

# Software Engineering using Formal Methods

## First-Order Logic

Wolfgang Ahrendt and  
*Bart van Delft*

30th September 2014

# Install the KeY-Tool...

KeY used in Tuesday's exercise

**Requires:** Java  $\geq$  5

Follow instructions on course page, under:

⇒ [Links, Papers, and Software / Tools](#)

We recommend using **Java Web Start**:

- ▶ Start KeY with two clicks  
(you need to trust our self-signed certificate)
- ▶ Java Web Start installed with standard JDK/JRE
- ▶ Usually browsers know filetype.  
Otherwise open KeY.jnlp with javaws.

If you want to install KeY locally instead, download from [www.key-project.org](http://www.key-project.org). For this course, install version 1.6.x.

# Motivation for Introducing First-Order Logic

1) we specify JAVA programs with **Java Modeling Language (JML)**

## JML combines

- ▶ JAVA expressions
- ▶ **First-Order Logic (FOL)**

2) we verify JAVA programs using **Dynamic Logic**

## Dynamic Logic combines

- ▶ **First-Order Logic (FOL)**
- ▶ JAVA programs

we introduce:

- ▶ FOL as a language
- ▶ calculus for proving FOL formulas
- ▶ KeY system as propositional, and first-order, prover (for now)
- ▶ (formal semantics: if time)

# Part I

## The Language of FOL

# First-Order Logic: Signature

## Signature

A first-order signature  $\Sigma$  consists of

- ▶ a set  $T_\Sigma$  of types
- ▶ a set  $F_\Sigma$  of function symbols
- ▶ a set  $P_\Sigma$  of predicate symbols
- ▶ a typing  $\alpha_\Sigma$

# First-Order Logic: Signature

## Signature

A first-order signature  $\Sigma$  consists of

- ▶ a set  $T_\Sigma$  of types
- ▶ a set  $F_\Sigma$  of function symbols
- ▶ a set  $P_\Sigma$  of predicate symbols
- ▶ a typing  $\alpha_\Sigma$

intuitively, the typing  $\alpha_\Sigma$  determines

- ▶ for each function and predicate symbol:
  - ▶ its arity, i.e., number of arguments
  - ▶ its argument types
- ▶ for each function symbol its result type.

formally:

- ▶  $\alpha_\Sigma(p) \in T_\Sigma^*$  for all  $p \in P_\Sigma$  (arity of  $p$  is  $|\alpha_\Sigma(p)|$ )
- ▶  $\alpha_\Sigma(f) \in T_\Sigma^* \times T_\Sigma$  for all  $f \in F_\Sigma$  (arity of  $f$  is  $|\alpha_\Sigma(f)| - 1$ )

## Example Signature 1 + Constants

$$T_{\Sigma_1} = \{\text{int}\},$$

$$F_{\Sigma_1} = \{+, -\} \cup \{\dots, -2, -1, 0, 1, 2, \dots\},$$

$$P_{\Sigma_1} = \{<\}$$

$$\alpha_{\Sigma_1}(<) = (\text{int}, \text{int})$$

$$\alpha_{\Sigma_1}(+) = \alpha_{\Sigma_1}(-) = (\text{int}, \text{int}, \text{int})$$

$$\alpha_{\Sigma_1}(0) = \alpha_{\Sigma_1}(1) = \alpha_{\Sigma_1}(-1) = \dots = (\text{int})$$

### Constant Symbols

A function symbol  $f$  with  $|\alpha_{\Sigma_1}(f)| = 1$  (i.e., with arity 0) is called *constant symbol*.

here, the constant symbols are:  $\dots, -2, -1, 0, 1, 2, \dots$



# Syntax of First-Order Logic: Signature Cont'd

## Type declaration of signature symbols

- ▶ Write  $\tau x$ ; to declare variable  $x$  of type  $\tau$
- ▶ Write  $p(\tau_1, \dots, \tau_r)$ ; for  $\alpha(p) = (\tau_1, \dots, \tau_r)$
- ▶ Write  $\tau f(\tau_1, \dots, \tau_r)$ ; for  $\alpha(f) = (\tau_1, \dots, \tau_r, \tau)$

$r = 0$  is allowed, then write  $f$  instead of  $f()$ , etc.

## Example

**Variables**     `integerArray a;   int i;`

**Predicate Symbols**     `isEmpty(List);   alertOn;`

**Function Symbols**     `int arrayLookup(int);   Object o;`

# Example Signature 1 + Notation

typing of Signature 1:

$$\alpha_{\Sigma_1}(<) = (\text{int}, \text{int})$$

$$\alpha_{\Sigma_1}(+) = \alpha_{\Sigma_1}(-) = (\text{int}, \text{int}, \text{int})$$

$$\alpha_{\Sigma_1}(0) = \alpha_{\Sigma_1}(1) = \alpha_{\Sigma_1}(-1) = \dots = (\text{int})$$

can alternatively be written as:

```
<(int,int);
```

```
int +(int,int);
```

```
int 0;  int 1;  int -1;  ...
```

## Example Signature 2

$$\begin{aligned}T_{\Sigma_2} &= \{\text{int}, \text{LinkedList}\}, \\F_{\Sigma_2} &= \{\text{null}, \text{new}, \text{elem}, \text{next}\} \cup \{\dots, -2, -1, 0, 1, 2, \dots\} \\P_{\Sigma_2} &= \{\}\end{aligned}$$

intuitively, elem and next model fields of LinkedList objects

type declarations:

```
LinkedList null;  
LinkedList new(int,LinkedList);  
int elem(LinkedList);  
LinkedList next(LinkedList);
```

and as before:

```
int 0; int 1; int -1; ...
```

# First-Order Terms

We assume a set  $V$  of variables ( $V \cap (F_\Sigma \cup P_\Sigma) = \emptyset$ ).  
Each  $v \in V$  has a unique type  $\alpha_\Sigma(v) \in T_\Sigma$ .

Terms are defined recursively:

## Terms

A first-order term of type  $\tau \in T_\Sigma$

- ▶ is either a variable of type  $\tau$ , or
- ▶ has the form  $f(t_1, \dots, t_n)$ ,  
where  $f \in F_\Sigma$  has result type  $\tau$ , and each  $t_i$  is term of the correct type, following the typing  $\alpha_\Sigma$  of  $f$ .

If  $f$  is a constant symbol, the term is written  $f$ , instead of  $f()$ .

# Terms over Signature 1

example terms over  $\Sigma_1$ :

(assume variables `int v1`; `int v2`;) )

# Terms over Signature 1

example terms over  $\Sigma_1$ :

(assume variables  $\text{int } v_1; \text{ int } v_2;$ )

- ▶  $-7$
- ▶  $+(-2, 99)$
- ▶  $-(7, 8)$
- ▶  $+(-(7, 8), 1)$
- ▶  $+(-(v_1, 8), v_2)$

# Terms over Signature 1

example terms over  $\Sigma_1$ :

(assume variables  $\text{int } v_1; \text{ int } v_2;$ )

- ▶  $-7$
- ▶  $+(-2, 99)$
- ▶  $-(7, 8)$
- ▶  $+(-(7, 8), 1)$
- ▶  $+(-(v_1, 8), v_2)$

some variants of FOL allow infix notation of functions:

- ▶  $-2 + 99$
- ▶  $7 - 8$
- ▶  $(7 - 8) + 1$
- ▶  $(v_1 - 8) + v_2$

## Terms over Signature 2

example terms over  $\Sigma_2$ :

(assume variables `LinkedList`  $o$ ; `int`  $v$ ;) )



## Terms over Signature 2

example terms over  $\Sigma_2$ :

(assume variables `LinkedList o`; `int v`;) )

- ▶ `-7`
- ▶ `null`
- ▶ `new(v, null)`
- ▶ `elem(new(13, null))`
- ▶ `next(next(o))`

## Terms over Signature 2

example terms over  $\Sigma_2$ :

(assume variables `LinkedList` `o`; `int` `v`;

- ▶ `-7`
- ▶ `null`
- ▶ `new(v, null)`
- ▶ `elem(new(13, null))`
- ▶ `next(next(o))`

for first-order functions modeling object fields,  
we allow dotted postfix notation:

- ▶ `new(13, null).elem`
- ▶ `o.next.next`

## Atomic Formulas

Given a signature  $\Sigma$ .

