Formal Specification and Verification
Proving Theorems in First-Order Logic with KeY

Bernhard Beckert

Based on a lecture by Wolfgang Ahrendt and Reiner Hähnle at Chalmers University, Göteborg
Syntax, Semantics, Calculus

Syntax
Formula/Program

Semantics
"valid"

Calculus
"Derivable"
Syntax, Semantics, Calculus

Syntax

Formula/Program

Semantics

"valid"

Completeness

Calculus

"Derivable"
Syntax, Semantics, Calculus

- Syntax
  - Formula/Program
- Semantics
  - "valid"
- Calculus
  - "Derivable"

Completeness → Soundness
Notation for Sequents

\[ \psi_1, \ldots, \psi_m \implies \phi_1, \ldots, \phi_n \]

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

\[ \psi_1, \ldots, \psi_m \Rightarrow \phi_1, \ldots, \phi_n \]

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

**Schema Variables**

\[ \phi, \psi, \ldots \text{ match formulas, } \Gamma, \Delta, \ldots \text{ match sets of formulas} \]

Characterize infinitely many sequents with a single schematic sequent

\[ \Gamma \Rightarrow \Delta, \phi \& \psi \]

Matches any sequent with occurrence of conjunction in succedent

Call \( \phi \& \psi \) main formula and \( \Gamma, \Delta \) side formulas of sequent

Any sequent of the form \( \Gamma, \phi \Rightarrow \Delta, \phi \) is logically valid: axiom
Write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible

RuleName

Premisses

\[ \Gamma_1 \implies \Delta_1 \quad \ldots \quad \Gamma_r \implies \Delta_r \]

\[ \Gamma \implies \Delta \]

Conclusion
Sequent Calculus Rules of Propositional Logic

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

RuleName

```
Γ_1 \rightarrow Δ_1 \quad \cdots \quad Γ_r \rightarrow Δ_r
```

Premisses

```
Γ \rightarrow Δ
```

Conclusion

**Example**

**andRight**

```
Γ \rightarrow φ, Δ \quad Γ \rightarrow ψ, Δ
```

```
Γ \rightarrow φ \& ψ, Δ
```
Write syntactic transformation schema for sequents that reflects semantics of connectives as closely as possible.

\[
\text{RuleName} \quad \frac{
\Gamma_1 \Rightarrow \Delta_1 \quad \cdots \quad \Gamma_r \Rightarrow \Delta_r 
}{\Gamma \Rightarrow \Delta}
\]

**Example**

\[
\text{andRight} \quad \frac{
\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta 
}{\Gamma \Rightarrow \phi \& \psi, \Delta}
\]

**Sound** rule (essential): \( \vdash (\Gamma_1 \Rightarrow \Delta_1 \& \cdots \& \Gamma_r \Rightarrow \Delta_r) \rightarrow (\Gamma \Rightarrow \Delta) \)
Sequent Calculus Rules of Propositional Logic

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

\[
\text{RuleName} \quad \begin{array}{c}
\Gamma_1 \Rightarrow \Delta_1 \\
\vdots \\
\Gamma_r \Rightarrow \Delta_r \\
\hline
\Gamma \Rightarrow \Delta
\end{array}
\]

Example

\[
\text{andRight} \quad \frac{\Gamma \Rightarrow \phi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \ & \ \psi, \Delta}
\]

Sound rule (essential): \( \vdash (\Gamma_1 \Rightarrow \Delta_1 \ & \ \cdots \ & \ \Gamma_r \Rightarrow \Delta_r) \rightarrow (\Gamma \Rightarrow \Delta) \)

Complete rule (desirable): \( \vdash (\Gamma \Rightarrow \Delta) \rightarrow (\Gamma_1 \Rightarrow \Delta_1 \ & \ \cdots \ & \ \Gamma_r \Rightarrow \Delta_r) \)

Admissible to have no premisses (iff conclusion is valid, eg axiom)
## Rules of Propositional Sequent Calculus

<table>
<thead>
<tr>
<th>main</th>
<th>left side (antecedent)</th>
<th>right side (succedent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>( \Gamma \Rightarrow \phi, \Delta )</td>
<td>( \Gamma, \phi \Rightarrow \Delta )</td>
</tr>
<tr>
<td></td>
<td>( \Gamma, \neg \phi \Rightarrow \Delta )</td>
<td>( \Gamma \Rightarrow \neg \phi, \Delta )</td>
</tr>
</tbody>
</table>
# Rules of Propositional Sequent Calculus

