First-Order Logic

Given signature $(\mathbf{V}, \mathbf{P}, \mathbf{F}, \alpha)$

Terms: Set $Term_z$ of terms of type z, one for each static type $z \in \mathcal{T}$

- x **is term of type z for each variable $x : z$**
- $f(t_1, \ldots, t_r)$ **is term of type z for each function symbol** $f : z_1, \ldots, z_r \to z$ **and terms t_i of type $z_i' \sqsubseteq z_i$ for $1 \leq i \leq r$**

  If f is constant ($r = 0$) we write f instead of $f()$

Example:

$\mathcal{T}_d = \{\mathbf{Car}, \mathbf{Person}, \top\}$ **where** $\mathbf{Person} \sqsubseteq \top$, $\mathbf{Car} \sqsubseteq \top$

$\mathbf{F} = \{\mathbf{owner} : \mathbf{Car} \to \mathbf{Person}, \mathbf{pat} : \to \mathbf{Person}, \mathbf{herbie} : \to \mathbf{Car}\}$, $x : \mathbf{Car}$

Terms: $\mathbf{herbie}$, $\mathbf{owner}(\mathbf{herbie})$, $\mathbf{owner}((\mathbf{Car})\mathbf{pat})$ **(!)**, $\mathbf{owner}(x)$

Non-terms: $\mathbf{Car}$, $\mathbf{owner}(\mathbf{pat})$, $\mathbf{owner}((\mathbf{Person})\mathbf{herbie})$

Formulas of First-Order Logic

First-Order Formulas: Set For of (first-order) formulas

- $p(t_1, \ldots, t_r)$ is an **atomic** formula for predicate symbol $p : z_1, \ldots, z_r$ and terms t_i of type $z'_i \sqsubseteq z_i$ for $1 \leq i \leq r$
- **Truth constants**, **connectives** as in propositional logic
- If x is any variable, ϕ a formula, then $\forall x \,.\, \phi$ and $\exists x \,.\, \phi$ are formulas

  We call ϕ the **scope** of variable x. We say that x is **bound** by the **quantifier** $\forall$ in $\forall x \,.\, \phi$ (similarly for $\exists x \,.\, \phi$)

Formulas of First-Order Logic

First-Order Formulas: Set For of (first-order) formulas

- $p(t_1, \ldots, t_r)$ is an **atomic** formula for predicate symbol $p : z_1, \ldots, z_r$ and terms t_i of type $z'_i \sqsubseteq z_i$ for $1 \leq i \leq r$
- **Truth constants**, **connectives** as in propositional logic
- If x is any variable, ϕ a formula, then $\forall x . \phi$ and $\exists x . \phi$ are formulas

  We call ϕ the **scope** of variable x. We say that x is **bound** by the **quantifier** $\forall$ in $\forall x . \phi$ (similarly for $\exists x . \phi$)

Bound variables in quantified formulas are analogous to local variables/formal parameters in programs

Use pathentheses and usual precedence rules to avoid syntactic ambiguity

First-Order Syntax Example

Sticks and stones may break your bones, but flowers will never hurt

Types $\mathcal{T}_d = \{\mathbf{Stick}, \mathbf{Stone}, \mathbf{Flower}\}, \quad \mathcal{T}_a = \{\mathbf{Weapon}, \mathbf{Any}\}$

$\mathbf{Stick}, \mathbf{Stone} \sqsubseteq \mathbf{Weapon} \sqsubseteq \mathbf{Any}, \; \mathbf{Flower} \sqsubseteq \mathbf{Any}$

Predicates $\mathbf{P} = \{\mathbf{hurts} : \; \mathbf{Any}\}$

Functions $\mathbf{F} = \{\mathbf{stick} : \to \mathbf{Stick}, \; \mathbf{stone} : \to \mathbf{Stone}, \mathbf{r} : \to \mathbf{Flower}\}$

Variables $\mathbf{V} = \{x : \mathbf{Weapon}, \; y : \mathbf{Flower}\}$

Examples:

First-Order Syntax Example

Sticks and stones may break your bones, but flowers will never hurt

Types $\mathcal{T}_d = \{\mathbf{Stick}, \mathbf{Stone}, \mathbf{Flower}\}, \quad \mathcal{T}_a = \{\mathbf{Weapon}, \mathbf{Any}\}$

$\mathbf{Stick}, \mathbf{Stone} \sqsubseteq \mathbf{Weapon} \sqsubseteq \mathbf{Any},\ \mathbf{Flower} \sqsubseteq \mathbf{Any}$

Predicates $\mathbf{P} = \{\mathbf{hurts} : \ \mathbf{Any}\}$

Functions $\mathbf{F} = \{\mathbf{stick} : \rightarrow \mathbf{Stick},\ \mathbf{stone} : \rightarrow \mathbf{Stone}, \mathbf{r} : \rightarrow \mathbf{Flower}\}$

Variables $\mathbf{V} = \{x : \mathbf{Weapon},\ y : \mathbf{Flower}\}$

Examples:

$$\forall x\,.\,\mathbf{hurts}(x) \quad \mathbf{\&} \quad \forall y\,.\,\mathbf{!hurts}(y)$$

We sometimes write the type of quantified variables explicitly.

First-Order Syntax Example

Sticks and stones may break your bones, but flowers will never hurt

Types $\mathcal{T}_d = \{\textbf{Stick}, \textbf{Stone}, \textbf{Flower}\}, \quad \mathcal{T}_a = \{\textbf{Weapon}, \textbf{Any}\}$

$\textbf{Stick}, \textbf{Stone} \sqsubseteq \textbf{Weapon} \sqsubseteq \textbf{Any}, \; \textbf{Flower} \sqsubseteq \textbf{Any}$

Predicates $\mathbf{P} = \{\textbf{hurts} : \; \textbf{Any}\}$

Functions $\mathbf{F} = \{\textbf{stick} : \rightarrow \textbf{Stick}, \; \textbf{stone} : \rightarrow \textbf{Stone}, \textbf{r} : \rightarrow \textbf{Flower}\}$

Variables $\mathbf{V} = \{x : \textbf{Weapon}, \; y : \textbf{Flower}\}$

Examples:

$$\forall\, x : \textbf{Weapon} \,.\, \textbf{hurts}(x) \;\&\; \forall\, y : \textbf{Flower} \,.\, !\textbf{hurts}(y)$$

First-Order Syntax Example

Sticks and stones may break your bones, but flowers will never hurt

Types $\mathcal{T}_d = \{\textbf{Stick}, \textbf{Stone}, \textbf{Flower}\}, \quad \mathcal{T}_a = \{\textbf{Weapon}, \textbf{Any}\}$