An atomic formula has either of the forms

- ▶ *true*
- ▶ *false*
- ▶  $t_1 = t_2$  (“equality”),  
where  $t_1$  and  $t_2$  are first-order terms of the same type.
- ▶  $p(t_1, \dots, t_n)$  (“predicate”),  
where  $p \in P_\Sigma$ , and each  $t_i$  is term of the correct type,  
following the typing  $\alpha_\Sigma$  of  $p$ .

# Atomic Formulas over Signature 1

example formulas over  $\Sigma_1$ :  
(assume variable `int v`;

- ▶  $7 = 8$
- ▶  $7 < 8$
- ▶  $-2 - v < 99$
- ▶  $v < (v + 1)$

## Atomic Formulas over Signature 2

example formulas over  $\Sigma_2$ :

(assume variables `LinkedList o`; `int v`;) )

- ▶ `new(v, null) = null`
- ▶ `elem(new(13, null)) = 13`
- ▶ `next(new(13, null)) = null`
- ▶ `next(next(o)) = o`

# First-order Formulas

## Formulas

- ▶ each atomic formula is a formula
- ▶ with  $\phi$  and  $\psi$  formulas,  $x$  a variable, and  $\tau$  a type, the following are also formulas:
  - ▶  $\neg\phi$  (“not  $\phi$ ”)
  - ▶  $\phi \wedge \psi$  (“ $\phi$  and  $\psi$ ”)
  - ▶  $\phi \vee \psi$  (“ $\phi$  or  $\psi$ ”)
  - ▶  $\phi \rightarrow \psi$  (“ $\phi$  implies  $\psi$ ”)
  - ▶  $\phi \leftrightarrow \psi$  (“ $\phi$  is equivalent to  $\psi$ ”)
  - ▶  $\forall \tau x; \phi$  (“for all  $x$  of type  $\tau$  holds  $\phi$ ”)
  - ▶  $\exists \tau x; \phi$  (“there exists an  $x$  of type  $\tau$  such that  $\phi$ ”)

In  $\forall \tau x; \phi$  and  $\exists \tau x; \phi$  the variable  $x$  is ‘bound’ (i.e., ‘not free’).  
Formulas with no free variable are ‘closed’.

# First-order Formulas: Examples

(signatures/types left out here)

**Example (There are at least two elements)**



# First-order Formulas: Examples

(signatures/types left out here)

**Example (There are at least two elements)**

$$\exists x, y; \neg(x = y)$$



# First-order Formulas: Examples

(signatures/types left out here)

**Example (Strict partial order)**



# First-order Formulas: Examples

(signatures/types left out here)

## Example (Strict partial order)

Irreflexivity  $\forall x; \neg(x < x)$

Asymmetry  $\forall x; \forall y; (x < y \rightarrow \neg(y < x))$

Transitivity  $\forall x; \forall y; \forall z;$   
 $(x < y \wedge y < z \rightarrow x < z)$

# First-order Formulas: Examples

(signatures/types left out here)

## Example (Strict partial order)

Irreflexivity  $\forall x; \neg(x < x)$

Asymmetry  $\forall x; \forall y; (x < y \rightarrow \neg(y < x))$

Transitivity  $\forall x; \forall y; \forall z;$   
 $(x < y \wedge y < z \rightarrow x < z)$

(is any of the three formulas redundant?)

# Semantics (briefly here, more thorough later)

## Domain

A domain  $\mathcal{D}$  is a set of elements which are (potentially) the *meaning* of terms and variables.

## Interpretation

An interpretation  $\mathcal{I}$  (over  $\mathcal{D}$ ) assigns *meaning* to the symbols in  $F_\Sigma \cup P_\Sigma$  (assigning functions to function symbols, relations to predicate symbols).

## Valuation

In a given  $\mathcal{D}$  and  $\mathcal{I}$ , a closed formula evaluates to either  $T$  or  $F$ .

## Validity

A closed formula is **valid** if it evaluates to  $T$  in **all**  $\mathcal{D}$  and  $\mathcal{I}$ .

In the context of specification/verification of programs:  
each  $(\mathcal{D}, \mathcal{I})$  is called a **'state'**.

# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

- ▶  $\neg(\phi \wedge \psi) \leftrightarrow$

# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

- ▶  $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

▶  $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

▶  $\neg(\phi \vee \psi) \leftrightarrow$



# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

▶  $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

▶  $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$

# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

- ▶  $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$
- ▶  $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$
- ▶  $(\text{true} \wedge \phi) \leftrightarrow$

# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

- ▶  $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$
- ▶  $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$
- ▶  $(\text{true} \wedge \phi) \leftrightarrow \phi$

# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

- ▶  $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$
- ▶  $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$
- ▶  $(\text{true} \wedge \phi) \leftrightarrow \phi$
- ▶  $(\text{false} \vee \phi) \leftrightarrow \phi$

# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

- ▶  $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$
- ▶  $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$
- ▶  $(\text{true} \wedge \phi) \leftrightarrow \phi$
- ▶  $(\text{false} \vee \phi) \leftrightarrow \phi$

# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

- ▶  $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$
- ▶  $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$
- ▶  $(\text{true} \wedge \phi) \leftrightarrow \phi$
- ▶  $(\text{false} \vee \phi) \leftrightarrow \phi$
- ▶  $\text{true} \vee \phi$

# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

- ▶  $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$
- ▶  $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$
- ▶  $(\text{true} \wedge \phi) \leftrightarrow \phi$
- ▶  $(\text{false} \vee \phi) \leftrightarrow \phi$
- ▶  $\text{true} \vee \phi$
- ▶  $\neg(\text{false} \wedge \phi)$

# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

▶  $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

▶  $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$

▶  $(\text{true} \wedge \phi) \leftrightarrow \phi$

▶  $(\text{false} \vee \phi) \leftrightarrow \phi$

▶  $\text{true} \vee \phi$

▶  $\neg(\text{false} \wedge \phi)$

▶  $(\phi \rightarrow \psi) \leftrightarrow$



# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

- ▶  $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$
- ▶  $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$
- ▶  $(\text{true} \wedge \phi) \leftrightarrow \phi$
- ▶  $(\text{false} \vee \phi) \leftrightarrow \phi$
- ▶  $\text{true} \vee \phi$
- ▶  $\neg(\text{false} \wedge \phi)$
- ▶  $(\phi \rightarrow \psi) \leftrightarrow (\neg\phi \vee \psi)$

# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

- ▶  $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$
- ▶  $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$
- ▶  $(\text{true} \wedge \phi) \leftrightarrow \phi$
- ▶  $(\text{false} \vee \phi) \leftrightarrow \phi$
- ▶  $\text{true} \vee \phi$
- ▶  $\neg(\text{false} \wedge \phi)$
- ▶  $(\phi \rightarrow \psi) \leftrightarrow (\neg\phi \vee \psi)$
- ▶  $\phi \rightarrow \text{true}$

# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

- ▶  $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$
- ▶  $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$
- ▶  $(\text{true} \wedge \phi) \leftrightarrow \phi$
- ▶  $(\text{false} \vee \phi) \leftrightarrow \phi$
- ▶  $\text{true} \vee \phi$
- ▶  $\neg(\text{false} \wedge \phi)$
- ▶  $(\phi \rightarrow \psi) \leftrightarrow (\neg\phi \vee \psi)$
- ▶  $\phi \rightarrow \text{true}$
- ▶  $\text{false} \rightarrow \phi$

# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

- ▶  $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$
- ▶  $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$
- ▶  $(\text{true} \wedge \phi) \leftrightarrow \phi$
- ▶  $(\text{false} \vee \phi) \leftrightarrow \phi$
- ▶  $\text{true} \vee \phi$
- ▶  $\neg(\text{false} \wedge \phi)$
- ▶  $(\phi \rightarrow \psi) \leftrightarrow (\neg\phi \vee \psi)$
- ▶  $\phi \rightarrow \text{true}$
- ▶  $\text{false} \rightarrow \phi$
- ▶  $(\text{true} \rightarrow \phi) \leftrightarrow \phi$

# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

- ▶  $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$
- ▶  $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$
- ▶  $(\text{true} \wedge \phi) \leftrightarrow \phi$
- ▶  $(\text{false} \vee \phi) \leftrightarrow \phi$
- ▶  $\text{true} \vee \phi$
- ▶  $\neg(\text{false} \wedge \phi)$
- ▶  $(\phi \rightarrow \psi) \leftrightarrow (\neg\phi \vee \psi)$
- ▶  $\phi \rightarrow \text{true}$
- ▶  $\text{false} \rightarrow \phi$
- ▶  $(\text{true} \rightarrow \phi) \leftrightarrow \phi$

# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

▶  $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

▶  $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$

▶  $(\text{true} \wedge \phi) \leftrightarrow \phi$

▶  $(\text{false} \vee \phi) \leftrightarrow \phi$

▶  $\text{true} \vee \phi$

▶  $\neg(\text{false} \wedge \phi)$

▶  $(\phi \rightarrow \psi) \leftrightarrow (\neg\phi \vee \psi)$

▶  $\phi \rightarrow \text{true}$

▶  $\text{false} \rightarrow \phi$

▶  $(\text{true} \rightarrow \phi) \leftrightarrow \phi$

▶  $(\phi \rightarrow \text{false}) \leftrightarrow$

# Useful Valid Formulas

Let  $\phi$  and  $\psi$  be arbitrary, closed formulas (whether valid or not).

The following formulas are valid:

▶  $\neg(\phi \wedge \psi) \leftrightarrow \neg\phi \vee \neg\psi$

▶  $\neg(\phi \vee \psi) \leftrightarrow \neg\phi \wedge \neg\psi$

▶  $(\text{true} \wedge \phi) \leftrightarrow \phi$

▶  $(\text{false} \vee \phi) \leftrightarrow \phi$

▶  $\text{true} \vee \phi$

▶  $\neg(\text{false} \wedge \phi)$

▶  $(\phi \rightarrow \psi) \leftrightarrow (\neg\phi \vee \psi)$

▶  $\phi \rightarrow \text{true}$

▶  $\text{false} \rightarrow \phi$

▶  $(\text{true} \rightarrow \phi) \leftrightarrow \phi$

▶  $(\phi \rightarrow \text{false}) \leftrightarrow \neg\phi$

# Useful Valid Formulas

Assume that  $x$  is the only variable which may appear freely in  $\phi$  or  $\psi$ .

The following formulas are valid:



# Useful Valid Formulas

Assume that  $x$  is the only variable which may appear freely in  $\phi$  or  $\psi$ .

The following formulas are valid:

- ▶  $\neg(\exists x \phi) \leftrightarrow \forall x \neg \phi$

# Useful Valid Formulas

Assume that  $x$  is the only variable which may appear freely in  $\phi$  or  $\psi$ .

The following formulas are valid:

- ▶  $\neg(\exists x; \phi) \leftrightarrow \forall x; \neg\phi$

# Useful Valid Formulas

Assume that  $x$  is the only variable which may appear freely in  $\phi$  or  $\psi$ .

The following formulas are valid:

- ▶  $\neg(\exists \tau x; \phi) \leftrightarrow \forall \tau x; \neg\phi$
- ▶  $\neg(\forall \tau x; \phi) \leftrightarrow$

# Useful Valid Formulas

Assume that  $x$  is the only variable which may appear freely in  $\phi$  or  $\psi$ .

The following formulas are valid:

- ▶  $\neg(\exists \tau x; \phi) \leftrightarrow \forall \tau x; \neg\phi$
- ▶  $\neg(\forall \tau x; \phi) \leftrightarrow \exists \tau x; \neg\phi$

# Useful Valid Formulas

Assume that  $x$  is the only variable which may appear freely in  $\phi$  or  $\psi$ .

The following formulas are valid:

- ▶  $\neg(\exists \tau x; \phi) \leftrightarrow \forall \tau x; \neg\phi$
- ▶  $\neg(\forall \tau x; \phi) \leftrightarrow \exists \tau x; \neg\phi$
- ▶  $(\forall \tau x; \phi \wedge \psi) \leftrightarrow$

# Useful Valid Formulas

Assume that  $x$  is the only variable which may appear freely in  $\phi$  or  $\psi$ .

The following formulas are valid:

- ▶  $\neg(\exists \tau x; \phi) \leftrightarrow \forall \tau x; \neg\phi$
- ▶  $\neg(\forall \tau x; \phi) \leftrightarrow \exists \tau x; \neg\phi$
- ▶  $(\forall \tau x; \phi \wedge \psi) \leftrightarrow (\forall \tau x; \phi) \wedge (\forall \tau x; \psi)$

# Useful Valid Formulas

Assume that  $x$  is the only variable which may appear freely in  $\phi$  or  $\psi$ .

The following formulas are valid:

- ▶  $\neg(\exists \tau x; \phi) \leftrightarrow \forall \tau x; \neg\phi$
- ▶  $\neg(\forall \tau x; \phi) \leftrightarrow \exists \tau x; \neg\phi$
- ▶  $(\forall \tau x; \phi \wedge \psi) \leftrightarrow (\forall \tau x; \phi) \wedge (\forall \tau x; \psi)$
- ▶  $(\exists \tau x; \phi \vee \psi) \leftrightarrow$

# Useful Valid Formulas

Assume that  $x$  is the only variable which may appear freely in  $\phi$  or  $\psi$ .

The following formulas are valid:

- ▶  $\neg(\exists \tau x; \phi) \leftrightarrow \forall \tau x; \neg\phi$
- ▶  $\neg(\forall \tau x; \phi) \leftrightarrow \exists \tau x; \neg\phi$
- ▶  $(\forall \tau x; \phi \wedge \psi) \leftrightarrow (\forall \tau x; \phi) \wedge (\forall \tau x; \psi)$
- ▶  $(\exists \tau x; \phi \vee \psi) \leftrightarrow (\exists \tau x; \phi) \vee (\exists \tau x; \psi)$



# Useful Valid Formulas

Assume that  $x$  is the only variable which may appear freely in  $\phi$  or  $\psi$ .

The following formulas are valid:

- ▶  $\neg(\exists \tau x; \phi) \leftrightarrow \forall \tau x; \neg\phi$
- ▶  $\neg(\forall \tau x; \phi) \leftrightarrow \exists \tau x; \neg\phi$
- ▶  $(\forall \tau x; \phi \wedge \psi) \leftrightarrow (\forall \tau x; \phi) \wedge (\forall \tau x; \psi)$
- ▶  $(\exists \tau x; \phi \vee \psi) \leftrightarrow (\exists \tau x; \phi) \vee (\exists \tau x; \psi)$

Are the following formulas also valid?