<table>
<thead>
<tr>
<th>main</th>
<th>left side (antecedent)</th>
<th>right side (succedent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>[\Gamma \Rightarrow \phi, \Delta]</td>
<td>[\Gamma, \phi \Rightarrow \Delta]</td>
</tr>
<tr>
<td></td>
<td>[\Gamma \Rightarrow \phi \Rightarrow \Delta]</td>
<td>[\Gamma \Rightarrow \phi, \Delta]</td>
</tr>
<tr>
<td>and</td>
<td>[\Gamma, \phi, \psi \Rightarrow \Delta]</td>
<td>[\Gamma \Rightarrow \phi, \Delta]</td>
</tr>
<tr>
<td></td>
<td>[\Gamma \Rightarrow \phi &amp; \psi \Rightarrow \Delta]</td>
<td>[\Gamma \Rightarrow \phi &amp; \psi, \Delta]</td>
</tr>
</tbody>
</table>
## Rules of Propositional Sequent Calculus

<table>
<thead>
<tr>
<th>main</th>
<th>left side (antecedent)</th>
<th>right side (succedent)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>not</strong></td>
<td>$\Gamma \implies \phi, \Delta$</td>
<td>$\Gamma, \phi \implies \Delta$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma, ! \phi \implies \Delta$</td>
<td>$\Gamma \implies ! \phi, \Delta$</td>
</tr>
<tr>
<td><strong>and</strong></td>
<td>$\Gamma, \phi, \psi \implies \Delta$</td>
<td>$\Gamma \implies \phi, \Delta$ $\Gamma \implies \psi, \Delta$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma, \phi \implies \Delta$ $\Gamma, \psi \implies \Delta$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\Gamma, \phi \implies \Delta$ $\Gamma, \psi \implies \Delta$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\Gamma, \phi \implies \Delta$ $\Gamma, \psi \implies \Delta$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\Gamma \implies \phi &amp; \psi, \Delta$</td>
<td>$\Gamma \implies \phi &amp; \psi, \Delta$</td>
</tr>
<tr>
<td><strong>or</strong></td>
<td>$\Gamma, \phi \implies \Delta$ $\Gamma, \psi \implies \Delta$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\Gamma \implies \phi &amp; \psi, \Delta$</td>
<td>$\Gamma \implies \phi &amp; \psi, \Delta$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma \implies \phi &amp; \psi, \Delta$</td>
<td>$\Gamma \implies \phi &amp; \psi, \Delta$</td>
</tr>
</tbody>
</table>
### Rules of Propositional Sequent Calculus

<table>
<thead>
<tr>
<th>main</th>
<th>left side (antecedent)</th>
<th>right side (succedent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>[ \Gamma \Rightarrow \phi, \Delta ]</td>
<td>[ \Gamma, \phi \Rightarrow \Delta ]</td>
</tr>
<tr>
<td></td>
<td>[ \Gamma, ! \phi \Rightarrow \Delta ]</td>
<td>[ \Gamma \Rightarrow ! \phi, \Delta ]</td>
</tr>
<tr>
<td>and</td>
<td>[ \Gamma, \phi, \psi \Rightarrow \Delta ]</td>
<td>[ \Gamma \Rightarrow \phi, \Delta ]</td>
</tr>
<tr>
<td></td>
<td>[ \Gamma, \phi &amp; \psi \Rightarrow \Delta ]</td>
<td>[ \Gamma \Rightarrow \psi, \Delta ]</td>
</tr>
<tr>
<td>or</td>
<td>[ \Gamma, \phi \Rightarrow \Delta ]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ \Gamma, \psi \Rightarrow \Delta ]</td>
<td>[ \Gamma \Rightarrow \phi &amp; \psi, \Delta ]</td>
</tr>
<tr>
<td></td>
<td>[ \Gamma \Rightarrow \phi, \psi, \Delta ]</td>
<td>[ \Gamma \Rightarrow \phi \mid \psi, \Delta ]</td>
</tr>
<tr>
<td>imp</td>
<td>[ \Gamma \Rightarrow \phi \Rightarrow \Delta ]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ \Gamma, \phi \Rightarrow \Delta ]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ \Gamma, \psi \Rightarrow \Delta ]</td>
<td>[ \Gamma \Rightarrow \phi \Rightarrow \psi, \Delta ]</td>
</tr>
<tr>
<td></td>
<td>[ \Gamma \Rightarrow \phi \Rightarrow \psi, \Delta ]</td>
<td>[ \Gamma \Rightarrow \phi \Rightarrow \psi, \Delta ]</td>
</tr>
</tbody>
</table>
## Rules of Propositional Sequent Calculus