$\textbf{Stick}, \textbf{Stone} \sqsubseteq \textbf{Weapon} \sqsubseteq \textbf{Any}, \; \textbf{Flower} \sqsubseteq \textbf{Any}$

Predicates $\textbf{P} = \{\textbf{hurts} : \; \textbf{Any}\}$

Functions $\textbf{F} = \{\textbf{stick} : \rightarrow \textbf{Stick}, \textbf{stone} : \rightarrow \textbf{Stone}, \textbf{r} : \rightarrow \textbf{Flower}\}$

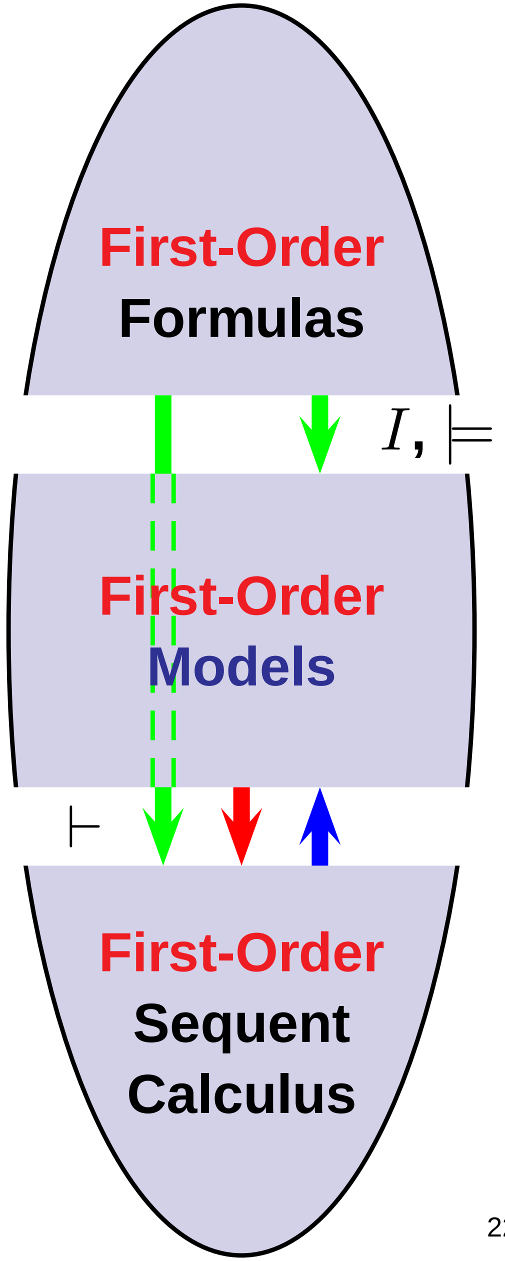
Variables $\textbf{V} = \{x : \textbf{Weapon}, \; y : \textbf{Flower}\}$

Examples:

$$\forall\, x : \textbf{Weapon} \,.\, \textbf{hurts}(x) \;\textbf{\&}\; \forall\, y : \textbf{Flower} \,.\, \textbf{!hurts}(y)$$

$$\textbf{hurts}(\textbf{r}) \;\; \textbf{->} \;\; \exists\, y \,.\, \textbf{hurts}(y)$$

Semantics of First-Order Logic



Semantics of First-Order Logic

A model of FOL is a triple $\mathcal{M} = (\mathcal{D}, \delta, \mathcal{I})$ where

- $\mathcal{D}$ is the universe or domain

  Contains "objects" and "values"

- δ is a dynamic typing function $\delta : \mathcal{D} \rightarrow \mathcal{T}_d$

  Each domain element has dynamic ("runtime") type

- $\mathcal{I}$ is an interpretation of the function and predicate symbols s.t.
  - If $p : z_1, \ldots, z_r \in \mathbf{P}$, then $\mathcal{I}(p) \subseteq \mathcal{D}^{z_1} \times \cdots \times \mathcal{D}^{z_r}$
  - If $f : z_1, \ldots, z_r \rightarrow z \in \mathbf{F}$, then $\mathcal{I}(f) : \mathcal{D}^{z_1} \times \cdots \times \mathcal{D}^{z_r} \rightarrow \mathcal{D}^z$

Moreover, let $\mathcal{D}^z = \{d \in \mathcal{D} \mid \delta(d) \sqsubseteq z\}$

(the domain elements of type z).

The dynamic types $z \in \mathcal{T}_d$ must be non-empty: $\mathcal{D}^z \neq \emptyset$

Semantics of Special Symbols

Equality symbol $\doteq$ in $\mathbf{P}$, with typing $\doteq: \top, \top$

Semantics: $\mathcal{I}(\doteq) = \{(d, d) \mid d \in \mathcal{D}\} \subseteq \mathcal{D}^{\top} \times \mathcal{D}^{\top}$

"Referential Equality"

Semantics of Special Symbols

Equality symbol $\doteq$ **in** $\mathbf{P}$**, with typing** $\doteq : \top, \top$

Semantics: $\mathcal{I}(\doteq) = \{(d, d) \mid d \in \mathcal{D}\} \subseteq \mathcal{D}^{\top} \times \mathcal{D}^{\top}$

"Referential Equality"

Type predicate symbol $\sqsubseteq_z$ **in** $\mathbf{P}$ **for each** $z \in \mathcal{T}_q$**, with typing** $\sqsubseteq_z : \top$

Semantics: $\mathcal{I}(\sqsubseteq_z) = \mathcal{D}^z \subseteq \mathcal{D}^{\top}$

Semantics of Special Symbols

Equality symbol $\doteq$ **in** $\mathbf{P}$**, with typing** $\doteq : \top, \top$

Semantics: $\mathcal{I}(\doteq) = \{(d,d) \mid d \in \mathcal{D}\} \subseteq \mathcal{D}^\top \times \mathcal{D}^\top$

"Referential Equality"

Type predicate symbol $\sqsubseteq_z$ **in** $\mathbf{P}$ **for each** $z \in \mathcal{T}_q$**, with typing** $\sqsubseteq_z : \top$

Semantics: $\mathcal{I}(\sqsubseteq_z) = \mathcal{D}^z \subseteq \mathcal{D}^\top$

Type cast symbol (z) **in** $\mathbf{F}$ **for each** $z \in \mathcal{T}_q$**, with typing** $(z) : \top, z$

Semantics: $\mathcal{I}((z))$ **is a function such that**

$$\mathcal{I}((z))(x) = \begin{cases} x & \textbf{if } \delta(x) \sqsubseteq z \\ d & \textbf{otherwise} \end{cases}$$

with d **an arbitrary but fixed element of** $\mathcal{D}^z$

Semantics of First-Order Logic: Example

Sticks and stones may break your bones, but flowers will never hurt

Types $\mathcal{T}_d = \{\mathbf{Stick}, \mathbf{Stone}, \mathbf{Flower}\}, \quad \mathcal{T}_a = \{\mathbf{Weapon}, \mathbf{Any}\}$

