

# Isabelle/FOL — First-Order Logic

Larry Paulson and Markus Wenzel

October 10, 2011

## Contents

<b>1</b>	<b>Intuitionistic first-order logic</b>	<b>1</b>
1.1	Syntax and axiomatic basis . . . . .	2
1.2	Lemmas and proof tools . . . . .	4
1.3	Intuitionistic Reasoning . . . . .	10
1.4	Atomizing meta-level rules . . . . .	11
1.5	Atomizing elimination rules . . . . .	11
1.6	Computational rules . . . . .	12
1.7	“Let” declarations . . . . .	12
1.8	Intuitionistic simplification rules . . . . .	13
<b>2</b>	<b>Classical first-order logic</b>	<b>15</b>
2.1	The classical axiom . . . . .	15
2.2	Lemmas and proof tools . . . . .	15
<b>3</b>	<b>Classical Reasoner</b>	<b>17</b>
3.1	Other simple lemmas . . . . .	19
3.2	Proof by cases and induction . . . . .	20

## 1 Intuitionistic first-order logic

```
theory IFOL
imports Pure
uses
  ~/src/Tools/misc-legacy.ML
  ~/src/Provers/splitter.ML
  ~/src/Provers/hypsubst.ML
  ~/src/Tools/IsaPlanner/zipper.ML
  ~/src/Tools/IsaPlanner/isand.ML
  ~/src/Tools/IsaPlanner/rw-tools.ML
  ~/src/Tools/IsaPlanner/rw-inst.ML
  ~/src/Tools/eqsubst.ML
  ~/src/Provers/quantifier1.ML
```

```

~~/src/Tools/intuitionistic.ML
~~/src/Tools/project-rule.ML
~~/src/Tools/atomize-elim.ML
(fologic.ML)
(intprover.ML)

```

**begin**

## 1.1 Syntax and axiomatic basis

$\langle ML \rangle$

```

classes term
default-sort term

```

```

typedecl o

```

**judgment**

```

Trueprop    :: o => prop          ((-) 5)

```

**consts**

```

True         :: o
False        :: o

```

```

eq           :: ['a, 'a] => o      (infixl = 50)

```

```

Not          :: o => o             (~ - [40] 40)
conj         :: [o, o] => o        (infixr & 35)
disj         :: [o, o] => o        (infixr | 30)
imp          :: [o, o] => o        (infixr --> 25)
iff          :: [o, o] => o        (infixr <-> 25)

```

```

All          :: ('a => o) => o      (binder ALL 10)
Ex           :: ('a => o) => o      (binder EX 10)
Ex1          :: ('a => o) => o      (binder EX! 10)

```

**abbreviation**

```

not-equal :: ['a, 'a] => o (infixl ~= 50) where
x ~= y == ~ (x = y)

```

**notation** (*xsymbols*)

```

not-equal (infixl ≠ 50)

```

**notation** (*HTML output*)

```

not-equal (infixl ≠ 50)

```

**notation** (*xsymbols*)

*Not* ( $\neg$  - [40] 40) **and**  
*conj* (**infixr**  $\wedge$  35) **and**  
*disj* (**infixr**  $\vee$  30) **and**  
*All* (**binder**  $\forall$  10) **and**  
*Ex* (**binder**  $\exists$  10) **and**  
*Ex1* (**binder**  $\exists!$  10) **and**  
*imp* (**infixr**  $\longrightarrow$  25) **and**  
*iff* (**infixr**  $\longleftrightarrow$  25)

**notation** (*HTML output*)

*Not* ( $\neg$  - [40] 40) **and**  
*conj* (**infixr**  $\wedge$  35) **and**  
*disj* (**infixr**  $\vee$  30) **and**  
*All* (**binder**  $\forall$  10) **and**  
*Ex* (**binder**  $\exists$  10) **and**  
*Ex1* (**binder**  $\exists!$  10)

**finalconsts**

*False All Ex eq conj disj imp*

**axiomatization where**

*refl:*  $a=a$  **and**  
*subst:*  $a=b \implies P(a) \implies P(b)$

**axiomatization where**

*conjI:*  $[| P; Q |] \implies P \& Q$  **and**  
*conjunct1:*  $P \& Q \implies P$  **and**  
*conjunct2:*  $P \& Q \implies Q$  **and**  
  
*disjI1:*  $P \implies P | Q$  **and**  
*disjI2:*  $Q \implies P | Q$  **and**  
*disjE:*  $[| P | Q; P \implies R; Q \implies R |] \implies R$  **and**  
  
*impI:*  $(P \implies Q) \implies P \longrightarrow Q$  **and**  
*mp:*  $[| P \longrightarrow Q; P |] \implies Q$  **and**  
  
*FalseE:*  $False \implies P$

**axiomatization where**

*allI:*  $(!!x. P(x)) \implies (ALL x. P(x))$  **and**  
*spec:*  $(ALL x. P(x)) \implies P(x)$  **and**  
  
*exI:*  $P(x) \implies (EX x. P(x))$  **and**

*exE*:  $\llbracket \text{EX } x. P(x); \text{!!}x. P(x) \implies R \rrbracket \implies R$

**axiomatization where**

*eq-reflection*:  $(x=y) \implies (x==y)$  **and**  
*iff-reflection*:  $(P<->Q) \implies (P==Q)$