<table>
<thead>
<tr>
<th>main</th>
<th>left side (antecedent)</th>
<th>right side (succedent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>$\Gamma \Rightarrow \phi, \Delta$</td>
<td>$\Gamma,\phi \Rightarrow \Delta$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma, !\phi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow !\phi, \Delta$</td>
</tr>
<tr>
<td>and</td>
<td>$\Gamma, \phi, \psi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow \phi, \Delta$ $\Gamma \Rightarrow \psi, \Delta$ $\Gamma \Rightarrow \phi &amp; \psi, \Delta$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma, \phi &amp; \psi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow \phi &amp; \psi, \Delta$</td>
</tr>
<tr>
<td>or</td>
<td>$\Gamma, \phi \Rightarrow \Delta$ $\Gamma, \psi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow \phi, \psi, \Delta$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma, \phi \mid \psi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow \phi \mid \psi, \Delta$</td>
</tr>
<tr>
<td>imp</td>
<td>$\Gamma \Rightarrow \phi, \Delta$ $\Gamma, \psi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow \phi \Rightarrow \psi, \Delta$ $\Gamma \Rightarrow \phi \Rightarrow \psi, \Delta$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma, \phi \Rightarrow \psi \Rightarrow \Delta$</td>
<td>$\Gamma \Rightarrow \phi \Rightarrow \psi, \Delta$</td>
</tr>
<tr>
<td>close</td>
<td>$\Gamma, \phi \Rightarrow \phi, \Delta$</td>
<td>$\Gamma \Rightarrow \text{true}, \Delta$</td>
</tr>
<tr>
<td></td>
<td>$\Gamma, \text{true} \Rightarrow \Delta$</td>
<td>$\Gamma, \text{false} \Rightarrow \Delta$</td>
</tr>
<tr>
<td>true</td>
<td>$\Gamma \Rightarrow \text{true}, \Delta$</td>
<td>$\Gamma \Rightarrow \text{false}, \Delta$</td>
</tr>
</tbody>
</table>
Sequent Calculus in KeY

Reduce a given sequent by applying rules and producing simpler subgoals until all leaves of proof tree are “axioms”

Example (KeY input syntax for propositional validity problem)

\predicates \{ 
  p; 
  q; 
\}

\problem \{ 
  (p & (p -> q)) -> q 
\}

Demo

Examples/lect09/prop.key
Proving Validity of First-Order Formulas

**Proving a universally quantified formula**

∀ T x; φ is true in any model M

How is such a claim proven in mathematics?
Proving Validity of First-Order Formulas

**Proving a universally quantified formula**

\[ \forall T \ x; \ \phi \text{ is true in any model } \mathcal{M} \]

How is such a claim proven in mathematics?

All even numbers are divisible by 2

\[ \forall \text{int} \ x; \ (\text{even}(x) \rightarrow \text{divByTwo}(x)) \]
Proving Validity of First-Order Formulas

**Proving a universally quantified formula**

\[ \forall T \, x; \, \phi \text{ is true in any model } M \]

How is such a claim proven 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 \([\text{int } c] \)
Proving a universally quantified formula

∀ T x; φ is true in any model M

How is such a claim proven in mathematics?

All even numbers are divisible by 2
∀ int x; (even(x) → divByTwo(x))

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

The even number c is divisible by 2
even(c) → divByTwo(c)
Proving Validity of First-Order Formulas

Proving a universally quantified formula

∀ T x; φ is true in any model M

How is such a claim proven in mathematics?

All even numbers are divisible by 2  ∀ int x; (even(x) → divByTwo(x))

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

The even number c is divisible by 2  even(c) → divByTwo(c)

Sequent rule ∀-right

forallRight \[
\frac{Γ ⊢ [x/c] φ, Δ}{Γ ⊢ ∀ T x; φ, Δ}
\]

- [x/c]φ is result of replacing each occurrence of x in φ with c
- c new constant of type T
Proving Validity of First-Order Formulas Cont’d

**Proving an existentially quantified formula**

\[ \exists T \, x ; \, \phi \text{ is true in any model } \mathcal{M} \]

How is such a claim proven in mathematics?
Proving Validity of First-Order Formulas Cont’d

**Proving an existentially quantified formula**

$\exists T \, x; \, \phi$ is true in any model $\mathcal{M}$

How is such a claim proven in mathematics?

**There is at least one prime number**  
$\exists \, \text{int} \, x; \, \text{prime}(x)$
Proving Validity of First-Order Formulas Cont’d

Proving an existentially quantified formula

∃ T x; φ is true in any model M

How is such a claim proven in mathematics?

There is at least one prime number  ∃ int x; prime(x)

Provide any “witness”, say, 7  Use variable-free term int 7
Proving an existentially quantified formula

$\exists T \, x; \phi$ is true in any model $M$

How is such a claim proven 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

$\text{prime}(7)$
Proving an existentially quantified formula

∃ T x; φ is true in any model M

How is such a claim proven in mathematics?