$\mathbf{Stick}, \mathbf{Stone} \sqsubseteq \mathbf{Weapon} \sqsubseteq \mathbf{Any}, \ \mathbf{Flower} \sqsubseteq \mathbf{Any}$

Predicates $\mathbf{P} = \{\mathbf{hurts} : \ \mathbf{Any}\}$

Functions $\mathbf{F} = \{\mathbf{stick} : \to \mathbf{Stick}, \ \mathbf{stone} : \to \mathbf{Stone}, \mathbf{r} : \to \mathbf{Flower}\}$

Variables $\mathbf{V} = \{x : \mathbf{Weapon}, \ y : \mathbf{Flower}\}$

One of (infinitely) many possible models:

Semantics of First-Order Logic: Example

Sticks and stones may break your bones, but flowers will never hurt

Types $\mathcal{T}_d = \{\mathbf{Stick}, \mathbf{Stone}, \mathbf{Flower}\}, \quad \mathcal{T}_a = \{\mathbf{Weapon}, \mathbf{Any}\}$

$\mathbf{Stick}, \mathbf{Stone} \sqsubseteq \mathbf{Weapon} \sqsubseteq \mathbf{Any}, \ \mathbf{Flower} \sqsubseteq \mathbf{Any}$

Predicates $\mathbf{P} = \{\mathbf{hurts} : \ \mathbf{Any}\}$

Functions $\mathbf{F} = \{\mathbf{stick} : \rightarrow \mathbf{Stick}, \ \mathbf{stone} : \rightarrow \mathbf{Stone}, \mathbf{r} : \rightarrow \mathbf{Flower}\}$

Variables $\mathbf{V} = \{x : \mathbf{Weapon}, \ y : \mathbf{Flower}\}$

One of (infinitely) many possible models:

Domain $\mathcal{D} = \{o_1, o_2, o_3, o_4\}$

Semantics of First-Order Logic: Example

Sticks and stones may break your bones, but flowers will never hurt

Types $\mathcal{T}_d = \{\textbf{Stick}, \textbf{Stone}, \textbf{Flower}\}, \quad \mathcal{T}_a = \{\textbf{Weapon}, \textbf{Any}\}$

$\textbf{Stick}, \textbf{Stone} \sqsubseteq \textbf{Weapon} \sqsubseteq \textbf{Any}, \ \textbf{Flower} \sqsubseteq \textbf{Any}$

Predicates $\textbf{P} = \{\textbf{hurts} : \ \textbf{Any}\}$

Functions $\textbf{F} = \{\textbf{stick} : \to \textbf{Stick}, \ \textbf{stone} : \to \textbf{Stone}, \textbf{r} : \to \textbf{Flower}\}$

Variables $\textbf{V} = \{x : \textbf{Weapon}, \ y : \textbf{Flower}\}$

One of (infinitely) many possible models:

Domain $\mathcal{D} = \{o_1, o_2, o_3, o_4\}$

Typing $\delta(o_1) = \delta(o_4) = \textbf{Stick}, \quad \delta(o_2) = \textbf{Stone}, \quad \delta(o_3) = \textbf{Flower}$

$\mathcal{D}^{\textbf{Stick}} = \{o_1, o_4\}$, $\mathcal{D}^{\textbf{Stone}} = \{o_2\}$, $\mathcal{D}^{\textbf{Flower}} = \{o_3\}$, $\mathcal{D}^{\textbf{Any}} = \{o_1, o_2, o_3, o_4\}$

Semantics of First-Order Logic: Example

Sticks and stones may break your bones, but flowers will never hurt

Types $\mathcal{T}_d = \{\mathbf{Stick}, \mathbf{Stone}, \mathbf{Flower}\}, \quad \mathcal{T}_a = \{\mathbf{Weapon}, \mathbf{Any}\}$

$\mathbf{Stick}, \mathbf{Stone} \sqsubseteq \mathbf{Weapon} \sqsubseteq \mathbf{Any}, \; \mathbf{Flower} \sqsubseteq \mathbf{Any}$

Predicates $\mathbf{P} = \{\mathbf{hurts} : \; \mathbf{Any}\}$

Functions $\mathbf{F} = \{\mathbf{stick} : \rightarrow \mathbf{Stick}, \; \mathbf{stone} : \rightarrow \mathbf{Stone}, \mathbf{r} : \rightarrow \mathbf{Flower}\}$

Variables $\mathbf{V} = \{x : \mathbf{Weapon}, \; y : \mathbf{Flower}\}$

One of (infinitely) many possible models:

Domain $\mathcal{D} = \{o_1, o_2, o_3, o_4\}$

Typing $\delta(o_1) = \delta(o_4) = \mathbf{Stick}, \quad \delta(o_2) = \mathbf{Stone}, \quad \delta(o_3) = \mathbf{Flower}$

$\mathcal{D}^{\mathbf{Stick}} = \{o_1, o_4\}$, $\mathcal{D}^{\mathbf{Stone}} = \{o_2\}$, $\mathcal{D}^{\mathbf{Flower}} = \{o_3\}$, $\mathcal{D}^{\mathbf{Any}} = \{o_1, o_2, o_3, o_4\}$

Interpretation $\mathcal{I}(\mathbf{hurts}) = \{o_1, o_2, o_4\}$

$\mathcal{I}(\mathbf{stick}) = o_1$, $\quad \mathcal{I}(\mathbf{stone}) = o_2$, $\quad \mathcal{I}(\mathbf{r}) = o_3$

Semantics of First-Order Logic, Cont'd

Assigning meaning to variables

Let x be variable of static type z

A Variable Assignment β maps x to an element of $\mathcal{D}^z$

Semantics of First-Order Logic, Cont'd

Assigning meaning to variables

Let x be variable of static type z

A Variable Assignment β maps x to an element of $\mathcal{D}^z$

Assigning meaning to terms: a mapping $val_{\mathcal{M},\beta}$ from $Term_z(t)$ to $\mathcal{D}^z$ (dependind on model $\mathcal{M}$ and variable assignment β) such that

- $val_{\mathcal{M},\beta}(x) = \beta(x)$ **(element in $\mathcal{D}^z$, where x has type z)**
- $val_{\mathcal{M},\beta}(f(t_1, \ldots, t_r)) = \mathcal{I}(f)(val_{\mathcal{M},\beta}(t_1), \ldots, val_{\mathcal{M},\beta}(t_r))$