**lemmas** *strip = impI allI*

**defs**

*True-def*:  $\text{True} == \text{False} \dashrightarrow \text{False}$   
*not-def*:  $\sim P == P \dashrightarrow \text{False}$   
*iff-def*:  $P<->Q == (P \dashrightarrow Q) \ \& \ (Q \dashrightarrow P)$

*ex1-def*:  $\text{Ex1}(P) == \text{EX } x. P(x) \ \& \ (\text{ALL } y. P(y) \dashrightarrow y=x)$

## 1.2 Lemmas and proof tools

**lemma** *TrueI*: *True*  
*<proof>*

**lemma** *conjE*:  
**assumes major**:  $P \ \& \ Q$   
**and r**:  $\llbracket P; Q \rrbracket \implies R$   
**shows**  $R$   
*<proof>*

**lemma** *impE*:  
**assumes major**:  $P \dashrightarrow Q$   
**and**  $P$   
**and r**:  $Q \implies R$   
**shows**  $R$   
*<proof>*

**lemma** *allE*:  
**assumes major**:  $\text{ALL } x. P(x)$   
**and r**:  $P(x) \implies R$   
**shows**  $R$   
*<proof>*

**lemma** *all-dupE*:  
 **assumes** *major*:  $ALL\ x.\ P(x)$   
 **and** *r*:  $[| P(x); ALL\ x.\ P(x) |] ==> R$   
 **shows**  $R$   
  $\langle proof \rangle$

**lemma** *notI*:  $(P ==> False) ==> \sim P$   
  $\langle proof \rangle$

**lemma** *notE*:  $[| \sim P; P |] ==> R$   
  $\langle proof \rangle$

**lemma** *rev-notE*:  $[| P; \sim P |] ==> R$   
  $\langle proof \rangle$

**lemma** *not-to-imp*:  
 **assumes**  $\sim P$   
 **and** *r*:  $P --> False ==> Q$   
 **shows**  $Q$   
  $\langle proof \rangle$

**lemma** *rev-mp*:  $[| P; P --> Q |] ==> Q$   
  $\langle proof \rangle$

**lemma** *contrapos*:  
 **assumes** *major*:  $\sim Q$   
 **and** *minor*:  $P ==> Q$   
 **shows**  $\sim P$   
  $\langle proof \rangle$

$\langle ML \rangle$

**lemma** *iffI*:  $[| P ==> Q; Q ==> P |] ==> P <-> Q$   
  $\langle proof \rangle$

**lemma iffE:**

**assumes major:**  $P \leftrightarrow Q$

**and r:**  $P \rightarrow Q \implies Q \rightarrow P \implies R$

**shows**  $R$

*<proof>*

**lemma iffD1:**  $[| P \leftrightarrow Q; P |] \implies Q$

*<proof>*

**lemma iffD2:**  $[| P \leftrightarrow Q; Q |] \implies P$

*<proof>*

**lemma rev-iffD1:**  $[| P; P \leftrightarrow Q |] \implies Q$

*<proof>*

**lemma rev-iffD2:**  $[| Q; P \leftrightarrow Q |] \implies P$

*<proof>*

**lemma iff-refl:**  $P \leftrightarrow P$

*<proof>*

**lemma iff-sym:**  $Q \leftrightarrow P \implies P \leftrightarrow Q$

*<proof>*

**lemma iff-trans:**  $[| P \leftrightarrow Q; Q \leftrightarrow R |] \implies P \leftrightarrow R$

*<proof>*

**lemma exI1:**

$P(a) \implies (!x. P(x) \implies x=a) \implies EX! x. P(x)$

*<proof>*

**lemma ex-exI1:**

$EX x. P(x) \implies (!x y. [| P(x); P(y) |] \implies x=y) \implies EX! x. P(x)$

*<proof>*

**lemma ex1E:**

$EX! x. P(x) \implies (!x. [| P(x); ALL y. P(y) \rightarrow y=x |] \implies R) \implies R$

*<proof>*

$\langle ML \rangle$

**lemma** *conj-cong*:

**assumes**  $P \leftrightarrow P'$   
**and**  $P' \implies Q \leftrightarrow Q'$   
**shows**  $(P \& Q) \leftrightarrow (P' \& Q')$   
 $\langle proof \rangle$

**lemma** *conj-cong2*:

**assumes**  $P \leftrightarrow P'$   
**and**  $P' \implies Q \leftrightarrow Q'$   
**shows**  $(Q \& P) \leftrightarrow (Q' \& P')$   
 $\langle proof \rangle$

**lemma** *disj-cong*:

**assumes**  $P \leftrightarrow P'$  **and**  $Q \leftrightarrow Q'$   
**shows**  $(P | Q) \leftrightarrow (P' | Q')$   
 $\langle proof \rangle$

**lemma** *imp-cong*:

**assumes**  $P \leftrightarrow P'$   
**and**  $P' \implies Q \leftrightarrow Q'$   
**shows**  $(P \dashrightarrow Q) \leftrightarrow (P' \dashrightarrow Q')$   
 $\langle proof \rangle$