There is at least one prime number ∃ int x; prime(x)

Provide any “witness”, say, 7 Use variable-free term int 7

7 is a prime number prime(7)

Sequent rule ∃-right

existsRight  \[ \Gamma \Rightarrow [x/t'] \phi, \exists T x; \phi, \Delta \]

\[ \Gamma \Rightarrow \exists T x; \phi, \Delta \]

- t’ any variable-free term with declared type T’ ⊑ T
- Proof might not work with t’! Need to keep premise to try again
Using a universally quantified formula

∀ T x; φ is true in any model M

How is such a fact used in a mathematical proof?
Using a universally quantified formula

∀ T x; φ is true in any model M

How is such a fact used in a mathematical proof?

We know that all primes are odd

∀ int x; (prime(x) → odd(x))
Using a universally quantified formula

\[ \forall T \, x; \, \phi \text{ is true in any model } M \]

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

\[ \text{Use variable-free term int} \, 17 \]
Proving Validity of First-Order Formulas Cont’d

**Using a universally quantified formula**

$\forall T \, x; \, \phi$ is true in any model $\mathcal{M}$

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 $\text{int} \, 17$

We know: if 17 is prime it is odd

$\text{prime}(17) \rightarrow \text{odd}(17)$
Using a universally quantified formula

∀ T x; φ is true in any model M

How is such a fact used in a mathematical proof?

We know that all primes are odd

∀ int x; (prime(x) → odd(x))

In particular, this holds for 17

Use variable-free term int 17

We know: if 17 is prime it is odd

prime(17) → odd(17)

Sequent rule ∀-left

∀-left \[ \Gamma, \forall T x; \phi, [x/t'] \phi \implies \Delta \]

∀-left \[ \Gamma, \forall T x; \phi \implies \Delta \]

- t' any variable-free term with declared type \( T' \subseteq T \)
- We might need other instances besides t'! Keep premise
Using an existentially quantified formula

\[ \exists T \, x; \phi \text{ is true in any model } \mathcal{M} \]

How is such a fact used in a mathematical proof?
Using an existentially quantified formula

$\exists T \; x; \; \phi$ is true in any model $\mathcal{M}$

How is such a fact used in a mathematical proof?

Every set $s$ can be well-ordered  

$\exists \text{Set } x; \; (\text{sameElem}(s, \; x) \; \& \; \text{wellOrder}(x))$
Using an existentially quantified formula

\[ \exists T \, x; \, \phi \text{ is true in any model } M \]

How is such a fact used in a mathematical proof?

Every set \( s \) can be well-ordered

\[ \exists \text{Set } x; \, \left( \text{sameElem}(s, x) \& \text{wellOrder}(x) \right) \]

Let \( s' \) be a well-order of \( s \)

\[ s' \text{ new constant of type OrdSet} \]
Using an existentially quantified formula

∀ T x; φ is true in any model M

How is such a fact used in a mathematical proof?

Every set s can be well-ordered  ∀ Set x; (sameElem(s, x) & wellOrder(x))

Let s’ be a well-order of s  s’ new constant of type OrdSet

We know: s’ is well-order of s  sameElem(s, s’) & wellOrder(s’)
Using an existentially quantified formula

\[ \exists T \, x; \phi \text{ is true in any model } M \]

How is such a fact used in a mathematical proof?

Every set \( s \) can be well-ordered

\[ \exists \text{Set } x; (\text{sameElem}(s, x) \& \text{wellOrder}(x)) \]

Let \( s' \) be a well-order of \( s \)

\( s' \) new constant of type OrdSet

We know: \( s' \) is well-order of \( s \)

\[ \text{sameElem}(s, s') \& \text{wellOrder}(s') \]

Sequent rule \( \exists \)-left

\[
\begin{align*}
\text{existsLeft} & \quad \Gamma, [x/c] \phi \Rightarrow \Delta \\
\therefore & \quad \Gamma, \exists T \, x; \phi \Rightarrow \Delta
\end{align*}
\]

\( c \) new constant of type \( T \)
Proving Validity of First-Order Formulas Cont’d

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)

\[ \forall y; \ p(c, y) \implies \forall y; \ \exists x; \ p(x, y) \]
\[ \exists x; \ \forall y; \ p(x, y) \implies \forall y; \ \exists x; \ p(x, y) \]