- ▶  $(\forall \tau x; \phi \vee \psi) \leftrightarrow (\forall \tau x; \phi) \vee (\forall \tau x; \psi)$
- ▶  $(\exists \tau x; \phi \wedge \psi) \leftrightarrow (\exists \tau x; \phi) \wedge (\exists \tau x; \psi)$

# Remark on Concrete Syntax

	Text book	SPIN	KeY
Negation	$\neg$	!	!
Conjunction	$\wedge$	&&	&
Disjunction	$\vee$		
Implication	$\rightarrow, \supset$	$\rightarrow$	$\rightarrow$
Equivalence	$\leftrightarrow$	$\langle \leftrightarrow \rangle$	$\langle \leftrightarrow \rangle$
Universal Quantifier	$\forall x; \phi$	n/a	<code>\forall x; <math>\phi</math></code>
Existential Quantifier	$\exists x; \phi$	n/a	<code>\exists x; <math>\phi</math></code>
Value equality	=	==	=

## Part II

# Sequent Calculus for FOL

# Motivation for a Sequent Calculus

How to show a formula valid in propositional logic?

→ use a semantic truth table.

How about FOL? Formula:  $\text{isEven}(x) \vee \text{isOdd}(x)$

$x$	$\text{isEven}(x)$	$\text{isOdd}(x)$	$\text{isEven}(x) \vee \text{isOdd}(x)$
1	$F$	$T$	$T$
2	$T$	$F$	$T$
...	...	...	...

And what about the interpretation of  $\text{isOdd}$  and  $\text{isEqual}$ ?

Checking validity via **semantics** does not work.

Instead...

# Reasoning by Syntactic Transformation

Prove Validity of  $\phi$  by **syntactic** transformation of  $\phi$

# Reasoning by Syntactic Transformation

Prove Validity of  $\phi$  by **syntactic** transformation of  $\phi$

Logic Calculus: **Sequent Calculus** based on notion of **sequent**:

$$\underbrace{\psi_1, \dots, \psi_m}_{\text{Antecedent}} \Rightarrow \underbrace{\phi_1, \dots, \phi_n}_{\text{Succedent}}$$

# Reasoning by Syntactic Transformation

Prove Validity of  $\phi$  by **syntactic** transformation of  $\phi$

Logic Calculus: **Sequent Calculus** based on notion of **sequent**:

$$\underbrace{\psi_1, \dots, \psi_m}_{\text{Antecedent}} \Rightarrow \underbrace{\phi_1, \dots, \phi_n}_{\text{Succedent}}$$

has same meaning as

$$(\psi_1 \wedge \dots \wedge \psi_m) \rightarrow (\phi_1 \vee \dots \vee \phi_n)$$

# Reasoning by Syntactic Transformation

Prove Validity of  $\phi$  by **syntactic** transformation of  $\phi$

Logic Calculus: **Sequent Calculus** based on notion of **sequent**:

$$\underbrace{\psi_1, \dots, \psi_m}_{\text{Antecedent}} \Rightarrow \underbrace{\phi_1, \dots, \phi_n}_{\text{Succedent}}$$

has same meaning as

$$(\psi_1 \wedge \dots \wedge \psi_m) \rightarrow (\phi_1 \vee \dots \vee \phi_n)$$

which has (for closed formulas  $\psi_i, \phi_i$ ) same meaning as

$$\{\psi_1, \dots, \psi_m\} \models \phi_1 \vee \dots \vee \phi_n$$



# Notation for Sequents

$$\psi_1, \dots, \psi_m \Rightarrow \phi_1, \dots, \phi_n$$

Consider antecedent/succedent as sets of formulas, may be empty

# Notation for Sequents

$$\psi_1, \dots, \psi_m \Rightarrow \phi_1, \dots, \phi_n$$

Consider antecedent/succedent as sets of formulas, may be empty

## Schema Variables

$\phi, \psi, \dots$  match formulas,  $\Gamma, \Delta, \dots$  match sets of formulas

Characterize infinitely many sequents with single schematic sequent, e.g.,

$$\Gamma \Rightarrow \phi \wedge \psi, \Delta$$

Matches any sequent with occurrence of conjunction in succedent

Call  $\phi \wedge \psi$  **main formula** and  $\Gamma, \Delta$  **side formulas** of sequent

Any sequent of the form  $\Gamma, \phi \Rightarrow \phi, \Delta$  is logically valid: **axiom**

# Sequent Calculus Rules

Write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible

$$\text{RuleName} \frac{\overbrace{\Gamma_1 \Rightarrow \Delta_1 \quad \dots \quad \Gamma_r \Rightarrow \Delta_r}^{\text{Premises}}}{\underbrace{\Gamma \Rightarrow \Delta}_{\text{Conclusion}}}$$

Meaning: For proving the Conclusion, it suffices to prove all Premises.

# Sequent Calculus Rules

Write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible

$$\text{RuleName} \frac{\overbrace{\Gamma_1 \Rightarrow \Delta_1 \quad \dots \quad \Gamma_r \Rightarrow \Delta_r}^{\text{Premises}}}{\underbrace{\Gamma \Rightarrow \Delta}_{\text{Conclusion}}}$$

Meaning: For proving the Conclusion, it suffices to prove all Premises.

## Example

$$\text{andRight} \frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$$

# Sequent Calculus Rules

Write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible

$$\text{RuleName} \frac{\overbrace{\Gamma_1 \Rightarrow \Delta_1 \quad \cdots \quad \Gamma_r \Rightarrow \Delta_r}^{\text{Premises}}}{\underbrace{\Gamma \Rightarrow \Delta}_{\text{Conclusion}}}$$

Meaning: For proving the Conclusion, it suffices to prove all Premises.

## Example

$$\text{andRight} \frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$$

Admissible to have no premisses (iff conclusion is valid, e.g., axiom)

A rule is **sound** (correct) iff the validity of its premisses implies the validity of its conclusion.

# 'Propositional' Sequent Calculus Rules

close  $\frac{}{\Gamma, \phi \Rightarrow \phi, \Delta}$

# 'Propositional' Sequent Calculus Rules

$$\text{close} \quad \frac{}{\Gamma, \phi \Rightarrow \phi, \Delta} \quad \text{true} \quad \frac{}{\Gamma \Rightarrow \text{true}, \Delta}$$

# 'Propositional' Sequent Calculus Rules

$$\text{close} \quad \frac{}{\Gamma, \phi \Rightarrow \phi, \Delta} \quad \text{true} \quad \frac{}{\Gamma \Rightarrow \text{true}, \Delta} \quad \text{false} \quad \frac{}{\Gamma, \text{false} \Rightarrow \Delta}$$



# 'Propositional' Sequent Calculus Rules

close  $\frac{}{\Gamma, \phi \Rightarrow \phi, \Delta}$     true  $\frac{}{\Gamma \Rightarrow \text{true}, \Delta}$     false  $\frac{}{\Gamma, \text{false} \Rightarrow \Delta}$

main	left side (antecedent)	right side (succedent)
not	$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, \neg\phi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \neg\phi, \Delta}$

# 'Propositional' Sequent Calculus Rules

$$\text{close} \quad \frac{}{\Gamma, \phi \Rightarrow \phi, \Delta} \quad \text{true} \quad \frac{}{\Gamma \Rightarrow \text{true}, \Delta} \quad \text{false} \quad \frac{}{\Gamma, \text{false} \Rightarrow \Delta}$$

main	left side (antecedent)	right side (succedent)
not	$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, \neg\phi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \neg\phi, \Delta}$
and	$\frac{\Gamma, \phi, \psi \Rightarrow \Delta}{\Gamma, \phi \wedge \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$