**lemma** *iff-cong*:  $[[ P \leftrightarrow P'; Q \leftrightarrow Q' ]] \implies (P \leftrightarrow Q) \leftrightarrow (P' \leftrightarrow Q')$   
 $\langle proof \rangle$

**lemma** *not-cong*:  $P \leftrightarrow P' \implies \sim P \leftrightarrow \sim P'$   
 $\langle proof \rangle$

**lemma** *all-cong*:

**assumes**  $\forall x. P(x) \leftrightarrow Q(x)$   
**shows**  $(\text{ALL } x. P(x)) \leftrightarrow (\text{ALL } x. Q(x))$   
 $\langle proof \rangle$

**lemma** *ex-cong*:

**assumes**  $\forall x. P(x) \leftrightarrow Q(x)$   
**shows**  $(\text{EX } x. P(x)) \leftrightarrow (\text{EX } x. Q(x))$   
 $\langle proof \rangle$

**lemma** *ex1-cong*:

**assumes**  $\forall x. P(x) \leftrightarrow Q(x)$   
**shows**  $(\text{EX! } x. P(x)) \leftrightarrow (\text{EX! } x. Q(x))$   
 $\langle proof \rangle$

**lemma** *sym*:  $a=b \implies b=a$   
*<proof>*

**lemma** *trans*:  $[| a=b; b=c |] \implies a=c$   
*<proof>*

**lemma** *not-sym*:  $b \sim a \implies a \sim b$   
*<proof>*

**lemma** *def-imp-iff*:  $(A == B) \implies A <-> B$   
*<proof>*

**lemma** *meta-eq-to-obj-eq*:  $(A == B) \implies A = B$   
*<proof>*

**lemma** *meta-eq-to-iff*:  $x==y \implies x<->y$   
*<proof>*

**lemma** *ssubst*:  $[| b = a; P(a) |] \implies P(b)$   
*<proof>*

**lemma** *ex1-equalsE*:  
 $[| EX! x. P(x); P(a); P(b) |] \implies a=b$   
*<proof>*

**lemma** *subst-context*:  $[| a=b |] \implies t(a)=t(b)$   
*<proof>*

**lemma** *subst-context2*:  $[| a=b; c=d |] \implies t(a,c)=t(b,d)$   
*<proof>*

**lemma** *subst-context3*:  $[| a=b; c=d; e=f |] \implies t(a,c,e)=t(b,d,f)$   
*<proof>*

**lemma** *box-equals*:  $[| a=b; a=c; b=d |] \implies c=d$   
*<proof>*

**lemma** *simp-equals*:  $[[ a=c; b=d; c=d ]] ==> a=b$   
*<proof>*

**lemma** *pred1-cong*:  $a=a' ==> P(a) <-> P(a')$   
*<proof>*

**lemma** *pred2-cong*:  $[[ a=a'; b=b' ]] ==> P(a,b) <-> P(a',b')$   
*<proof>*

**lemma** *pred3-cong*:  $[[ a=a'; b=b'; c=c' ]] ==> P(a,b,c) <-> P(a',b',c')$   
*<proof>*

**lemma** *eq-cong*:  $[[ a = a'; b = b' ]] ==> a = b <-> a' = b'$   
*<proof>*

**lemma** *conj-impE*:  
 **assumes** *major*:  $(P \& Q) \dashrightarrow S$   
 **and** *r*:  $P \dashrightarrow (Q \dashrightarrow S) ==> R$   
 **shows** *R*  
*<proof>*

**lemma** *disj-impE*:  
 **assumes** *major*:  $(P | Q) \dashrightarrow S$   
 **and** *r*:  $[[ P \dashrightarrow S; Q \dashrightarrow S ]] ==> R$   
 **shows** *R*  
*<proof>*

**lemma** *imp-impE*:  
 **assumes** *major*:  $(P \dashrightarrow Q) \dashrightarrow S$   
 **and** *r1*:  $[[ P; Q \dashrightarrow S ]] ==> Q$   
 **and** *r2*:  $S ==> R$   
 **shows** *R*  
*<proof>*

**lemma** *not-impE*:  
 $\sim P \dashrightarrow S \implies (P \implies \text{False}) \implies (S \implies R) \implies R$   
*<proof>*

**lemma** *iff-impE*:  
 **assumes** *major*:  $(P <-> Q) \dashrightarrow S$

**and**  $r1: \llbracket P; Q \dashrightarrow S \rrbracket \implies Q$   
**and**  $r2: \llbracket Q; P \dashrightarrow S \rrbracket \implies P$   
**and**  $r3: S \implies R$   
**shows**  $R$   
 $\langle proof \rangle$

**lemma** *all-impE*:  
**assumes** *major*:  $(\text{ALL } x. P(x)) \dashrightarrow S$   
**and**  $r1: \forall x. P(x)$   
**and**  $r2: S \implies R$   
**shows**  $R$   
 $\langle proof \rangle$

**lemma** *ex-impE*:  
**assumes** *major*:  $(\text{EX } x. P(x)) \dashrightarrow S$   
**and**  $r: P(x) \dashrightarrow S \implies R$   
**shows**  $R$   
 $\langle proof \rangle$

**lemma** *disj-imp-disj*:  
 $P \mid Q \implies (P \implies R) \implies (Q \implies S) \implies R \mid S$   
 $\langle proof \rangle$