∃-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)

\[
\forall y; \ p(c, y) \Rightarrow \exists x; \ p(x, d)
\]

\[
\forall y; \ p(c, y) \Rightarrow \forall y; \ \exists x; \ p(x, y)
\]

\[
\exists x; \ \forall y; \ p(x, y) \Rightarrow \forall y; \ \exists x; \ p(x, y)
\]

\[
\forall \text{-right: substitute new constant } d \text{ of type } \top \text{ for } y
\]
Proving Validity of First-Order Formulas Cont’d

Example (A simple theorem about binary relations)

\[
\begin{align*}
  p(c, d), \forall y; p(c, y) &\implies \exists x; p(x, d) \\
  \forall y; p(c, y) &\implies \exists x; p(x, d) \\
  \forall y; p(c, y) &\implies \forall y; \exists x; p(x, y) \\
  \exists x; \forall y; p(x, y) &\implies \forall y; \exists x; p(x, y)
\end{align*}
\]

\(-left: \text{free to substitute any term of type } \top \text{ for } y, \text{ choose } d\)
Proving Validity of First-Order Formulas Cont’d

Example (A simple theorem about binary relations)

\[
p(c, d) \implies \exists x; p(x, d)
\]

\[
\forall y; p(c, y) \implies \exists x; p(x, d)
\]

\[
\forall y; p(c, y) \implies \forall y; \exists x; p(x, y)
\]

\[
\exists x; \forall y; p(x, y) \implies \forall y; \exists x; p(x, y)
\]

\[\forall\text{-left not needed anymore (hide)}\]
Proving Validity of First-Order Formulas Cont’d

Example (A simple theorem about binary relations)

\[
\begin{align*}
p(c, d) & \implies p(c, d), \ \exists x; \ p(x, y) \\
p(c, d) & \implies \exists x; \ p(x, d) \\
\forall y; \ p(c, y) & \implies \exists x; \ p(x, d) \\
\forall y; \ p(c, y) & \implies \forall y; \ \exists x; \ p(x, y) \\
\exists x; \ \forall y; \ p(x, y) & \implies \forall y; \ \exists x; \ p(x, y)
\end{align*}
\]

- 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)**

\[
\begin{align*}
p(c, d) \implies p(c, d) \\
p(c, d) \implies \exists x; p(x, d) \\
\forall y; p(c, y) \implies \exists x; p(x, d) \\
\forall y; p(c, y) \implies \forall y; \exists x; p(x, y) \\
\exists x; \forall y; p(x, y) \implies \forall y; \exists x; p(x, y)
\end{align*}
\]

\(\exists\)-right not needed anymore (hide)
Example (A simple theorem about binary relations)

\[
\begin{align*}
\ast & \quad p(c, d) \Rightarrow p(c, d) \\
& \Rightarrow p(c, d) \Rightarrow \exists x; p(x, d) \\
& \Rightarrow \forall y; p(c, y) \Rightarrow \exists x; p(x, d) \\
& \Rightarrow \forall y; p(c, y) \Rightarrow \forall y; \exists x; p(x, y) \\
& \Rightarrow \exists x; \forall y; p(x, y) \Rightarrow \forall y; \exists x; p(x, y)
\end{align*}
\]

Close
Example (A simple theorem about binary relations)

\[
\begin{align*}
* \\
p(c, d) & \implies p(c, d) \\
p(c, d) & \implies \exists x; p(x, d) \\
\forall y; p(c, y) & \implies \exists x; p(x, d) \\
\forall y; p(c, y) & \implies \forall y; \exists x; p(x, y) \\
\exists x; \forall y; p(x, y) & \implies \forall y; \exists x; p(x, y)
\end{align*}
\]
Using an equation between terms

\( t \simeq t' \) is true in any model \( \mathcal{M} \)

How is such a fact used in a mathematical proof?
**Using an equation between terms**

$t = t'$ is true in any model $\mathcal{M}$

How is such a fact used in a mathematical proof?

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

$x = y - 1 \Rightarrow 1 = x + 1/y$
Using an equation between terms

$t \equiv t'$ is true in any model $\mathcal{M}$

How is such a fact used in a mathematical proof?

Use $x \equiv y - 1$ to simplify $x + 1/y$

$x \equiv y - 1 \Rightarrow 1 \equiv x + 1/y$

Replace $x$ in conclusion with right-hand side of equation
Proving Validity of First-Order Formulas Cont’d

**Using an equation between terms**

$t \equiv t'$ is true in any model $\mathcal{M}$

How is such a fact used in a mathematical proof?

Use $x \equiv y - 1$ to simplify $x + 1/y$  

$x \equiv y - 1 \implies 1 \equiv x + 1/y$

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

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

$x \equiv y - 1 \implies 1 \equiv y - 1 + 1/y$
Using an equation between terms

$t \equiv t'$ is true in any model $M$

How is such a fact used in a mathematical proof?