# 'Propositional' Sequent Calculus Rules

$$\text{close} \quad \frac{}{\Gamma, \phi \Rightarrow \phi, \Delta} \quad \text{true} \quad \frac{}{\Gamma \Rightarrow \text{true}, \Delta} \quad \text{false} \quad \frac{}{\Gamma, \text{false} \Rightarrow \Delta}$$

main	left side (antecedent)	right side (succedent)
not	$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, \neg\phi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \neg\phi, \Delta}$
and	$\frac{\Gamma, \phi, \psi \Rightarrow \Delta}{\Gamma, \phi \wedge \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$
or	$\frac{\Gamma, \phi \Rightarrow \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \vee \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}$

# 'Propositional' Sequent Calculus Rules

$$\text{close} \quad \frac{}{\Gamma, \phi \Rightarrow \phi, \Delta} \quad \text{true} \quad \frac{}{\Gamma \Rightarrow \text{true}, \Delta} \quad \text{false} \quad \frac{}{\Gamma, \text{false} \Rightarrow \Delta}$$

main	left side (antecedent)	right side (succedent)
not	$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, \neg \phi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \neg \phi, \Delta}$
and	$\frac{\Gamma, \phi, \psi \Rightarrow \Delta}{\Gamma, \phi \wedge \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$
or	$\frac{\Gamma, \phi \Rightarrow \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \vee \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}$
imp	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \rightarrow \psi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \rightarrow \psi, \Delta}$

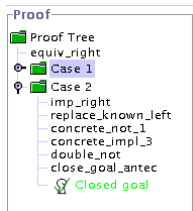
# Sequent Calculus Proofs

Goal to prove:  $\mathcal{G} = \psi_1, \dots, \psi_m \Rightarrow \phi_1, \dots, \phi_n$

- ▶ find rule  $\mathcal{R}$  whose conclusion **matches**  $\mathcal{G}$
- ▶ instantiate  $\mathcal{R}$  such that its conclusion is **identical** to  $\mathcal{G}$
- ▶ apply that instantiation to all premisses of  $\mathcal{R}$ , resulting in new goals  $\mathcal{G}_1, \dots, \mathcal{G}_r$
- ▶ recursively find proofs for  $\mathcal{G}_1, \dots, \mathcal{G}_r$
- ▶ tree structure with goal as root
- ▶ **close** proof branch when rule without premiss encountered

## Goal-directed proof search

In KeY tool proof displayed as JAVA Swing tree



# A Simple Proof

$$\frac{\begin{array}{c} \text{---} \quad \text{---} \\ \hline \hline \hline \end{array}}{\Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q}$$

# A Simple Proof

$$\frac{\frac{\frac{}{p \wedge (p \rightarrow q)} \Rightarrow q}{\Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q}}{\Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q}}$$

# A Simple Proof

$$\frac{\frac{\frac{}{p, (p \rightarrow q) \Rightarrow q}}{p \wedge (p \rightarrow q) \Rightarrow q}}{\Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q}}$$



# A Simple Proof

$$\frac{\frac{\frac{}{p \Rightarrow p, q}}{p, (p \rightarrow q) \Rightarrow q}}{p \wedge (p \rightarrow q) \Rightarrow q}}{\Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q}$$

# A Simple Proof

$$\begin{array}{c} \text{CLOSE} \frac{*}{p \Rightarrow p, q} \quad \frac{*}{p, q \Rightarrow q} \text{CLOSE} \\ \hline p, (p \rightarrow q) \Rightarrow q \\ \hline p \wedge (p \rightarrow q) \Rightarrow q \\ \hline \Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q \end{array}$$

# A Simple Proof

$$\frac{\frac{\text{CLOSE} \frac{*}{p \Rightarrow p, q}}{p, (p \rightarrow q) \Rightarrow q} \quad \frac{*}{p, q \Rightarrow q} \text{CLOSE}}{p \wedge (p \rightarrow q) \Rightarrow q}}{\Rightarrow (p \wedge (p \rightarrow q)) \rightarrow q}$$

A proof is **closed** iff all its branches are closed

Demo

prop.key

# Proving Validity of First-Order Formulas

## Proving a universally quantified formula

Claim:  $\forall x; \phi$  is true

How is such a claim proved in mathematics?

All even numbers are divisible by 2      $\forall \text{int } x; (\text{even}(x) \rightarrow \text{divByTwo}(x))$

Let  $c$  be an arbitrary number     Declare “unused” constant `int c`

The even number  $c$  is divisible by 2     prove `even(c)  $\rightarrow$  divByTwo(c)`

# Proving Validity of First-Order Formulas

## Proving a universally quantified formula

Claim:  $\forall \tau x; \phi$  is true

How is such a claim proved in mathematics?

All even numbers are divisible by 2      $\forall \text{int } x; (\text{even}(x) \rightarrow \text{divByTwo}(x))$

Let  $c$  be an arbitrary number     Declare “unused” constant  $\text{int } c$

The even number  $c$  is divisible by 2     prove  $\text{even}(c) \rightarrow \text{divByTwo}(c)$

## Sequent rule $\forall$ -right

$$\text{forallRight} \frac{\Gamma \Rightarrow [x/c] \phi, \Delta}{\Gamma \Rightarrow \forall \tau x; \phi, \Delta}$$

- ▶  $[x/c] \phi$  is result of replacing each occurrence of  $x$  in  $\phi$  with  $c$
- ▶  $c$  **new** constant of type  $\tau$

# Proving Validity of First-Order Formulas Cont'd

## Proving an existentially quantified formula

Claim:  $\exists x; \phi$  is true

How is such a claim proved in mathematics?

There is at least one prime number  $\exists \text{int } x; \text{prime}(x)$

Provide any "witness", say, 7 Use variable-free term `int 7`

7 is a prime number `prime(7)`

# Proving Validity of First-Order Formulas Cont'd

## Proving an existentially quantified formula

Claim:  $\exists \tau x; \phi$  is true

How is such a claim proved in mathematics?

There is at least one prime number  $\exists \text{int } x; \text{prime}(x)$

Provide any "witness", say, 7 Use variable-free term  $\text{int } 7$

7 is a prime number  $\text{prime}(7)$

## Sequent rule $\exists$ -right

$$\text{existsRight} \frac{\Gamma \Rightarrow [x/t] \phi, \exists \tau x; \phi, \Delta}{\Gamma \Rightarrow \exists \tau x; \phi, \Delta}$$

- ▶  $t$  any variable-free term of type  $\tau$
- ▶ Proof might not work with  $t$ ! Need to keep premise to try again

# Proving Validity of First-Order Formulas Cont'd

## Using a universally quantified formula

We assume  $\forall x; \phi$  is true

How is such a fact **used** in a mathematical proof?

We know that all primes are odd      $\forall \text{int } x; (\text{prime}(x) \rightarrow \text{odd}(x))$

In particular, this holds for 17     Use variable-free term `int 17`

We know: if 17 is prime it is odd      $\text{prime}(17) \rightarrow \text{odd}(17)$



# Proving Validity of First-Order Formulas Cont'd

## Using a universally quantified formula

We assume  $\forall \tau x; \phi$  is true

How is such a fact **used** in a mathematical proof?