$\langle ML \rangle$

**lemma** *thin-refl*:  $\llbracket x=x; \text{PROP } W \rrbracket \implies \text{PROP } W$   $\langle proof \rangle$

$\langle ML \rangle$

### 1.3 Intuitionistic Reasoning

$\langle ML \rangle$

**lemma** *impE'*:  
**assumes**  $1: P \dashrightarrow Q$   
**and**  $2: Q \implies R$   
**and**  $3: P \dashrightarrow Q \implies P$   
**shows**  $R$   
 $\langle proof \rangle$

**lemma** *allE'*:  
**assumes**  $1: \text{ALL } x. P(x)$   
**and**  $2: P(x) \implies \text{ALL } x. P(x) \implies Q$   
**shows**  $Q$   
 $\langle proof \rangle$

**lemma** *notE'*:  
**assumes** 1:  $\sim P$   
**and** 2:  $\sim P \implies P$   
**shows**  $R$   
 $\langle proof \rangle$

**lemmas** [*Pure.elim!*] = *disjE iffE FalseE conjE exE*  
**and** [*Pure.intro!*] = *iffI conjI impI TrueI notI allI refl*  
**and** [*Pure.elim 2*] = *allE notE' impE'*  
**and** [*Pure.intro*] = *exI disjI2 disjI1*

$\langle ML \rangle$

**lemma** *iff-not-sym*:  $\sim (Q \leftrightarrow P) \implies \sim (P \leftrightarrow Q)$   
 $\langle proof \rangle$

**lemmas** [*sym*] = *sym iff-sym not-sym iff-not-sym*  
**and** [*Pure.elim?*] = *iffD1 iffD2 impE*

**lemma** *eq-commute*:  $a=b \leftrightarrow b=a$   
 $\langle proof \rangle$

## 1.4 Atomizing meta-level rules

**lemma** *atomize-all* [*atomize*]:  $(!!x. P(x)) \implies Trueprop (ALL x. P(x))$   
 $\langle proof \rangle$

**lemma** *atomize-imp* [*atomize*]:  $(A \implies B) \implies Trueprop (A \dashrightarrow B)$   
 $\langle proof \rangle$

**lemma** *atomize-eq* [*atomize*]:  $(x == y) \implies Trueprop (x = y)$   
 $\langle proof \rangle$

**lemma** *atomize-iff* [*atomize*]:  $(A == B) \implies Trueprop (A \leftrightarrow B)$   
 $\langle proof \rangle$

**lemma** *atomize-conj* [*atomize*]:  $(A \&\&\& B) \implies Trueprop (A \& B)$   
 $\langle proof \rangle$

**lemmas** [*symmetric, rulify*] = *atomize-all atomize-imp*  
**and** [*symmetric, defn*] = *atomize-all atomize-imp atomize-eq atomize-iff*

## 1.5 Atomizing elimination rules

$\langle ML \rangle$

**lemma** *atomize-exL*[*atomize-elim*]:  $(!!x. P(x) ==> Q) == ((EX x. P(x)) ==> Q)$   
 ⟨*proof*⟩

**lemma** *atomize-conjL*[*atomize-elim*]:  $(A ==> B ==> C) == (A \& B ==> C)$   
 ⟨*proof*⟩

**lemma** *atomize-disjL*[*atomize-elim*]:  $((A ==> C) ==> (B ==> C)) ==> C$   
 $== ((A | B ==> C) ==> C)$   
 ⟨*proof*⟩

**lemma** *atomize-elimL*[*atomize-elim*]:  $(!!B. (A ==> B) ==> B) == Trueprop(A)$   
 ⟨*proof*⟩

## 1.6 Calculational rules

**lemma** *forw-subst*:  $a = b ==> P(b) ==> P(a)$   
 ⟨*proof*⟩

**lemma** *back-subst*:  $P(a) ==> a = b ==> P(b)$   
 ⟨*proof*⟩

Note that this list of rules is in reverse order of priorities.

**lemmas** *basic-trans-rules* [*trans*] =  
*forw-subst*  
*back-subst*  
*rev-mp*  
*mp*  
*trans*

## 1.7 “Let” declarations

**nonterminal** *letbinds* and *letbind*

**definition** *Let* :: [*a*::{*s*}, '*a* => '*b*] => ('*b*::{*s*}) **where**  
 $Let(s, f) == f(s)$

**syntax**

*-bind* :: [*pttrn*, '*a*] => *letbind* ((*2*- =/ -) 10)  
 :: *letbind* => *letbinds* (-)  
*-binds* :: [*letbind*, *letbinds*] => *letbinds* (-;/ -)  
*-Let* :: [*letbinds*, '*a*] => '*a* ((*let* (-)/ *in* (-)) 10)

**translations**

$-Let(-binds(b, bs), e) == -Let(b, -Let(bs, e))$   
 $let\ x = a\ in\ e == CONST\ Let(a, \%x.\ e)$

**lemma** *LetI*:

**assumes**  $!!x. x=t \implies P(u(x))$   
**shows**  $P(\text{let } x=t \text{ in } u(x))$   
 $\langle \text{proof} \rangle$