Semantics of First-Order Logic, Cont'd

Assigning meaning to variables

Let x be variable of static type z

A Variable Assignment β maps x to an element of $\mathcal{D}^z$

Assigning meaning to terms: **a mapping $val_{\mathcal{M},\beta}$ from $Term_z(t)$ to $\mathcal{D}^z$ (dependind on model $\mathcal{M}$ and variable assignment β) such that**

- $val_{\mathcal{M},\beta}(x) = \beta(x)$ **(element in $\mathcal{D}^z$, where x has type z)**
- $val_{\mathcal{M},\beta}(f(t_1, \ldots, t_r)) = \mathcal{I}(f)(val_{\mathcal{M},\beta}(t_1), \ldots, val_{\mathcal{M},\beta}(t_r))$

Modified variable assignment:

For $d \in \mathcal{D}^z$ let $\beta_y^d(x) := \begin{cases} \beta(x) & \text{if } x \neq y \\ d & \text{if } x = y \end{cases}$

Semantics of First-Order Logic, Cont'd

Assigning meaning to formulas

Validity relation: $\mathcal{M}, \beta \models \phi$ **for** $\phi \in For$

- $\mathcal{M}, \beta \models p(t_1, \ldots, t_r)$ **iff** $(val_{\mathcal{M},\beta}(t_1), \ldots, val_{\mathcal{M},\beta}(t_r)) \in \mathcal{I}(p)$
- $\mathcal{M}, \beta \models \phi$ **&** ψ **iff** $\mathcal{M}, \beta \models \phi$ **and** $\mathcal{M}, \beta \models \psi$
- **. . .**
- $\mathcal{M}, \beta \models \forall x \, . \, \phi$ **iff** $\mathcal{M}, \beta_x^d \models \phi$ **for all** $d \in \mathcal{D}^z$
  where the type of x **is** z
- $\mathcal{M}, \beta \models \exists x \, . \, \phi$ **iff** $\mathcal{M}, \beta_x^d \models \phi$ **for at least one** $d \in \mathcal{D}^z$
  where the type of x **is** z

Semantics of First-Order Logic: Example

Sticks and stones may break your bones, but flowers will never hurt

Types $\mathcal{T}_d = \{\textbf{Stick}, \textbf{Stone}, \textbf{Flower}\}, \quad \mathcal{T}_a = \{\textbf{Weapon}, \textbf{Any}\}$

$\textbf{Stick}, \textbf{Stone} \sqsubseteq \textbf{Weapon} \sqsubseteq \textbf{Any}, \ \textbf{Flower} \sqsubseteq \textbf{Any}$

Predicates $\mathbf{P} = \{\textbf{hurts} : \ \textbf{Any}\}$

Functions $\mathbf{F} = \{\textbf{stick} : \rightarrow \textbf{Stick}, \ \textbf{stone} : \rightarrow \textbf{Stone}, \textbf{r} : \rightarrow \textbf{Flower}\}$

Variables $\mathbf{V} = \{x : \textbf{Weapon}, \ y : \textbf{Flower}\}$

In our previous model $\mathcal{M}$:

$\mathcal{D}^{\textbf{Stick}} = \{o_1, o_4\}, \quad \mathcal{D}^{\textbf{Stone}} = \{o_2\}, \quad \mathcal{D}^{\textbf{Flower}} = \{o_3\}$

$\mathcal{D}^{\textbf{Weapon}} = \{o_1, o_2, o_4\}, \quad \mathcal{I}(\textbf{hurts}) = \{o_1, o_2, o_4\} \subseteq \mathcal{D}^{\textbf{Any}}$

Evaluate these formulas: $\exists x \,.\, \textbf{hurts}(x), \quad \forall x \,.\, \textbf{hurts}(x), \quad \exists y \,.\, \textbf{hurts}(y)$

Semantics of First-Order Logic: Evaluation Example

Let β be arbitrary.

$\mathcal{M}, \beta \models \exists x : \mathbf{Weapon} \,.\, \mathbf{hurts}(x)$ **iff**

Semantic Rule

Information from model $(\mathcal{D}, \delta, \mathcal{I})$

Semantics of First-Order Logic: Evaluation Example

Let β be arbitrary.

$\mathcal{M}, \beta \models \exists\, x : \mathbf{Weapon}\, .\, \mathbf{hurts}(x)$ **iff**

There exists $d \in \mathcal{D}^{\mathbf{Weapon}}$ such that $\mathcal{M}, \beta_x^d \models \mathbf{hurts}(x)$ if

Semantic Rule

$\mathcal{M}, \beta \models \exists\, x\, .\, \phi$ **iff** $\mathcal{M}, \beta_x^d \models \phi$ **for at least one $d \in \mathcal{D}^z$**
where the type of x is z

Information from model $(\mathcal{D}, \delta, \mathcal{I})$

Semantics of First-Order Logic: Evaluation Example

Let β be arbitrary.

$\mathcal{M}, \beta \models \exists\, x : \mathbf{Weapon}\,.\,\mathbf{hurts}(x)$ **iff**

There exists $d \in \mathcal{D}^{\mathbf{Weapon}}$ such that $\mathcal{M}, \beta_x^d \models \mathbf{hurts}(x)$ if

$\mathcal{M}, \beta_x^{o_1} \models \mathbf{hurts}(x)$ **iff**

Semantic Rule

Information from model $(\mathcal{D}, \delta, \mathcal{I})$

$\mathcal{D}^{\mathbf{Weapon}} = \{o_1, o_2, o_4\}$

Semantics of First-Order Logic: Evaluation Example

Let β be arbitrary.

$\mathcal{M}, \beta \models \exists\, x : \textbf{Weapon} \,.\, \textbf{hurts}(x)$ **iff**

There exists $d \in \mathcal{D}^{\textbf{Weapon}}$ **such that** $\mathcal{M}, \beta_x^d \models \textbf{hurts}(x)$ **if**

$\mathcal{M}, \beta_x^{o_1} \models \textbf{hurts}(x)$ **iff**

$val_{\mathcal{M}, \beta_x^{o_1}}(x) \in \mathcal{I}(\textbf{hurts})$

Semantic Rule

$\mathcal{M}, \beta \models p(t_1, \ldots, t_r)$ **iff** $(val_{\mathcal{M},\beta}(t_1), \ldots, val_{\mathcal{M},\beta}(t_r)) \in \mathcal{I}(p)$

Information from model $(\mathcal{D}, \delta, \mathcal{I})$

Semantics of First-Order Logic: Evaluation Example

Let β be arbitrary.