We know that all primes are odd  $\forall \text{int } x; (\text{prime}(x) \rightarrow \text{odd}(x))$

In particular, this holds for 17 Use variable-free term `int 17`

We know: if 17 is prime it is odd  $\text{prime}(17) \rightarrow \text{odd}(17)$

## Sequent rule $\forall$ -left

$$\text{forallLeft} \frac{\Gamma, \forall \tau x; \phi, [x/t'] \phi \Rightarrow \Delta}{\Gamma, \forall \tau x; \phi \Rightarrow \Delta}$$

- ▶  $t'$  any variable-free term of type  $\tau$
- ▶ We might need other instances besides  $t'$ ! Keep premise  $\forall \tau x; \phi$

# Proving Validity of First-Order Formulas Cont'd

## Using an existentially quantified formula

We assume  $\exists x; \phi$  is true

How is such a fact **used** in a mathematical proof?

We know such an element exists. Let's give it a new name for future reference.

# Proving Validity of First-Order Formulas Cont'd

## Using an existentially quantified formula

We assume  $\exists \tau x; \phi$  is true

How is such a fact **used** in a mathematical proof?

We know such an element exists. Let's give it a new name for future reference.

## Sequent rule $\exists$ -left

$$\text{existsLeft} \frac{\Gamma, [x/c] \phi \Rightarrow \Delta}{\Gamma, \exists \tau x; \phi \Rightarrow \Delta}$$

- ▶  $c$  **new** constant of type  $\tau$

# Proving Validity of First-Order Formulas Cont'd

## Using an equation between terms

We assume  $t = t'$  is true

How is such a fact used in a mathematical proof?

Use  $x = y-1$  to simplify  $x+1/y$        $x = y-1 \Rightarrow 1 = x+1/y$

Replace  $x$  in conclusion with right-hand side of equation

We know:  $x+1/y$  equal to  $y-1+1/y$        $x = y-1 \Rightarrow 1 = y-1+1/y$

# Proving Validity of First-Order Formulas Cont'd

## Using an equation between terms

We assume  $t = t'$  is true

How is such a fact used in a mathematical proof?

Use  $x = y-1$  to simplify  $x+1/y$        $x = y-1 \Rightarrow 1 = x+1/y$

Replace  $x$  in conclusion with right-hand side of equation

We know:  $x+1/y$  equal to  $y-1+1/y$        $x = y-1 \Rightarrow 1 = y-1+1/y$

## Sequent rule =-left

$$\text{applyEqL} \frac{\Gamma, t = t', [t/t'] \phi \Rightarrow \Delta}{\Gamma, t = t', \phi \Rightarrow \Delta} \quad \text{applyEqR} \frac{\Gamma, t = t' \Rightarrow [t/t'] \phi, \Delta}{\Gamma, t = t' \Rightarrow \phi, \Delta}$$

- ▶ Always replace left- with right-hand side (use **eqSymm** if necessary)
- ▶  $t, t'$  variable-free terms of the same type

# Proving Validity of First-Order Formulas Cont'd

## Closing a subgoal in a proof

- ▶ We derived a sequent that is obviously valid

$$\text{close } \frac{}{\Gamma, \phi \Rightarrow \phi, \Delta} \quad \text{true } \frac{}{\Gamma \Rightarrow \text{true}, \Delta} \quad \text{false } \frac{}{\Gamma, \text{false} \Rightarrow \Delta}$$

- ▶ We derived an **equation** that is obviously valid

$$\text{eqClose } \frac{}{\Gamma \Rightarrow t = t, \Delta}$$

# Sequent Calculus for FOL at One Glance

	left side, antecedent	right side, succedent
$\forall$	$\frac{\Gamma, \forall \tau x; \phi, [x/t'] \phi \Rightarrow \Delta}{\Gamma, \forall \tau x; \phi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow [x/c] \phi, \Delta}{\Gamma \Rightarrow \forall \tau x; \phi, \Delta}$
$\exists$	$\frac{\Gamma, [x/c] \phi \Rightarrow \Delta}{\Gamma, \exists \tau x; \phi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow [x/t'] \phi, \exists \tau x; \phi, \Delta}{\Gamma \Rightarrow \exists \tau x; \phi, \Delta}$
$=$	$\frac{\Gamma, t = t' \Rightarrow [t/t'] \phi, \Delta}{\Gamma, t = t' \Rightarrow \phi, \Delta}$	$\frac{}{\Gamma \Rightarrow t = t, \Delta}$
	(+ application rule on left side)	

- ▶  $[t/t'] \phi$  is result of replacing each occurrence of  $t$  in  $\phi$  with  $t'$
- ▶  $t, t'$  variable-free terms of type  $\tau$
- ▶  $c$  **new** constant of type  $\tau$  (occurs not on current proof branch)
- ▶ Equations can be reversed by commutativity

# Recap: 'Propositional' Sequent Calculus Rules

main	left side (antecedent)	right side (succedent)
not	$\frac{\Gamma \Rightarrow \phi, \Delta}{\Gamma, \neg\phi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \Delta}{\Gamma \Rightarrow \neg\phi, \Delta}$
and	$\frac{\Gamma, \phi, \psi \Rightarrow \Delta}{\Gamma, \phi \wedge \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}$
or	$\frac{\Gamma, \phi \Rightarrow \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \vee \psi \Rightarrow \Delta}$	$\frac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}$
imp	$\frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \phi \rightarrow \psi \Rightarrow \Delta}$	$\frac{\Gamma, \phi \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \rightarrow \psi, \Delta}$
close	$\frac{}{\Gamma, \phi \Rightarrow \phi, \Delta}$	true $\frac{}{\Gamma \Rightarrow \text{true}, \Delta}$ false $\frac{}{\Gamma, \text{false} \Rightarrow \Delta}$



## Example (A simple theorem about binary relations)

$$\exists x; \forall y; p(x, y) \implies \forall y; \exists x; p(x, y)$$

Untyped logic: let static type of  $x$  and  $y$  be  $\top$

# Proving Validity of First-Order Formulas Cont'd

## Example (A simple theorem about binary relations)

$$\frac{\forall y; p(c, y) \Rightarrow \forall y; \exists x; p(x, y)}{\exists x; \forall y; p(x, y) \Rightarrow \forall y; \exists x; p(x, y)}$$

$\exists$ -left: substitute **new** constant  $c$  of type  $\top$  for  $x$

# Proving Validity of First-Order Formulas Cont'd

## Example (A simple theorem about binary relations)

$$\frac{\frac{\forall y; p(c, y) \Rightarrow \exists x; p(x, d)}{\forall y; p(c, y) \Rightarrow \forall y; \exists x; p(x, y)}}{\exists x; \forall y; p(x, y) \Rightarrow \forall y; \exists x; p(x, y)}$$

$\forall$ -right: substitute **new** constant  $d$  of type  $\top$  for  $y$

# Proving Validity of First-Order Formulas Cont'd

## Example (A simple theorem about binary relations)

$$\frac{\frac{\frac{p(c, d), \forall y; p(c, y) \Rightarrow \exists x; p(x, d)}{\forall y; p(c, y) \Rightarrow \exists x; p(x, d)}}{\forall y; p(c, y) \Rightarrow \forall y; \exists x; p(x, y)}}{\exists x; \forall y; p(x, y) \Rightarrow \forall y; \exists x; p(x, y)}$$

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

# Proving Validity of First-Order Formulas Cont'd

## Example (A simple theorem about binary relations)

$$\frac{\frac{\frac{p(c, d), \forall y; p(c, y) \Rightarrow p(c, d), \exists x; p(x, y)}{p(c, d), \forall y; p(c, y) \Rightarrow \exists x; p(x, d)}}{\forall y; p(c, y) \Rightarrow \exists x; p(x, d)}}{\forall y; p(c, y) \Rightarrow \forall y; \exists x; p(x, y)} \\ \frac{\forall y; p(c, y) \Rightarrow \forall y; \exists x; p(x, y)}{\exists x; \forall y; p(x, y) \Rightarrow \forall y; \exists x; p(x, y)}$$