## 1.8 Intuitionistic simplification rules

**lemma** *conj-simps*:

$P \ \& \ \text{True} \ \longleftrightarrow \ P$   
 $\text{True} \ \& \ P \ \longleftrightarrow \ P$   
 $P \ \& \ \text{False} \ \longleftrightarrow \ \text{False}$   
 $\text{False} \ \& \ P \ \longleftrightarrow \ \text{False}$   
 $P \ \& \ P \ \longleftrightarrow \ P$   
 $P \ \& \ P \ \& \ Q \ \longleftrightarrow \ P \ \& \ Q$   
 $P \ \& \ \sim P \ \longleftrightarrow \ \text{False}$   
 $\sim P \ \& \ P \ \longleftrightarrow \ \text{False}$   
 $(P \ \& \ Q) \ \& \ R \ \longleftrightarrow \ P \ \& \ (Q \ \& \ R)$   
 $\langle \text{proof} \rangle$

**lemma** *disj-simps*:

$P \ | \ \text{True} \ \longleftrightarrow \ \text{True}$   
 $\text{True} \ | \ P \ \longleftrightarrow \ \text{True}$   
 $P \ | \ \text{False} \ \longleftrightarrow \ P$   
 $\text{False} \ | \ P \ \longleftrightarrow \ P$   
 $P \ | \ P \ \longleftrightarrow \ P$   
 $P \ | \ P \ | \ Q \ \longleftrightarrow \ P \ | \ Q$   
 $(P \ | \ Q) \ | \ R \ \longleftrightarrow \ P \ | \ (Q \ | \ R)$   
 $\langle \text{proof} \rangle$

**lemma** *not-simps*:

$\sim(P|Q) \ \longleftrightarrow \ \sim P \ \& \ \sim Q$   
 $\sim \text{False} \ \longleftrightarrow \ \text{True}$   
 $\sim \text{True} \ \longleftrightarrow \ \text{False}$   
 $\langle \text{proof} \rangle$

**lemma** *imp-simps*:

$(P \ \longrightarrow \ \text{False}) \ \longleftrightarrow \ \sim P$   
 $(P \ \longrightarrow \ \text{True}) \ \longleftrightarrow \ \text{True}$   
 $(\text{False} \ \longrightarrow \ P) \ \longleftrightarrow \ \text{True}$   
 $(\text{True} \ \longrightarrow \ P) \ \longleftrightarrow \ P$   
 $(P \ \longrightarrow \ P) \ \longleftrightarrow \ \text{True}$   
 $(P \ \longrightarrow \ \sim P) \ \longleftrightarrow \ \sim P$   
 $\langle \text{proof} \rangle$

**lemma** *iff-simps*:

$(\text{True} \ \longleftrightarrow \ P) \ \longleftrightarrow \ P$   
 $(P \ \longleftrightarrow \ \text{True}) \ \longleftrightarrow \ P$   
 $(P \ \longleftrightarrow \ P) \ \longleftrightarrow \ \text{True}$   
 $(\text{False} \ \longleftrightarrow \ P) \ \longleftrightarrow \ \sim P$   
 $(P \ \longleftrightarrow \ \text{False}) \ \longleftrightarrow \ \sim P$

$\langle proof \rangle$

**lemma** *quant-simps*:

$!!P. (ALL x. P) \leftrightarrow P$   
 $(ALL x. x=t \rightarrow P(x)) \leftrightarrow P(t)$   
 $(ALL x. t=x \rightarrow P(x)) \leftrightarrow P(t)$   
 $!!P. (EX x. P) \leftrightarrow P$   
 $EX x. x=t$   
 $EX x. t=x$   
 $(EX x. x=t \ \& \ P(x)) \leftrightarrow P(t)$   
 $(EX x. t=x \ \& \ P(x)) \leftrightarrow P(t)$   
 $\langle proof \rangle$

**lemma** *distrib-simps*:

$P \ \& \ (Q \ | \ R) \leftrightarrow P \ \& \ Q \ | \ P \ \& \ R$   
 $(Q \ | \ R) \ \& \ P \leftrightarrow Q \ \& \ P \ | \ R \ \& \ P$   
 $(P \ | \ Q \ \rightarrow \ R) \leftrightarrow (P \ \rightarrow \ R) \ \& \ (Q \ \rightarrow \ R)$   
 $\langle proof \rangle$

Conversion into rewrite rules

**lemma** *P-iff-F*:  $\sim P \implies (P \leftrightarrow False)$   $\langle proof \rangle$

**lemma** *iff-reflection-F*:  $\sim P \implies (P == False)$   $\langle proof \rangle$

**lemma** *P-iff-T*:  $P \implies (P \leftrightarrow True)$   $\langle proof \rangle$

**lemma** *iff-reflection-T*:  $P \implies (P == True)$   $\langle proof \rangle$

More rewrite rules

**lemma** *conj-commute*:  $P \ \& \ Q \leftrightarrow Q \ \& \ P$   $\langle proof \rangle$

**lemma** *conj-left-commute*:  $P \ \& \ (Q \ \& \ R) \leftrightarrow Q \ \& \ (P \ \& \ R)$   $\langle proof \rangle$

**lemmas** *conj-comms* = *conj-commute conj-left-commute*

**lemma** *disj-commute*:  $P \ | \ Q \leftrightarrow Q \ | \ P$   $\langle proof \rangle$

**lemma** *disj-left-commute*:  $P \ | \ (Q \ | \ R) \leftrightarrow Q \ | \ (P \ | \ R)$   $\langle proof \rangle$

**lemmas** *disj-comms* = *disj-commute disj-left-commute*

**lemma** *conj-disj-distribL*:  $P \ \& \ (Q \ | \ R) \leftrightarrow (P \ \& \ Q \ | \ P \ \& \ R)$   $\langle proof \rangle$