$\mathcal{M}, \beta \models \exists x : \mathbf{Weapon}\,.\,\mathbf{hurts}(x)$ **iff**

There exists $d \in \mathcal{D}^{\mathbf{Weapon}}$ **such that** $\mathcal{M}, \beta_x^d \models \mathbf{hurts}(x)$ **if**

$\mathcal{M}, \beta_x^{o_1} \models \mathbf{hurts}(x)$ **iff**

$val_{\mathcal{M},\beta_x^{o_1}}(x) \in \mathcal{I}(\mathbf{hurts})$

since $val_{\mathcal{M},\beta_x^{o_1}}(x) = \beta_x^{o_1}(x) = o_1$ **iff**

Semantic Rule

$$val_{\mathcal{M},\beta}(x) = \beta(x), \qquad \beta_y^d(x) := \begin{cases} \beta(x) & x \neq y \\ d & x = y \end{cases}$$

Information from model $(\mathcal{D}, \delta, \mathcal{I})$

Semantics of First-Order Logic: Evaluation Example

Let β be arbitrary.

$\mathcal{M}, \beta \models \exists x : \mathbf{Weapon} \,.\, \mathbf{hurts}(x)$ **iff**

There exists $d \in \mathcal{D}^{\mathbf{Weapon}}$ **such that** $\mathcal{M}, \beta_x^d \models \mathbf{hurts}(x)$ **if**

$\mathcal{M}, \beta_x^{o_1} \models \mathbf{hurts}(x)$ **iff**

$val_{\mathcal{M}, \beta_x^{o_1}}(x) \in \mathcal{I}(\mathbf{hurts})$

since $val_{\mathcal{M}, \beta_x^{o_1}}(x) = \beta_x^{o_1}(x) = o_1$ **iff**

$o_1 \in \mathcal{I}(\mathbf{hurts}) = \{o_1, o_2, o_4\}$

Semantic Rule

Information from model $(\mathcal{D}, \delta, \mathcal{I})$

$I(\mathbf{hurts}) = \{o_1, o_2, o_4\}$

Semantics of First-Order Logic: Evaluation Example

Let β be arbitrary.

$\mathcal{M}, \beta \models \exists\, x : \mathbf{Weapon} \,.\, \mathbf{hurts}(x)$ **iff**

There exists $d \in \mathcal{D}^{\mathbf{Weapon}}$ **such that** $\mathcal{M}, \beta_x^d \models \mathbf{hurts}(x)$ **if**

$\mathcal{M}, \beta_x^{o_1} \models \mathbf{hurts}(x)$ **iff**

$val_{\mathcal{M}, \beta_x^{o_1}}(x) \in \mathcal{I}(\mathbf{hurts})$

since $val_{\mathcal{M}, \beta_x^{o_1}}(x) = \beta_x^{o_1}(x) = o_1$ **iff**

$o_1 \in \mathcal{I}(\mathbf{hurts}) = \{o_1, o_2, o_4\}$ **ok!**

Semantic Rule

Information from model $(\mathcal{D}, \delta, \mathcal{I})$

First-Order Semantic Notions

Satisfiability, truth, and validity

$\mathcal{M}, \beta \models \phi$				(ϕ **is satisfiable**)
$\mathcal{M} \models \phi$	**iff**	**for all** β :	$\mathcal{M}, \beta \models \phi$	(ϕ **is true in** $\mathcal{M}$)
$\models \phi$	**iff**	**for all** $\mathcal{M}$:	$\mathcal{M} \models \phi$	(ϕ **is valid**)

Formula containing only variables in scope of a quantifier is closed

Closed formulas that are satisfiable are also true: only one notion

From now on only *closed* formulas are considered.

First-Order Logic Example

Types $\mathcal{T}_d = \{\textbf{Stick}, \textbf{Stone}, \textbf{Flower}\}, \quad \mathcal{T}_a = \{\textbf{Weapon}, \textbf{Any}\}$

$\textbf{Stick}, \textbf{Stone} \sqsubseteq \textbf{Weapon} \sqsubseteq \textbf{Any}, \ \textbf{Flower} \sqsubseteq \textbf{Any}$

Predicates $\mathbf{P} = \{\textbf{hurts} : \ \textbf{Any}\}$

Variables $\mathbf{V} = \{x : \textbf{Weapon}, \ y : \textbf{Flower}\}$

First-Order Logic Example

Types $\mathcal{T}_d = \{\textbf{Stick}, \textbf{Stone}, \textbf{Flower}\}, \quad \mathcal{T}_a = \{\textbf{Weapon}, \textbf{Any}\}$

$\textbf{Stick}, \textbf{Stone} \sqsubseteq \textbf{Weapon} \sqsubseteq \textbf{Any}, \ \textbf{Flower} \sqsubseteq \textbf{Any}$

Predicates $\mathbf{P} = \{\textbf{hurts} : \ \textbf{Any}\}$

Variables $\mathbf{V} = \{x : \textbf{Weapon}, \ y : \textbf{Flower}\}$

$$\forall x : \textbf{Weapon} \,.\, \textbf{hurts}(x) \quad \textbf{\&} \quad \forall y : \textbf{Flower} \,.\, \textbf{!hurts}(y)$$

Satisfiable? True? Valid?

First-Order Logic Example

Types $\mathcal{T}_d = \{\textbf{Stick}, \textbf{Stone}, \textbf{Flower}\}, \quad \mathcal{T}_a = \{\textbf{Weapon}, \textbf{Any}\}$

$\textbf{Stick}, \textbf{Stone} \sqsubseteq \textbf{Weapon} \sqsubseteq \textbf{Any}, \ \textbf{Flower} \sqsubseteq \textbf{Any}$

Predicates $\mathbf{P} = \{\textbf{hurts} : \ \textbf{Any}\}$

Variables $\mathbf{V} = \{x : \textbf{Weapon}, \ y : \textbf{Flower}\}$

$$\forall x : \textbf{Weapon} \,.\, \textbf{hurts}(x) \quad \textbf{\&} \quad \forall y : \textbf{Flower} \,.\, \textbf{!hurts}(y)$$

Satisfiable? True? Valid?

Model:

$\mathcal{D} = \{o_1, o_2\}$**,** $\quad \delta(o_1) = \textbf{Stone,} \quad \delta(o_2) = \textbf{Flower}$

$\mathcal{I}(\textbf{hurts}) = \{o_1\}$

First-Order Logic Example

Types $\mathcal{T}_d = \{\textbf{Stick}, \textbf{Stone}, \textbf{Flower}\}, \quad \mathcal{T}_a = \{\textbf{Weapon}, \textbf{Any}\}$

$\textbf{Stick}, \textbf{Stone} \sqsubseteq \textbf{Weapon} \sqsubseteq \textbf{Any}, \; \textbf{Flower} \sqsubseteq \textbf{Any}$

Predicates $\mathbf{P} = \{\textbf{hurts} : \; \textbf{Any}\}$

Variables $\mathbf{V} = \{x : \textbf{Weapon}, \, y : \textbf{Flower}\}$

$$\forall x : \textbf{Weapon} \,.\, \textbf{hurts}(x) \quad \textbf{\&} \quad \forall y : \textbf{Flower} \,.\, \textbf{!hurts}(y)$$

Satisfiable? True? Valid?