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

# Proving Validity of First-Order Formulas Cont'd

## Example (A simple theorem about binary relations)

$$\begin{array}{c} * \\ \hline p(c, d), \forall y; p(c, y) \Rightarrow p(c, d), \exists x; p(x, y) \\ \hline p(c, d), \forall y; p(c, y) \Rightarrow \exists x; p(x, d) \\ \hline \forall y; p(c, y) \Rightarrow \exists x; p(x, d) \\ \hline \forall y; p(c, y) \Rightarrow \forall y; \exists x; p(x, y) \\ \hline \exists x; \forall y; p(x, y) \Rightarrow \forall y; \exists x; p(x, y) \end{array}$$

Close

# Proving Validity of First-Order Formulas Cont'd

## Example (A simple theorem about binary relations)

$$\begin{array}{c} * \\ \hline p(c, d), \forall y; p(c, y) \Rightarrow p(c, d), \exists x; p(x, y) \\ \hline p(c, d), \forall y; p(c, y) \Rightarrow \exists x; p(x, d) \\ \hline \forall y; p(c, y) \Rightarrow \exists x; p(x, d) \\ \hline \forall y; p(c, y) \Rightarrow \forall y; \exists x; p(x, y) \\ \hline \exists x; \forall y; p(x, y) \Rightarrow \forall y; \exists x; p(x, y) \end{array}$$

Demo

relSimple.key

# Proving Validity of First-Order Formulas Cont'd

Using an existentially quantified formula



# Proving Validity of First-Order Formulas Cont'd

Using an existentially quantified formula

Let  $x, y$  denote integer constants, both are not zero.

$$\neg(x = 0), \neg(y = 0) \quad \Rightarrow$$

# Proving Validity of First-Order Formulas Cont'd

Using an existentially quantified formula

Let  $x, y$  denote integer constants, both are not zero. We know further that  $x$  divides  $y$ .

$$\neg(x = 0), \neg(y = 0), \exists \text{int } k; k * x = y \implies$$

# Proving Validity of First-Order Formulas Cont'd

## Using an existentially quantified formula

Let  $x, y$  denote integer constants, both are not zero. We know further that  $x$  divides  $y$ .

**Show:**  $(y/x) * x = y$  ('/' is division on integers, i.e. the equation is not always true, e.g.  $x = 2, y = 1$ )

$$\neg(x = 0), \neg(y = 0), \exists \text{int } k; k * x = y \implies (y/x) * x = y$$

# Proving Validity of First-Order Formulas Cont'd

## Using an existentially quantified formula

Let  $x, y$  denote integer constants, both are not zero. We know further that  $x$  divides  $y$ .

**Show:**  $(y/x) * x = y$  ('/' is division on integers, i.e. the equation is not always true, e.g.  $x = 2, y = 1$ )

**Proof:** We know  $x$  divides  $y$ , i.e. there exists a  $k$  such that  $k * x = y$ .

Let now  $c$  denote such a  $k$ .

$$\frac{\neg(x = 0), \neg(y = 0), c * x = y \implies (y/x) * x = y}{\neg(x = 0), \neg(y = 0), \exists \text{int } k; k * x = y \implies (y/x) * x = y}$$

# Proving Validity of First-Order Formulas Cont'd

## Using an existentially quantified formula

Let  $x, y$  denote integer constants, both are not zero. We know further that  $x$  divides  $y$ .

**Show:**  $(y/x) * x = y$  ('/' is division on integers, i.e. the equation is not always true, e.g.  $x = 2, y = 1$ )

**Proof:** We know  $x$  divides  $y$ , i.e. there exists a  $k$  such that  $k * x = y$ . Let now  $c$  denote such a  $k$ . Hence we can replace  $y$  by  $c * x$  on the right side.

$$\frac{\frac{\neg(x = 0), \neg(y = 0), c * x = y \implies ((c * x)/x) * x = y}{\neg(x = 0), \neg(y = 0), c * x = y \implies (y/x) * x = y}}{\neg(x = 0), \neg(y = 0), \exists \text{int } k; k * x = y \implies (y/x) * x = y}$$

# Proving Validity of First-Order Formulas Cont'd

## Using an existentially quantified formula

Let  $x, y$  denote integer constants, both are not zero. We know further that  $x$  divides  $y$ .

**Show:**  $(y/x) * x = y$  ('/' is division on integers, i.e. the equation is not always true, e.g.  $x = 2, y = 1$ )

**Proof:** We know  $x$  divides  $y$ , i.e. there exists a  $k$  such that  $k * x = y$ . Let now  $c$  denote such a  $k$ . Hence we can replace  $y$  by  $c * x$  on the right side. ...  $\square$

\*

⋮

---

$$\neg(x = 0), \neg(y = 0), c * x = y \implies ((c * x)/x) * x = y$$

---

$$\neg(x = 0), \neg(y = 0), c * x = y \implies (y/x) * x = y$$

---

$$\neg(x = 0), \neg(y = 0), \exists \text{int } k; k * x = y \implies (y/x) * x = y$$

# Features of the KeY Theorem Prover

## Demo

`rel.key, twoInstances.key`

### Feature List

- ▶ Can work on multiple proofs simultaneously (task list)
- ▶ Proof trees visualized as JAVA Swing tree
- ▶ Point-and-click navigation within proof
- ▶ Undo proof steps, prune proof trees
- ▶ Pop-up menu with proof rules applicable in pointer focus
- ▶ Preview of rule effect as tool tip
- ▶ Quantifier instantiation and equality rules by drag-and-drop
- ▶ Possible to hide (and unhide) parts of a sequent
- ▶ Saving and loading of proofs

# Literature for this Lecture

essential:

- ▶ W. Ahrendt  
Using KeY  
Chapter 10 in [KeYbook]

further reading:

- ▶ M. Giese  
First-Order Logic  
Chapter 2 in [KeYbook]

**KeYbook** B. Beckert, R. Hähnle, and P. Schmitt, editors, **Verification of Object-Oriented Software: The KeY Approach**, vol 4334 of *LNCS* (Lecture Notes in Computer Science), Springer, 2006 (access via Chalmers library → E-books → Lecture Notes in Computer Science)



## Part III

# First-Order Semantics

# First-Order Semantics

## From propositional to first-order semantics

- ▶ In prop. logic, an interpretation of variables with  $\{T, F\}$  sufficed
- ▶ In first-order logic we must assign meaning to:
  - ▶ variables bound in quantifiers
  - ▶ constant and function symbols
  - ▶ predicate symbols
- ▶ Each variable or function value may denote a different item
- ▶ Respect typing: `int i`, `List l` **must** denote different items

## What we need (to interpret a first-order formula)

1. A collection of **typed universes** of items
2. A mapping from **variables** to items
3. A mapping from **function** arguments to function values
4. The set of argument tuples where a **predicate** is true

# First-Order Domains/Universes

1. A collection of **typed universes** of items

## Definition (Universe/Domain)

A non-empty set  $\mathcal{D}$  of items is a **universe** or **domain**

Each element of  $\mathcal{D}$  has a fixed type given by  $\delta : \mathcal{D} \rightarrow \mathcal{T}$

- ▶ Notation for the domain elements of type  $\tau \in \mathcal{T}$ :

$$\mathcal{D}^\tau = \{d \in \mathcal{D} \mid \delta(d) = \tau\}$$

- ▶ Each type  $\tau \in \mathcal{T}$  must 'contain' at least one domain element:

$$\mathcal{D}^\tau \neq \emptyset$$

# First-Order States

3. A mapping from function arguments to function values
4. The set of argument tuples where a predicate is true

## Definition (First-Order State)

Let  $\mathcal{D}$  be a domain with typing function  $\delta$

Let  $f$  be declared as  $\tau f(\tau_1, \dots, \tau_r)$ ;