**lemma** *conj-disj-distribR*:  $(P \ | \ Q) \ \& \ R \leftrightarrow (P \ \& \ R \ | \ Q \ \& \ R)$   $\langle proof \rangle$

**lemma** *disj-conj-distribL*:  $P \ | \ (Q \ \& \ R) \leftrightarrow (P \ | \ Q) \ \& \ (P \ | \ R)$   $\langle proof \rangle$

**lemma** *disj-conj-distribR*:  $(P \ \& \ Q) \ | \ R \leftrightarrow (P \ | \ R) \ \& \ (Q \ | \ R)$   $\langle proof \rangle$

**lemma** *imp-conj-distrib*:  $(P \ \rightarrow \ (Q \ \& \ R)) \leftrightarrow (P \ \rightarrow \ Q) \ \& \ (P \ \rightarrow \ R)$   $\langle proof \rangle$

**lemma** *imp-conj*:  $((P \ \& \ Q) \ \rightarrow \ R) \leftrightarrow (P \ \rightarrow \ (Q \ \rightarrow \ R))$   $\langle proof \rangle$

**lemma** *imp-disj*:  $(P \ | \ Q \ \rightarrow \ R) \leftrightarrow (P \ \rightarrow \ R) \ \& \ (Q \ \rightarrow \ R)$   $\langle proof \rangle$

**lemma** *de-Morgan-disj*:  $(\sim(P \ | \ Q)) \leftrightarrow (\sim P \ \& \ \sim Q)$   $\langle proof \rangle$

```

lemma not-ex: ( $\sim (EX\ x.\ P(x))$ )  $\leftrightarrow$  ( $ALL\ x.\ \sim P(x)$ )  $\langle proof \rangle$ 
lemma imp-ex: ( $(EX\ x.\ P(x)) \dashrightarrow Q$ )  $\leftrightarrow$  ( $ALL\ x.\ P(x) \dashrightarrow Q$ )  $\langle proof \rangle$ 

lemma ex-disj-distrib:
  ( $EX\ x.\ P(x) \mid Q(x)$ )  $\leftrightarrow$  ( $(EX\ x.\ P(x)) \mid (EX\ x.\ Q(x))$ )  $\langle proof \rangle$ 

lemma all-conj-distrib:
  ( $ALL\ x.\ P(x) \ \&\ Q(x)$ )  $\leftrightarrow$  ( $(ALL\ x.\ P(x)) \ \&\ (ALL\ x.\ Q(x))$ )  $\langle proof \rangle$ 

end

```

## 2 Classical first-order logic

```

theory FOL
imports IFOL
uses
   $\sim$ /src/Provers/classical.ML
   $\sim$ /src/Provers/blast.ML
   $\sim$ /src/Provers/clasimp.ML
   $\sim$ /src/Tools/induct.ML
   $\sim$ /src/Tools/case-product.ML
  (simpdata.ML)
begin

```

### 2.1 The classical axiom

```

axiomatization where
  classical: ( $\sim P \implies P$ )  $\implies P$ 

```

### 2.2 Lemmas and proof tools

```

lemma ccontr: ( $\neg P \implies False$ )  $\implies P$ 
   $\langle proof \rangle$ 

```

```

lemma disjCI: ( $\sim Q \implies P$ )  $\implies P \mid Q$ 
   $\langle proof \rangle$ 

```

```

lemma ex-classical:
  assumes r:  $\sim (EX\ x.\ P(x)) \implies P(a)$ 
  shows  $EX\ x.\ P(x)$ 
   $\langle proof \rangle$ 

```

```

lemma exCI:
  assumes r:  $ALL\ x.\ \sim P(x) \implies P(a)$ 

```

**shows**  $EX x. P(x)$   
*<proof>*

**lemma** *excluded-middle*:  $\sim P \mid P$   
*<proof>*

**lemma** *case-split* [*case-names True False*]:  
**assumes**  $r1: P \implies Q$   
**and**  $r2: \sim P \implies Q$   
**shows**  $Q$   
*<proof>*

*<ML>*

**lemma** *impCE*:  
**assumes**  $major: P \dashrightarrow Q$   
**and**  $r1: \sim P \implies R$   
**and**  $r2: Q \implies R$   
**shows**  $R$   
*<proof>*

**lemma** *impCE'*:  
**assumes**  $major: P \dashrightarrow Q$   
**and**  $r1: Q \implies R$   
**and**  $r2: \sim P \implies R$   
**shows**  $R$   
*<proof>*

**lemma** *notnotD*:  $\sim\sim P \implies P$   
*<proof>*

**lemma** *contrapos2*:  $[[ Q; \sim P \implies \sim Q ]] \implies P$   
*<proof>*

**lemma** *iffCE*:  
**assumes**  $major: P \leftrightarrow Q$   
**and**  $r1: [[ P; Q ]] \implies R$   
**and**  $r2: [[ \sim P; \sim Q ]] \implies R$   
**shows**  $R$