Counter-model:

$\mathcal{D} = \{o_1, o_2\}$**,** $\quad \delta(o_1) = \textbf{Stone,} \quad \delta(o_2) = \textbf{Flower}$

$\mathcal{I}(\textbf{hurts}) = \{\}$

First-Order Logic Example

Types $\mathcal{T}_d = \{\textbf{Stick}, \textbf{Stone}, \textbf{Flower}\}, \quad \mathcal{T}_a = \{\textbf{Weapon}, \textbf{Any}\}$

$\textbf{Stick}, \textbf{Stone} \sqsubseteq \textbf{Weapon} \sqsubseteq \textbf{Any}, \; \textbf{Flower} \sqsubseteq \textbf{Any}$

Predicates $\mathbf{P} = \{\textbf{hurts} : \; \textbf{Any}\}$

Variables $\mathbf{V} = \{x : \textbf{Weapon}, \; y : \textbf{Flower}\}$

$$\forall x : \textbf{Weapon} \,.\, \textbf{hurts}(x) \quad \textbf{\&} \quad \forall y : \textbf{Flower} \,.\, \textbf{!hurts}(y)$$

Satisfiable? True? Valid?

Another Counter-model:

$\mathcal{D} = \{o_1, \, o_2, o_3\}$**,** $\quad \delta(o_1) = \textbf{Stone,} \quad \delta(o_2) = \delta(o_3) = \textbf{Flower}$

$\mathcal{I}(\textbf{hurts}) = \{o_1, o_3\}$

Untyped First-Order Logic

Standard FOL (as in most logic textbooks is untyped [single typed])

Obtained as special case of typed signature:

$\mathcal{T}_d = \{\top\}$, $\mathcal{T}_a = \{\bot\}$

Hence, $\mathcal{D} = \mathcal{D}^\top \neq \emptyset$, $\delta(d) = \top$ **for all** $d \in \mathcal{D}$

All variables, predicate and function symbols declared on $\top$

Don't need type information of variables (omit)

Only arity in signature of function/predicate symbols matters

Untyped First-Order Logic

Standard FOL (as in most logic textbooks is untyped [single typed])

Obtained as special case of typed signature:

$\mathcal{T}_d = \{\top\}$, $\mathcal{T}_a = \{\bot\}$

Hence, $\mathcal{D} = \mathcal{D}^\top \neq \emptyset$, $\delta(d) = \top$ **for all** $d \in \mathcal{D}$

All variables, predicate and function symbols declared on $\top$

Don't need type information of variables (omit)

Only arity in signature of function/predicate symbols matters

Example: $\mathbf{P} = \{\mathbf{person}/1, \mathbf{happy}/1\}$, $\mathbf{F} = \{\mathbf{pat}/0\}$

Untyped First-Order Logic

Standard FOL (as in most logic textbooks is untyped [single typed])

Obtained as special case of typed signature:

$\mathcal{T}_d = \{\top\}$, $\mathcal{T}_a = \{\bot\}$

Hence, $\mathcal{D} = \mathcal{D}^\top \neq \emptyset$, $\delta(d) = \top$ **for all** $d \in \mathcal{D}$

All variables, predicate and function symbols declared on $\top$

Don't need type information of variables (omit)

Only arity in signature of function/predicate symbols matters

Example: $\mathbf{P} = \{\mathbf{person}/1, \mathbf{happy}/1\}$, $\mathbf{F} = \{\mathbf{pat}/0\}$

$\forall x \,.\, (\mathbf{person}(x)$ **->** $\mathbf{happy}(x))$ ALL PERSONS ARE HAPPY

Untyped First-Order Logic

Standard FOL (as in most logic textbooks is untyped [single typed])

Obtained as special case of typed signature:

$\mathcal{T}_d = \{\top\}$, $\mathcal{T}_a = \{\bot\}$

Hence, $\mathcal{D} = \mathcal{D}^{\top} \neq \emptyset$, $\delta(d) = \top$ **for all** $d \in \mathcal{D}$

All variables, predicate and function symbols declared on $\top$

Don't need type information of variables (omit)

Only arity in signature of function/predicate symbols matters

Example: $\mathbf{P} = \{\mathbf{person}/1, \mathbf{happy}/1\}$, $\mathbf{F} = \{\mathbf{pat}/0\}$

$\forall x\,.\,(\mathbf{person}(x)$ **->** $\mathbf{happy}(x))$ — ALL PERSONS ARE HAPPY

$\mathbf{person}(\mathbf{pat})$ — PAT IS A PERSON

Untyped First-Order Logic

Standard FOL (as in most logic textbooks is untyped [single typed])

Obtained as special case of typed signature:

$\mathcal{T}_d = \{\top\}$, $\mathcal{T}_a = \{\bot\}$

Hence, $\mathcal{D} = \mathcal{D}^\top \neq \emptyset$, $\delta(d) = \top$ **for all** $d \in \mathcal{D}$

All variables, predicate and function symbols declared on $\top$

Don't need type information of variables (omit)

Only arity in signature of function/predicate symbols matters

Example: $\mathbf{P} = \{\mathbf{person}/1, \mathbf{happy}/1\}$, $\mathbf{F} = \{\mathbf{pat}/0\}$

$\forall x\,.\,(\mathbf{person}(x)$ **->** $\mathbf{happy}(x))$	ALL PERSONS ARE HAPPY
$\mathbf{person}(\mathbf{pat})$	PAT IS A PERSON
$\mathbf{happy}(\mathbf{pat})$	PAT IS HAPPY

Types and Symbols with Fixed Interpretation

Certain symbols should have “standard” meaning in all interpretations

So far: $\doteq$, $\sqsubseteq_z$, (z)

For certain types we also fix domain and dynamic typing:

$$\mathcal{D}^{\texttt{int}} = \{d \in \mathcal{D} \mid \delta(d) = \texttt{int}\} = \mathbb{Z}$$

These types appear between $\perp$ and $\top$, uncomparable to others

Examples of types, function/predicate symbols with fixed meaning

$\mathcal{I}(\mathbf{17})$ **should be always** 17**, not e.g.** $towel$

`int` **KeY can switch between JAVA 32-bit integers and** $\mathbb{Z}$
but in FOL always math integers $\mathcal{I}(\texttt{+}) = +_{\mathbb{Z}}$, $\mathcal{I}(\texttt{*}) = *_{\mathbb{Z}}$**, ...**