Use $x \equiv y - 1$ to simplify $x + 1/y$

$x \equiv y - 1 \Rightarrow 1 \equiv x + 1/y$

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

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

$x \equiv y - 1 \Rightarrow 1 \equiv y - 1 + 1/y$

Sequent rule $\vdash$-left

$$
\frac{
\text{applyEq} \quad \Gamma, t \vdash t', [t/t'] \psi \Rightarrow [t/t'] \phi, \Delta
}{
\Gamma, t \vdash t', \psi \Rightarrow \phi, \Delta
}
$$

- Always replace left- with right-hand side (use eqSymm if necessary)
- Replacing term must be type-compatible with replaced term
- $t$ any variable-free term with declared type $T$, $t'$ with type $T' \sqsubseteq T$
Proving Validity of First-Order Formulas Cont’d

Closing a subgoal in a proof

- We derived a sequent that is obviously valid

\[
\begin{align*}
\text{close} & : \Gamma, \phi \Rightarrow \phi, \Delta \\
\text{true} & : \Gamma \Rightarrow \text{true}, \Delta \\
\text{false} & : \Gamma, \text{false} \Rightarrow \Delta
\end{align*}
\]

- We derived an equation that is obviously valid

\[
\text{eqClose} : \Gamma \Rightarrow t \equiv t, \Delta
\]
Features of the KeY Theorem Prover

Demo

Examples/lect09/rel.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
### Sequent Calculus for FOL at One Glance

<table>
<thead>
<tr>
<th>left side, antecedent</th>
<th>right side, succedent</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \forall \Gamma, \forall T , x; \phi, [x/t'] \phi \Rightarrow \Delta ]</td>
<td>[ \Gamma \Rightarrow [x/c] \phi, \Delta ]</td>
</tr>
<tr>
<td>[ \Gamma, \forall T , x; \phi \Rightarrow \Delta ]</td>
<td>[ \Gamma \Rightarrow \forall T , x; \phi, \Delta ]</td>
</tr>
<tr>
<td>[ \Gamma, [x/c] \phi \Rightarrow \Delta ]</td>
<td>[ \Gamma \Rightarrow [x/t'] \phi, \exists T , x; \phi, \Delta ]</td>
</tr>
<tr>
<td>[ \Gamma, \exists T , x; \phi \Rightarrow \Delta ]</td>
<td>[ \Gamma \Rightarrow \exists T , x; \phi, \Delta ]</td>
</tr>
<tr>
<td>[ \Gamma, t \equiv t', [t/t'] \psi \Rightarrow [t/t'] \phi, \Delta ]</td>
<td>[ \Gamma \Rightarrow t \equiv t, \Delta ]</td>
</tr>
</tbody>
</table>

- \([t/t'] \phi\) is result of replacing each occurrence of \(t\) in \(\phi\) with \(t'\)
- \(t\) any variable-free term with declared type \(T\)
- \(t'\) any variable-free term with declared type \(T' \sqsubseteq T\)
- \(c\) new constant of type \(T\) (occurs not on current proof branch)
- Equations can be reversed by commutativity
\texttt{\textbf{sorts} \{ // types are called \textquote{\textbf{sorts}}''
        Person; // one declaration per line
\}}

\texttt{\textbf{functions} \{
        \texttt{int age(Person); // \textquote{\texttt{int}}' predefined type
}\}}

\texttt{\textbf{predicates} \{
        parent(Person,Person);
}\}}

\texttt{\textbf{problem} \{ // Formula to be proven valid
        \texttt{\textbf{forall} Person son; \textbf{forall} Person father;
            (parent(father,son) -> age(father) > age(son))
}\}}

When doing Java verification, we want many function and predicate symbols to have the semantics prescribed by the JLS in all models.

Reserved symbols with fixed meaning so far: \( \div, \in, (T) \)
Types and Symbols with Fixed Meaning

When doing Java verification, we want many function and predicate symbols to have the semantics prescribed by the JLS in all models.