*<proof>*

**lemma** *alt-ex1E*:

**assumes** *major*:  $EX! x. P(x)$

**and** *r*:  $!!x. [ P(x); ALL y y'. P(y) \ \& \ P(y') \ \longrightarrow \ y=y' ] \implies R$

**shows** *R*

*<proof>*

**lemma** *imp-elim*:  $P \longrightarrow Q \implies (\sim R \implies P) \implies (Q \implies R) \implies R$

*<proof>*

**lemma** *swap*:  $\sim P \implies (\sim R \implies P) \implies R$

*<proof>*

### 3 Classical Reasoner

*<ML>*

**lemmas** [*intro!*] = *refl TrueI conjI disjCI impI notI iffI*

**and** [*elim!*] = *conjE disjE impCE FalseE iffCE*

*<ML>*

**lemmas** [*intro!*] = *allI ex-ex1I*

**and** [*intro*] = *exI*

**and** [*elim!*] = *exE alt-ex1E*

**and** [*elim*] = *allE*

*<ML>*

**lemma** *ex1-functional*:  $[ [ EX! z. P(a,z); P(a,b); P(a,c) ] ] \implies b = c$

*<proof>*

**lemma** *True-implies-equals*:  $(True \implies PROP P) == PROP P$

*<proof>*

**lemma** *uncurry*:  $P \longrightarrow Q \longrightarrow R \implies P \ \& \ Q \longrightarrow R$

*<proof>*

**lemma** *iff-allI*:  $(!!x. P(x) \longleftrightarrow Q(x)) \implies (ALL x. P(x)) \longleftrightarrow (ALL x. Q(x))$

*<proof>*

**lemma** *iff-exI*:  $(!!x. P(x) \longleftrightarrow Q(x)) \implies (EX x. P(x)) \longleftrightarrow (EX x. Q(x))$

*<proof>*

**lemma** *all-comm*:  $(\text{ALL } x \ y. P(x,y)) \leftrightarrow (\text{ALL } y \ x. P(x,y))$  *<proof>*

**lemma** *ex-comm*:  $(\text{EX } x \ y. P(x,y)) \leftrightarrow (\text{EX } y \ x. P(x,y))$  *<proof>*

**lemma** *cases-simp*:  $(P \rightarrow Q) \ \& \ (\sim P \rightarrow Q) \leftrightarrow Q$  *<proof>*

**lemma** *int-ex-simps*:

!!P Q.  $(\text{EX } x. P(x) \ \& \ Q) \leftrightarrow (\text{EX } x. P(x)) \ \& \ Q$   
!!P Q.  $(\text{EX } x. P \ \& \ Q(x)) \leftrightarrow P \ \& \ (\text{EX } x. Q(x))$   
!!P Q.  $(\text{EX } x. P(x) \ | \ Q) \leftrightarrow (\text{EX } x. P(x)) \ | \ Q$   
!!P Q.  $(\text{EX } x. P \ | \ Q(x)) \leftrightarrow P \ | \ (\text{EX } x. Q(x))$   
*<proof>*

**lemma** *cla-ex-simps*:

!!P Q.  $(\text{EX } x. P(x) \ \rightarrow \ Q) \leftrightarrow (\text{ALL } x. P(x)) \ \rightarrow \ Q$   
!!P Q.  $(\text{EX } x. P \ \rightarrow \ Q(x)) \leftrightarrow P \ \rightarrow \ (\text{EX } x. Q(x))$   
*<proof>*

**lemmas** *ex-simps = int-ex-simps cla-ex-simps*

**lemma** *int-all-simps*:

!!P Q.  $(\text{ALL } x. P(x) \ \& \ Q) \leftrightarrow (\text{ALL } x. P(x)) \ \& \ Q$   
!!P Q.  $(\text{ALL } x. P \ \& \ Q(x)) \leftrightarrow P \ \& \ (\text{ALL } x. Q(x))$   
!!P Q.  $(\text{ALL } x. P(x) \ \rightarrow \ Q) \leftrightarrow (\text{EX } x. P(x)) \ \rightarrow \ Q$   
!!P Q.  $(\text{ALL } x. P \ \rightarrow \ Q(x)) \leftrightarrow P \ \rightarrow \ (\text{ALL } x. Q(x))$   
*<proof>*

**lemma** *cla-all-simps*:

!!P Q.  $(\text{ALL } x. P(x) \ | \ Q) \leftrightarrow (\text{ALL } x. P(x)) \ | \ Q$   
!!P Q.  $(\text{ALL } x. P \ | \ Q(x)) \leftrightarrow P \ | \ (\text{ALL } x. Q(x))$   
*<proof>*