`boolean`

Some Predefined Symbols in KeY FO Logic

Types

`int`, `short`, `byte`, `boolean` with standard meaning

All classes of current UML context diagram and Null

If T is one of these types then also $Set(T)$, $Bag(T)$, $Sequence(T)$

Predicates on integer types with standard meaning

>, <, >=, <=, . . . (infix)

Functions and Constants with standard meaning

`+`, `-`, `div`, `mod`, `0`, `1`, `...`

TRUE, FALSE

Notation for quantifiers, variables declared at quantifier symbol

```
\forall Type Variable; ScopeFormula
```

First-Order Problems in KeY Syntax: .key

```
\sorts { // types are called 'sorts'
   person; // one declaration per line, end with ';'
}
\functions { // ResultType FctSymbol(ParType,..,ParType)
   int age(person); // 'int' predefined type
}
\predicates { // PredSymbol(ParType,..,ParType)
   parent(person,person);
}
\problem { // Goal formula
   \forall person son; \forall person father;
      (parent(father,son) -> age(father) > age(son)) }
```

Contents

- Overview of KeY
- UML and its semantics
- Introduction to OCL
- Specifying requirements with OCL
- Modelling of Systems with Formal Semantics
- Propositional & First-order logic, sequent calculus
- OCL to Logic, horizontal proof obligations, using KeY
- Dynamic logic, proving program correctness
- Java Card DL
- Vertical proof obligations, using KeY
- Wrap-up, trends

Sequent Calculus for FOL

	left side, antecedent	right side, succedent

- $[t/t']\,\phi$ **is result of replacing each occurrence of** t **in** ϕ **with** t'
- $s^z, t^{z'}$ **and** t **are arbitrary variable free terms**
- x **and** s^z **have static type** z **and** $t^{z'}$ **has static type** $z' \sqsubseteq z$
- c^z **new constant of type** z **(does not occur in current proof branch)**
- **Equations can be reversed (by symmetry of equality)**

Sequent Calculus for FOL

	left side, antecedent	right side, succedent
$\forall$	$\dfrac{\Gamma,\ \forall x\,.\,\phi,\ \left[x/t^{z'}\right]\phi \ \textbf{==>}\ \Delta}{\Gamma, \forall x\,.\,\phi \ \textbf{==>}\ \Delta}$	$\dfrac{\Gamma \ \textbf{==>}\ [x/c^{z}]\,\phi, \Delta}{\Gamma \ \textbf{==>}\ \forall x\,.\,\phi,\ \Delta}$

- $[t/t']\,\phi$ **is result of replacing each occurrence of t in ϕ with t'**
- $s^z, t^{z'}$ **and t are arbitrary variable free terms**
- x **and s^z have static type z and $t^{z'}$ has static type $z' \sqsubseteq z$**
- c^z **new constant of type z (does not occur in current proof branch)**
- **Equations can be reversed (by symmetry of equality)**

Sequent Calculus for FOL

	left side, antecedent	right side, succedent
$\forall$	$\dfrac{\Gamma,\ \forall x\,.\,\phi,\ \left[x/t^{z'}\right]\phi \;\texttt{==>}\; \Delta}{\Gamma, \forall x\,.\,\phi \;\texttt{==>}\; \Delta}$	$\dfrac{\Gamma \;\texttt{==>}\; [x/c^{z}]\,\phi, \Delta}{\Gamma \;\texttt{==>}\; \forall x\,.\,\phi,\ \Delta}$
$\exists$	$\dfrac{\Gamma,\ [x/c^{z}]\,\phi \;\texttt{==>}\; \Delta}{\Gamma,\ \exists x\,.\,\phi \;\texttt{==>}\; \Delta}$	$\dfrac{\Gamma \;\texttt{==>}\; \left[x/t^{z'}\right]\phi,\ \exists x\,.\,\phi,\ \Delta}{\Gamma \;\texttt{==>}\; \exists x\,.\,\phi, \Delta}$

- $[t/t']\,\phi$ **is result of replacing each occurrence of** t **in** ϕ **with** t'
- $s^{z}, t^{z'}$ **and** t **are arbitrary variable free terms**
- x **and** s^{z} **have static type** z **and** $t^{z'}$ **has static type** $z' \sqsubseteq z$
- c^{z} **new constant of type** z **(does not occur in current proof branch)**
- **Equations can be reversed (by symmetry of equality)**

Sequent Calculus for FOL

	left side, antecedent	right side, succedent
$\forall$	$\dfrac{\Gamma,\ \forall x\,.\,\phi,\ \left[x/t^{z'}\right]\phi \text{ ==> } \Delta}{\Gamma, \forall x\,.\,\phi \text{ ==> } \Delta}$	$\dfrac{\Gamma \text{ ==> } [x/c^z]\,\phi, \Delta}{\Gamma \text{ ==> } \forall x\,.\,\phi,\ \Delta}$
$\exists$	$\dfrac{\Gamma,\ [x/c^z]\,\phi \text{ ==> } \Delta}{\Gamma,\ \exists x\,.\,\phi \text{ ==> } \Delta}$	$\dfrac{\Gamma \text{ ==> } \left[x/t^{z'}\right]\phi,\ \exists x\,.\,\phi,\ \Delta}{\Gamma \text{ ==> } \exists x\,.\,\phi, \Delta}$
$\doteq$	$\dfrac{\Gamma, s^z \doteq t^{z'}, \left[s^z/t^{z'}\right]\psi \text{ ==> } \left[s^z/t^{z'}\right]\phi, \Delta}{\Gamma, s^z \doteq t^{z'}, \psi \text{ ==> } \phi, \Delta}$	$\dfrac{}{\Gamma \text{ ==> } t \doteq t, \Delta}$

- $[t/t']\,\phi$ **is result of replacing each occurrence of** t **in** ϕ **with** t'
- $s^z, t^{z'}$ **and** t **are arbitrary variable free terms**
- x **and** s^z **have static type** z **and** $t^{z'}$ **has static type** $z' \sqsubseteq z$
- c^z **new constant of type** z **(does not occur in current proof branch)**
- **Equations can be reversed (by symmetry of equality)**

A Simple Proof (Exercises p3.key)

$$\exists x . \forall y . p(x, y) \text{ ==> } \forall y . \exists x . p(x, y)$$

Let static type of x and y be $\top$

A Simple Proof (Exercises p3.key)

$$\forall y \,.\, p(c, y) \text{ ==> } \forall y \,.\, \exists x \,.\, p(x, y)$$

$$\exists x \,.\, \forall y \,.\, p(x, y) \text{ ==> } \forall y \,.\, \exists x \,.\, p(x, y)$$

ex left: substitute new constant c of type $\top$ for x

A Simple Proof (Exercises p3.key)