Reserved symbols with fixed meaning so far: $\div$, $\in T$, $(T)$

Types & symbols with fixed meaning in context of modeling Java

- $D^{\text{int}} = \{ d \in D \mid \delta(d) = \text{int} \} = \mathbb{Z}$
- KeY can switch to $\{\text{Integer.MIN VALUE, \ldots, Integer.MAX VALUE}\}$
- Default interpretation (and always used in first-order) is $\mathbb{Z}$
- Similar for short, byte
- Value types incomparable to reference types
- $D^{\text{boolean}} = \{ d \in D \mid \delta(d) = \text{boolean} \} = \{F, T\}$
- Usual operators in expressions as pre-defined signature symbols:
  - Fixed meaning: $\mathcal{I}(+) = +\mathbb{Z}$, $\mathcal{I}(\ast) = \ast\mathbb{Z}$, ...
  - $+, -, *, /, \%, \text{mod}, \ldots, -1, 0, 1, \ldots, <, \leq, >, \geq, \text{TRUE, FALSE}$
Rules for Type Casts and Type Predicates

- **Type predicate** formulas \( t \sqsubseteq T \)
  
  true iff dynamic type \( \delta(val_M(t)) \) is subtype of \( T \)

- **Type cast** terms \((T)t\)
  
  yields \( val_M(t) \) (identity) if cast succeeds, arb. element otherwise

---

**Typical typing rule**

The run-time type of a term is always compatible to its declared type

\[
\begin{array}{c}
\text{typeStatic} \\
\Gamma, t \sqsubseteq T \Rightarrow \Delta \\
\hline
\Gamma \Rightarrow \Delta
\end{array}
\]

\( T \) declared type of \( t \)

Ensures **type-safety** of typed first-order logic

- KeY first-order strategy applies suitable typing rules automatically
- All rules in KeY-Book Chapter 2, p59 (won’t be asked in exam)
Schubert’s Steamroller

Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them. Also, there are some grains, and grains are plants. Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants. Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves. Wolves do not like to eat foxes or grains, while birds like to eat caterpillars but not snails. Caterpillars and snails like to eat some plants.

Therefore, there is an animal that likes to eat a grain-eating animal.
\texttt{\texttt{\textbackslash sorts } \{ \texttt{Animal; }\\ \texttt{Wolf }\texttt{\textbackslash extends Animal; }\\ \texttt{Bird }\texttt{\textbackslash extends Animal; }\\ \texttt{Fox }\texttt{\textbackslash extends Animal; }\\ \texttt{Caterpillar }\texttt{\textbackslash extends Animal; }\\ \texttt{Snail }\texttt{\textbackslash extends Animal; }\\ \texttt{Plant; }\\ \texttt{Grain }\texttt{\textbackslash extends Plant; }\} }
Schubert’s Steamroller

Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them. Also, there are some grains, and grains are plants. Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants. Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves. Wolves do not like to eat foxes or grains, while birds like to eat caterpillars but not snails. Caterpillars and snails like to eat some plants.

Therefore, there is an animal that likes to eat a grain-eating animal.

\texttt{\textit{predicates} \{ \\
\hspace{1em} \textit{eats(Animal,any)}; \\
\hspace{1em} \textit{smaller(any,any)};
\}}
Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them. Also, there are some grains, and grains are plants. Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants. Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves. Wolves do not like to eat foxes or grains, while birds like to eat caterpillars but not snails. Caterpillars and snails like to eat some plants.

Therefore, there is an animal that likes to eat a grain-eating animal.

\( (\forall \text{Caterpillar } c; \forall \text{ Bird } b; \text{smaller}(c, b)) \land (\forall \text{Snail } s; \forall \text{ Bird } b; \text{smaller}(s, b)) \land (\forall \text{Bird } b; \forall \text{ Fox } f; \text{smaller}(b, f)) \land (\forall \text{Fox } f; \forall \text{ Wolf } w; \text{smaller}(f, w)) \)
Schubert’s Steamroller

Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them. Also, there are some grains, and grains are plants. Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants. Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves. Wolves do not like to eat foxes or grains, while birds like to eat caterpillars but not snails. Caterpillars and snails like to eat some plants.

Therefore, there is an animal that likes to eat a grain-eating animal.

\[
\begin{align*}
\forall \text{Bird } b; \forall \text{Caterpillar } c; \text{eats}(b,c) & \& \\
\forall \text{Caterpillar } c; \exists \text{Plant } p; \text{eats}(c,p) & \& \\
\forall \text{Snail } s; \exists \text{Plant } p; \text{eats}(s,p) & \& \\
\forall \text{Wolf } w; \forall \text{Fox } f; \neg \text{eats}(w,f) & \& \\
\forall \text{Wolf } w; \forall \text{Grain } g; \neg \text{eats}(w,g) & \& \\
\forall \text{Bird } b; \forall \text{Snail } s; \neg \text{eats}(b,s)
\end{align*}
\]
Schubert’s Steamroller

Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them. Also, there are some grains, and grains are plants. Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants. Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves. Wolves do not like to eat foxes or grains, while birds like to eat caterpillars but not snails. Caterpillars and snails like to eat some plants.

Therefore, there is an animal that likes to eat a grain-eating animal.