**lemmas** *all-simps = int-all-simps cla-all-simps*

**lemma** *imp-disj1*:  $(P \dashrightarrow Q) \mid R \leftrightarrow (P \dashrightarrow Q \mid R)$  *<proof>*

**lemma** *imp-disj2*:  $Q \mid (P \dashrightarrow R) \leftrightarrow (P \dashrightarrow Q \mid R)$  *<proof>*

**lemma** *de-Morgan-conj*:  $(\sim(P \ \& \ Q)) \leftrightarrow (\sim P \mid \sim Q)$  *<proof>*

**lemma** *not-imp*:  $\sim(P \dashrightarrow Q) \leftrightarrow (P \ \& \ \sim Q)$  *<proof>*

**lemma** *not-iff*:  $\sim(P \leftrightarrow Q) \leftrightarrow (P \leftrightarrow \sim Q)$  *<proof>*

**lemma** *not-all*:  $(\sim(\text{ALL } x. P(x))) \leftrightarrow (\text{EX } x. \sim P(x))$  *<proof>*

**lemma** *imp-all*:  $((\text{ALL } x. P(x)) \dashrightarrow Q) \leftrightarrow (\text{EX } x. P(x) \dashrightarrow Q)$  *<proof>*

**lemmas** *meta-simps* =

*triv-forall-equality*

*True-implies-equals*

**lemmas** *IFOL-simps* =

*refl [THEN P-iff-T] conj-simps disj-simps not-simps*

*imp-simps iff-simps quant-simps*

**lemma** *notFalseI*:  $\sim \text{False}$  *<proof>*

**lemma** *cla-simps-misc*:

$\sim(P \ \& \ Q) \leftrightarrow \sim P \mid \sim Q$

$P \mid \sim P$

$\sim P \mid P$

$\sim \sim P \leftrightarrow P$

$(\sim P \dashrightarrow P) \leftrightarrow P$

$(\sim P \leftrightarrow \sim Q) \leftrightarrow (P \leftrightarrow Q)$  *<proof>*

**lemmas** *cla-simps* =

*de-Morgan-conj de-Morgan-disj imp-disj1 imp-disj2*

*not-imp not-all not-ex cases-simp cla-simps-misc*

*<ML>*

### 3.1 Other simple lemmas

**lemma** [*simp*]:  $((P \dashrightarrow R) \leftrightarrow (Q \dashrightarrow R)) \leftrightarrow ((P \leftrightarrow Q) \mid R)$   
*<proof>*

**lemma** [*simp*]:  $((P \dashrightarrow Q) \leftrightarrow (P \dashrightarrow R)) \leftrightarrow (P \dashrightarrow (Q \leftrightarrow R))$   
*<proof>*

**lemma** *not-disj-iff-imp*:  $\sim P \mid Q \leftrightarrow (P \dashrightarrow Q)$   
*<proof>*

**lemma** *conj-mono*:  $[| P1 \dashrightarrow Q1; P2 \dashrightarrow Q2 |] \implies (P1 \& P2) \dashrightarrow (Q1 \& Q2)$   
*<proof>*

**lemma** *disj-mono*:  $[| P1 \dashrightarrow Q1; P2 \dashrightarrow Q2 |] \implies (P1 | P2) \dashrightarrow (Q1 | Q2)$   
*<proof>*

**lemma** *imp-mono*:  $[| Q1 \dashrightarrow P1; P2 \dashrightarrow Q2 |] \implies (P1 \dashrightarrow P2) \dashrightarrow (Q1 \dashrightarrow Q2)$   
*<proof>*

**lemma** *imp-refl*:  $P \dashrightarrow P$   
*<proof>*

**lemma** *ex-mono*:  $(\exists x. P(x) \dashrightarrow Q(x)) \implies (EX x. P(x)) \dashrightarrow (EX x. Q(x))$   
*<proof>*

**lemma** *all-mono*:  $(\forall x. P(x) \dashrightarrow Q(x)) \implies (ALL x. P(x)) \dashrightarrow (ALL x. Q(x))$   
*<proof>*

### 3.2 Proof by cases and induction

Proper handling of non-atomic rule statements.

**definition** *induct-forall*( $P$ ) ==  $\forall x. P(x)$

**definition** *induct-implies*( $A, B$ ) ==  $A \longrightarrow B$

**definition** *induct-equal*( $x, y$ ) ==  $x = y$

**definition** *induct-conj*( $A, B$ ) ==  $A \wedge B$

**lemma** *induct-forall-eq*:  $(\exists x. P(x)) \implies \text{Trueprop}(\text{induct-forall}(\lambda x. P(x)))$   
*<proof>*

**lemma** *induct-implies-eq*:  $(A \implies B) \implies \text{Trueprop}(\text{induct-implies}(A, B))$   
*<proof>*

**lemma** *induct-equal-eq*:  $(x == y) \implies \text{Trueprop}(\text{induct-equal}(x, y))$   
*<proof>*

**lemma** *induct-conj-eq*:  $(A \&\& B) \implies \text{Trueprop}(\text{induct-conj}(A, B))$   
*<proof>*

**lemmas** *induct-atomize* = *induct-forall-eq induct-implies-eq induct-equal-eq induct-conj-eq*

**lemmas** *induct-rulify* [*symmetric, standard*] = *induct-atomize*

**lemmas** *induct-rulify-fallback* =

*induct-forall-def induct-implies-def induct-equal-def induct-conj-def*

**hide-const** *induct-forall induct-implies induct-equal induct-conj*

Method setup.

$\langle ML \rangle$   
**declare** *case-split* [*cases type: o*]

$\langle ML \rangle$

**hide-const** (**open**) *eq*

**end**