$\forall y \,.\, p(c, y)$ **==>** $\exists x \,.\, p(x, d)$

$\forall y \,.\, p(c, y)$ **==>** $\forall y \,.\, \exists x \,.\, p(x, y)$

$\exists x \,.\, \forall y \,.\, p(x, y)$ **==>** $\forall y \,.\, \exists x \,.\, p(x, y)$

all right: substitute new constant d of type $\top$ for y

A Simple Proof (Exercises p3.key)

$$\overline{}$$

$$\overline{p(c, d), \forall y \,.\, p(c, y) \text{ ==> } \exists x \,.\, p(x, d)}$$

$$\overline{\forall y \,.\, p(c, y) \text{ ==> } \exists x \,.\, p(x, d)}$$

$$\overline{\forall y \,.\, p(c, y) \text{ ==> } \forall y \,.\, \exists x \,.\, p(x, y)}$$

$$\exists x \,.\, \forall y \,.\, p(x, y) \text{ ==> } \forall y \,.\, \exists x \,.\, p(x, y)$$

all left: free to substitute any term of type $\top$ for y, choose d

A Simple Proof (Exercises p3.key)

$$p(c, d) \quad \texttt{==>} \quad \exists\, x\, .\, p(x, d)$$

$$\forall\, y\, .\, p(c, y) \quad \texttt{==>} \quad \exists\, x\, .\, p(x, d)$$

$$\forall\, y\, .\, p(c, y) \quad \texttt{==>} \quad \forall\, y\, .\, \exists\, x\, .\, p(x, y)$$

$$\exists\, x\, .\, \forall\, y\, .\, p(x, y) \quad \texttt{==>} \quad \forall\, y\, .\, \exists\, x\, .\, p(x, y)$$

all left not needed anymore (hide)

A Simple Proof (Exercises p3.key)

$$\dfrac{\dfrac{\dfrac{\dfrac{p(c,d) \quad \textbf{==>} \quad p(c,d),\ \exists x\,.\,p(x,y)}{p(c,d) \quad \textbf{==>} \quad \exists x\,.\,p(x,d)}}{\forall y\,.\,p(c,y) \quad \textbf{==>} \quad \exists x\,.\,p(x,d)}}{\forall y\,.\,p(c,y) \quad \textbf{==>} \quad \forall y\,.\,\exists x\,.\,p(x,y)}}{\exists x\,.\,\forall y\,.\,p(x,y) \quad \textbf{==>} \quad \forall y\,.\,\exists x\,.\,p(x,y)}$$

ex right: free to substitute any term of type $\top$ for x, choose c

A Simple Proof (Exercises p3.key)

$p(c, d)$	==> $p(c, d)$
$p(c, d)$	==> $\exists x \,.\, p(x, d)$
$\forall y \,.\, p(c, y)$	==> $\exists x \,.\, p(x, d)$
$\forall y \,.\, p(c, y)$	==> $\forall y \,.\, \exists x \,.\, p(x, y)$
$\exists x \,.\, \forall y \,.\, p(x, y)$	==> $\forall y \,.\, \exists x \,.\, p(x, y)$

ex right not needed anymore (hide)

A Simple Proof (Exercises p3.key)

$$*$$

$p(c, d)$ **==>** $p(c, d)$

$p(c, d)$ **==>** $\exists x \,.\, p(x, d)$

$\forall y \,.\, p(c, y)$ **==>** $\exists x \,.\, p(x, d)$

$\forall y \,.\, p(c, y)$ **==>** $\forall y \,.\, \exists x \,.\, p(x, y)$

$\exists x \,.\, \forall y \,.\, p(x, y)$ **==>** $\forall y \,.\, \exists x \,.\, p(x, y)$

Close

Rules for Type Casts and Type Predicates

- **Type predicate formulas** $t \sqsubseteq z$
  true iff dynamic type $val_{\mathcal{M}}(t)$ **is subtype of** z

- **Type cast terms** $(z)t$
  evaluates to $val_{\mathcal{M}}(t)$ **if cast succeeds, arb. element otherwise**

Rules for Type Casts and Type Predicates

- **Type predicate formulas** $t \sqsubseteq z$
  true iff dynamic type $val_{\mathcal{M}}(t)$ **is subtype of** z
- **Type cast terms** $(z)t$
  evaluates to $val_{\mathcal{M}}(t)$ **if cast succeeds, arb. element otherwise**

Typical rule:

Rules for Type Casts and Type Predicates

- **Type predicate formulas** $t \sqsubseteq z$
  true iff dynamic type $val_{\mathcal{M}}(t)$ **is subtype of** z
- **Type cast terms** $(z)t$
  evaluates to $val_{\mathcal{M}}(t)$ **if cast succeeds, arb. element otherwise**

Typical rule:

The dynamic type of a term must be typeable to its static type

$$\textsc{typeStatic}\ \frac{\Gamma, t \sqsubseteq z \ \texttt{==>}\ \Delta}{\Gamma\ \texttt{==>}\ \Delta} \quad z \text{ static (declared) type of } t$$

Expresses type-safety of typed first-order logic

Rules for Type Casts and Type Predicates

- **Type predicate formulas** $t \sqsubseteq z$
  true iff dynamic type $val_{\mathcal{M}}(t)$ **is subtype of** z
- **Type cast terms** $(z)t$
  evaluates to $val_{\mathcal{M}}(t)$ **if cast succeeds, arb. element otherwise**

Typical rule:

The dynamic type of a term must be typeable to its static type

$$\textsc{typeStatic}\ \frac{\Gamma, t \sqsubseteq z \ \texttt{==>}\ \Delta}{\Gamma\ \texttt{==>}\ \Delta} \quad z \text{ static (declared) type of } t$$

Expresses type-safety of typed first-order logic

KeY first-order strategy applies suitable typing rules automatically

Sequent Proofs: Important Issues

- Rules are applied to top-most connective/quantifier
- exLeft and allRight substitute new constant
- exRight and allLeft allow to substitute any variable-free term
- Formulas that are not needed in remaining proof may be hidden
- All branches must be closed with axiom
- There are many different possible proofs for a valid sequent
- KeY FO strategy applies all but exRight and allLeft automatically

Another Proof Example

Types $\mathcal{T} = \{\bot, \top\}$

Predicates **PSym** $= \{p\}, \qquad p : \top, \top$

Functions **FSym** $= \{\}$

$$(\exists x . \exists y . p(x, y) \text{ \& } \forall x . !p(x, x)) \quad \text{->} \quad \exists x . \exists y . (!x \doteq y)$$

Intuitive Meaning? Satisfiable? True? Valid?

Demo

`oclFol/rel.key`