

Logik für Informatiker
Logic for computer scientists
Proof rules for quantifiers

Till Mossakowski, Lutz Schröder

WiSe 2011/12

Universal Elimination (\forall Elim)

$$\triangleright \left| \begin{array}{l} \forall x S(x) \\ \vdots \\ S(c) \end{array} \right.$$

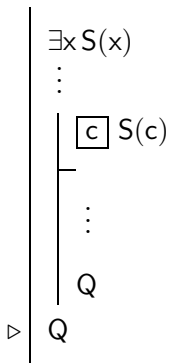
Existential Introduction (\exists Intro)

$$\triangleright \left| \begin{array}{c} S(c) \\ \vdots \\ \exists x S(x) \end{array} \right.$$

Example: \forall -Elim and \exists -Intro

$$\begin{array}{|l} \forall x[\text{Cube}(x) \rightarrow \text{Large}(x)] \\ \forall x[\text{Large}(x) \rightarrow \text{LeftOf}(x, b)] \\ \text{Cube}(d) \\ \hline \exists x[\text{Large}(x) \wedge \text{LeftOf}(x, b)] \end{array}$$

Existential Elimination (\exists Elim):

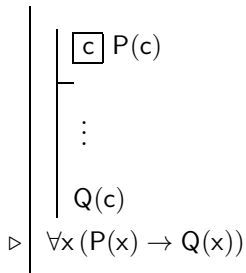


Where c does not occur outside the subproof where it is introduced.

Example: \exists -Elim

$$\begin{array}{|l} \forall x[\text{Cube}(x) \rightarrow \text{Large}(x)] \\ \forall x[\text{Large}(x) \rightarrow \text{LeftOf}(x, b)] \\ \exists x \text{ Cube}(x) \\ \hline \exists x[\text{Large}(x) \wedge \text{LeftOf}(x, b)] \end{array}$$

General Conditional Proof (\forall Intro):

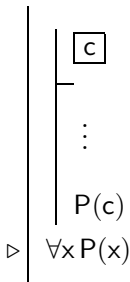


Where c does not occur outside the subproof where it is introduced.

Example: General Conditional Proof

$$\left| \begin{array}{l} \forall x[\text{Cube}(x) \rightarrow \text{Large}(x)] \\ \forall x[\text{Large}(x) \rightarrow \text{LeftOf}(x, b)] \\ \hline \forall x[\text{Cube}(x) \rightarrow \text{LeftOf}(x, b)] \end{array} \right.$$

Universal Introduction (\forall Intro):



Where c does not occur outside the subproof where it is introduced.

Prenex normal form (reminder)

$$\left\{ \begin{array}{l} \exists x \text{Cube}(x) \rightarrow \forall y \text{Small}(y) \\ \forall x \forall y (\text{Cube}(x) \rightarrow \text{Small}(y)) \end{array} \right.$$

Example with multiple quantifiers

$$\left\{ \begin{array}{l} \exists y[\text{Girl}(y) \wedge \forall x(\text{Boy}(x) \rightarrow \text{Likes}(x, y))] \\ \forall x[\text{Boy}(x) \rightarrow \exists y(\text{Girl}(y) \wedge \text{Likes}(x, y))] \end{array} \right.$$

Example: de Morgan's Law

$$\left\{ \begin{array}{l} \neg \forall x P(x) \\ \exists x \neg P(x) \end{array} \right.$$

(is not valid in intuitionistic logic, only in classical logic)

Example: The Barber Paradox

$$\exists z \exists x [ManOf(x, z) \wedge \forall y (ManOf(y, z) \rightarrow (Shave(x, y) \leftrightarrow \neg Shave(y, y)))]$$
$$\perp$$

Induction

Induction is like a chain of dominoes. You need

- the dominoes must be close enough together \Rightarrow one falling dominoe knocks down the next (*inductive step*)
- you need to knock down the first dominoe (*inductive basis*)



possible.

Note: in the inductive step, branching is

Induction

Induction is like a chain of dominoes. You need

- the dominoes must be close enough together \Rightarrow one falling dominoe knocks down the next (*inductive step*)
- you need to knock down the first dominoe (*inductive basis*)



possible.

Note: in the inductive step, branching is

Inductive definition: Natural numbers

- 1 0 is a natural number.
- 2 If n is natural number, then $suc(n)$ is a natural number.
- 3 There is no natural number whose successor is 0.
- 4 Two different natural numbers have different successors.
- 5 Nothing is a natural number unless generated by repeated applications of (1) and (2).

Recursive definition of functions

$$\forall y(0 + y = y)$$

$$\forall x \forall y(\text{succ}(x) + y = \text{succ}(x + y))$$

$$\forall y(0 * y = 0)$$

$$\forall x \forall y(\text{succ}(x) * y = (x * y) + y)$$

Formalization of Peano's axioms

- 1 a constant 0
- 2 a unary function symbol suc
- 3 $\forall n \neg suc(n) = 0$
- 4 $\forall m \forall n suc(m) = suc(n) \rightarrow m = n$
- 5 $(\Phi(x/0) \wedge \forall n (\Phi(x/n) \rightarrow \Phi(x/suc(n)))) \rightarrow \forall n \Phi(x/n)$
if Φ is a formula with a free variable x , and
 $\Phi(x/t)$ denotes the replacement of x with t within Φ

Inductive proofs

Take $\Phi(x) := \forall y \forall z (x + (y + z) = (x + y) + z)$. Then

$$(\Phi(x/0) \wedge \forall n (\Phi(x/n) \rightarrow \Phi(x/suc(n)))) \rightarrow \forall n \Phi(x/n)$$

is just

$$\begin{aligned} & (\forall y \forall z (0 + (y + z) = (0 + y) + z) \\ & \quad \wedge \forall n \forall y \forall z (n + (y + z) = (n + y) + z \\ & \quad \quad \rightarrow suc(n) + (y + z) = (suc(n) + y) + z)) \\ & \quad \rightarrow \forall n \forall y \forall z (n + (y + z) = (n + y) + z) \end{aligned}$$

With this, we can prove $\forall n \forall y \forall z (n + (y + z) = (n + y) + z)$

Inductive datatypes: Lists of natural numbers

- 1 The empty list $[]$ is a list.
- 2 If l is a list and n is natural number, then $cons(n, l)$ is a list.
- 3 Nothing is a list unless generated by repeated applications of (1) and (2).

Note: This needs *many-sorted* first-order logic.

We have two sorts of objects: natural numbers and lists.

Recursive definition of functions over lists

$$\text{length}([]) = 0$$

$$\forall n : \text{Nat} \forall l : \text{List} (\text{length}(\text{cons}(n, l)) = \text{suc}(\text{length}(l)))$$

$$\forall l : \text{List} ([] ++ l = l)$$

$$\forall n : \text{Nat} \forall l_1 : \text{List} \forall l_2 : \text{List}$$

$$(\text{cons}(n, l_1) ++ l_2 = \text{cons}(n, l_1 ++ l_2))$$

$$\forall l_1 : List \ \forall l_2 : List \ \forall l_3 : List \\ (l_1 ++ (l_2 ++ l_3) = (l_1 ++ l_2) ++ l_3)$$

$$\forall l_1 : List \ \forall l_2 : List \\ (length(l_1 ++ l_2) = length(l_1) + length(l_2))$$