Let  $p$  be declared as  $p(\tau_1, \dots, \tau_r)$ ;

Let  $\mathcal{I}(f) : \mathcal{D}^{\tau_1} \times \dots \times \mathcal{D}^{\tau_r} \rightarrow \mathcal{D}^{\tau}$

Let  $\mathcal{I}(p) \subseteq \mathcal{D}^{\tau_1} \times \dots \times \mathcal{D}^{\tau_r}$

Then  $\mathcal{S} = (\mathcal{D}, \delta, \mathcal{I})$  is a **first-order state**

# First-Order States Cont'd

## Example

Signature: `int i; short j; int f(int); Object obj; <(int,int);`  
 $\mathcal{D} = \{17, 2, o\}$  where all numbers are short

$$\mathcal{I}(i) = 17$$

$$\mathcal{I}(j) = 17$$

$$\mathcal{I}(obj) = o$$

$\mathcal{D}^{\text{int}}$	$\mathcal{I}(f)$
2	2
17	2

$\mathcal{D}^{\text{int}} \times \mathcal{D}^{\text{int}}$	in $\mathcal{I}(<)$ ?
(2, 2)	F
(2, 17)	T
(17, 2)	F
(17, 17)	F

One of uncountably many possible first-order states!

# Semantics of Reserved Signature Symbols

## Definition

**Equality** symbol  $=$  declared as  $= (\top, \top)$

Interpretation is fixed as  $\mathcal{I}(=) = \{(d, d) \mid d \in \mathcal{D}\}$

“Referential Equality” (holds if arguments refer to identical item)

Exercise: write down the predicate table for example domain

# Signature Symbols vs. Domain Elements

- ▶ Domain elements different from the terms representing them
- ▶ First-order formulas and terms have **no access** to domain

## Example

Signature: Object obj1, obj2;

Domain:  $\mathcal{D} = \{o\}$

In this state, necessarily  $\mathcal{I}(\text{obj1}) = \mathcal{I}(\text{obj2}) = o$

# Variable Assignments

2. A mapping from variables to objects

Think of variable assignment as environment for storage of local variables

## Definition (Variable Assignment)

A **variable assignment**  $\beta$  maps variables to domain elements

It respects the variable type, i.e., if  $x$  has type  $\tau$  then  $\beta(x) \in \mathcal{D}^\tau$

## Definition (Modified Variable Assignment)

Let  $y$  be variable of type  $\tau$ ,  $\beta$  variable assignment,  $d \in \mathcal{D}^\tau$ :

$$\beta_y^d(x) := \begin{cases} \beta(x) & x \neq y \\ d & x = y \end{cases}$$



# Semantic Evaluation of Terms

Given a first-order state  $\mathcal{S}$  and a variable assignment  $\beta$  it is possible to evaluate first-order terms under  $\mathcal{S}$  and  $\beta$

## Definition (Valuation of Terms)

$val_{\mathcal{S},\beta} : \text{Term} \rightarrow \mathcal{D}$  such that  $val_{\mathcal{S},\beta}(t) \in \mathcal{D}^\tau$  for  $t \in \text{Term}_\tau$ :

- ▶  $val_{\mathcal{S},\beta}(x) = \beta(x)$
- ▶  $val_{\mathcal{S},\beta}(f(t_1, \dots, t_r)) = \mathcal{I}(f)(val_{\mathcal{S},\beta}(t_1), \dots, val_{\mathcal{S},\beta}(t_r))$

# Semantic Evaluation of Terms Cont'd

## Example

Signature: `int i; short j; int f(int);`

$\mathcal{D} = \{17, 2, o\}$  where all numbers are short

Variables: Object `obj`; `int x`;

$$I(i) = 17$$

$$I(j) = 17$$

$\mathcal{D}^{\text{int}}$	$I(f)$
2	17
17	2

Var	$\beta$
obj	$o$
x	17

- ▶  $val_{S,\beta}(f(f(i)))$  ?
- ▶  $val_{S,\beta}(x)$  ?

# Semantic Evaluation of Formulas

## Definition (Valuation of Formulas)

$val_{S,\beta}(\phi)$  for  $\phi \in For$

- ▶  $val_{S,\beta}(p(t_1, \dots, t_r)) = T$  iff  $(val_{S,\beta}(t_1), \dots, val_{S,\beta}(t_r)) \in \mathcal{I}(p)$
- ▶  $val_{S,\beta}(\phi \wedge \psi) = T$  iff  $val_{S,\beta}(\phi) = T$  and  $val_{S,\beta}(\psi) = T$
- ▶ ... as in propositional logic
- ▶  $val_{S,\beta}(\forall \tau x; \phi) = T$  iff  $val_{S,\beta_x^d}(\phi) = T$  for all  $d \in \mathcal{D}^\tau$
- ▶  $val_{S,\beta}(\exists \tau x; \phi) = T$  iff  $val_{S,\beta_x^d}(\phi) = T$  for at least one  $d \in \mathcal{D}^\tau$

# Semantic Evaluation of Formulas Cont'd

## Example

Signature: `short j`; `int f(int)`; `Object obj`; `<(int,int)`;

$\mathcal{D} = \{17, 2, o\}$  where all numbers are short

$$\begin{aligned} \mathcal{I}(j) &= 17 \\ \mathcal{I}(\text{obj}) &= o \end{aligned}$$

$\mathcal{D}^{\text{int}}$	$\mathcal{I}(f)$
2	2
17	2

$\mathcal{D}^{\text{int}} \times \mathcal{D}^{\text{int}}$	in $\mathcal{I}(<)$ ?
(2, 2)	F
(2, 17)	T
(17, 2)	F
(17, 17)	F

- ▶  $\text{val}_{S,\beta}(f(j) < j)$  ?
- ▶  $\text{val}_{S,\beta}(\exists \text{int } x; f(x) = x)$  ?
- ▶  $\text{val}_{S,\beta}(\forall \text{Object } o1; \forall \text{Object } o2; o1 = o2)$  ?

# Semantic Notions

## Definition (Satisfiability, Truth, Validity)

$val_{S,\beta}(\phi) = T$		( $\phi$ is <b>satisfiable</b> )
$\mathcal{S} \models \phi$	iff for all $\beta : val_{S,\beta}(\phi) = T$	( $\phi$ is <b>true</b> in $\mathcal{S}$ )
$\models \phi$	iff for all $\mathcal{S} : \mathcal{S} \models \phi$	( $\phi$ is <b>valid</b> )

Closed formulas that are satisfiable are also true: one top-level notion

# Semantic Notions

## Definition (Satisfiability, Truth, Validity)

$$\begin{array}{lll} \text{val}_{\mathcal{S},\beta}(\phi) = T & & (\phi \text{ is } \mathbf{satisfiable}) \\ \mathcal{S} \models \phi & \text{iff for all } \beta : \text{val}_{\mathcal{S},\beta}(\phi) = T & (\phi \text{ is } \mathbf{true} \text{ in } \mathcal{S}) \\ \models \phi & \text{iff for all } \mathcal{S} : \mathcal{S} \models \phi & (\phi \text{ is } \mathbf{valid}) \end{array}$$

Closed formulas that are satisfiable are also true: one top-level notion

## Example

- ▶  $f(j) < j$  is true in  $\mathcal{S}$
- ▶  $\exists \mathbf{int} \ x; i = x$  is valid
- ▶  $\exists \mathbf{int} \ x; \neg(x = x)$  is not satisfiable