$(\forall \text{Animal } a;\\\\(\forall \text{Plant } p; \text{eats}(a, p)) |\\\\(\forall \text{Animal } as;\\\\((\text{smaller}(as, a) &\\\\\exists \text{Plant } p; \text{eats}(as, p)) \rightarrow \text{eats}(a, as))))$
Schubert’s Steamroller

Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them. Also, there are some grains, and grains are plants. Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants. Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves. Wolves do not like to eat foxes or grains, while birds like to eat caterpillars but not snails. Caterpillars and snails like to eat some plants.

Therefore, there is an animal that likes to eat a grain-eating animal.

( ... ) ->
(\exists Animal a;
 \exists Animal ga; ((\exists Grain g; eats(ga,g)) & eats(a,ga)))
Schubert’s Steamroller

Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them. Also, there are some grains, and grains are plants. Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants. Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves. Wolves do not like to eat foxes or grains, while birds like to eat caterpillars but not snails. Caterpillars and snails like to eat some plants.

Therefore, there is an animal that likes to eat a grain-eating animal.

( ... ) \rightarrow
(\exists \text{ Animal } a;
\exists \text{ Animal } ga; ((\exists \text{ Grain } g; \text{ eats}(ga,g)) \& \text{ eats}(a,ga)))

Demo

Examples/lect09/sr.key
Automated Proof Search

KeY has built-in heuristics to apply FO rules automatically

- Select **Proof Search Strategy** “FOL”
- Specify **Max. Rule Applications** or **Time limit**
- **Run/Stop** button
- See **Goals** tab

Look out for common problems
- Long branches with same rule applied to quantified formulas
- Too low bound on proof search
  - If search doesn’t terminate:
    - Check **Java DL Proof Search Strategy**
    - Instantiate quantifiers “by-hand” (might need to declare suitable constant in problem)
Automated Proof Search

KeY has built-in heuristics to apply FO rules automatically

- Select Proof Search Strategy “FOL”
- Specify Max. Rule Applications or Time limit
- Run/Stop button
- See Goals tab

Look out for common problems

- Long branches with same rule applied to quantified formulas
- Too low bound on proof search
- If search doesn’t terminate:
  - Check Java DL Proof Search Strategy
  - Instantiate quantifiers “by-hand”
    (might need to declare suitable constant in problem)
Failed Proofs

Sometimes (often) an interactive or automatic fail

Reasons for failed proofs

- The automatic proof strategy of KeY is too weak (did you check FOL?)
- When trying manually:
  - Did you use the right instantiations?
  - Perhaps you need to apply an equality?
- Your goal is not a valid formula!

An unsuccessful proof can give important clues why!
Theorem

Let the formula $G$ be the goal of a sequent proof. Assume there is an open leaf $L = \Gamma \implies \Delta$ in a sequent proof such that:

1. $L$ is not closed
2. There is a first-order model $M$ that:
   - $M \models \gamma$ for all $\gamma \in \Gamma$
   - $M \models \neg \delta$ for all $\delta \in \Delta$

Then $M \models \neg G$, i.e., $M$ is a counter example for $G$.

NOTE: This only holds as long as no hiding or weakening rules are used that destroy proof confluence (e.g., the invariant rule for loops).
Learning from Failed Proofs

How to proceed

1. **Java DL Proof Search Strategy** with Quantifier Treatment unrestricted
2. Run prover, inspect open Goals $L$
3. If necessary, instantiate $\forall$-left, $\exists$-right by hand
4. Find model that makes $L$’s antecedent true and succedent false
5. Go back to $G$ and find out was was wrong
   Often, the patch is to add a $\gamma \in L$ or a $! \delta \in L$ to the premise of $G$
Learning from Failed Proofs

How to proceed

1. Java DL Proof Search Strategy with Quantifier Treatment unrestricted
2. Run prover, inspect open Goals $L$
3. If necessary, instantiate $\forall$-left, $\exists$-right by hand
4. Find model that makes $L$’s antecedent true and succedent false
5. Go back to $G$ and find out what was wrong
   Often, the patch is to add a $\gamma \in L$ or a $! \delta \in L$ to the premise of $G$

Demo

Examples/lect09/model.key
## Literature for this Lecture

### Essential

| KeY Book | Verification of Object-Oriented Software (see course web page), Chapter 10: **Using KeY** (up to and incl. 10.2.2) |
| KeY Book | Verification of Object-Oriented Software (see course web page), Chapter 2: **First-Order Logic** |

### Recommended/Background

| Huth & Ryan | Logic in Computer Science, 2nd edn., Cambridge University Press, 2004 |
| Fitting     | First-Order Logic and Automated Theorem Proving, 2nd edn., Springer 1996 |