

# Isabelle/FOL — First-Order Logic

Larry Paulson and Markus Wenzel

May 22, 2012

## Contents

<b>1</b>	<b>Intuitionistic first-order logic</b>	<b>1</b>
1.1	Syntax and axiomatic basis . . . . .	2
1.1.1	Equality . . . . .	2
1.1.2	Propositional logic . . . . .	2
1.1.3	Quantifiers . . . . .	3
1.1.4	Definitions . . . . .	3
1.1.5	Additional notation . . . . .	3
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	Calculational rules . . . . .	12
1.7	“Let” declarations . . . . .	12
1.8	Intuitionistic simplification rules . . . . .	12
<b>2</b>	<b>Classical first-order logic</b>	<b>14</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*    ::  $o \Rightarrow prop$                     ((-) 5)

### 1.1.1 Equality

**axiomatization**  
*eq* :: [*a*, *a*]  $\Rightarrow o$  (**infixl** = 50)

**where**

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

### 1.1.2 Propositional logic

**axiomatization**

*False* :: *o* **and**  
*conj* :: [*o*, *o*]  $\Rightarrow o$  (**infixr** & 35) **and**  
*disj* :: [*o*, *o*]  $\Rightarrow o$  (**infixr** | 30) **and**  
*imp* :: [*o*, *o*]  $\Rightarrow o$  (**infixr**  $\longrightarrow$  25)

**where**

*conjI*: [| *P*; *Q* |]  $\Longrightarrow P \& Q$  **and**  
*conjunct1*:  $P \& Q \Longrightarrow P$  **and**  
*conjunct2*:  $P \& Q \Longrightarrow Q$  **and**  
  
*disjI1*:  $P \Longrightarrow P | Q$  **and**  
*disjI2*:  $Q \Longrightarrow P | Q$  **and**  
*disjE*: [|  $P | Q$ ;  $P \Longrightarrow R$ ;  $Q \Longrightarrow R$  |]  $\Longrightarrow R$  **and**

*impI*:  $(P \Longrightarrow Q) \Longrightarrow P \longrightarrow Q$  **and**  
*mp*: [|  $P \longrightarrow Q$ ; *P* |]  $\Longrightarrow Q$  **and**

*FalseE*:  $False ==> P$

### 1.1.3 Quantifiers

#### axiomatization

*All* ::  $('a ==> o) ==> o$  (**binder ALL 10**) **and**

*Ex* ::  $('a ==> o) ==> o$  (**binder EX 10**)

#### where

*allI*:  $(!!x. P(x)) ==> (ALL x. P(x))$  **and**

*spec*:  $(ALL x. P(x)) ==> P(x)$  **and**

*exI*:  $P(x) ==> (EX x. P(x))$  **and**

*exE*:  $[ EX x. P(x); !!x. P(x) ==> R ] ==> R$

### 1.1.4 Definitions

**definition** *True* ==  $False --> False$

**definition** *Not* ( $\sim$  - [40] 40) **where** *not-def*:  $\sim P == P --> False$

**definition** *iff* (**infixr**  $<->$  25) **where**  $P <-> Q == (P --> Q) \& (Q --> P)$

**definition** *Ex1* ::  $('a ==> o) ==> o$  (**binder EX! 10**)

**where** *ex1-def*:  $EX! x. P(x) == EX x. P(x) \& (ALL y. P(y) --> y=x)$

**axiomatization where** — Reflection, admissible

*eq-reflection*:  $(x=y) ==> (x==y)$  **and**

*iff-reflection*:  $(P <-> Q) ==> (P==Q)$

### 1.1.5 Additional notation

**abbreviation** *not-equal* ::  $[ 'a, 'a ] ==> o$  (**infixl**  $\sim =$  50)

**where**  $x \sim = y == \sim (x = y)$

**notation** (*xsymbols*)

*not-equal* (**infixl**  $\neq$  50)

**notation** (*HTML output*)

*not-equal* (**infixl**  $\neq$  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)

## 1.2 Lemmas and proof tools

**lemmas** *strip* = *impI all*

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

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

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

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

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

**lemma** *notI*:  $(P \implies False) \implies \sim P$   
*<proof>*

**lemma** *notE*:  $[ [ \sim P; P ] ] \implies R$

*<proof>*

**lemma** *rev-notE*:  $[[ P; \sim P ]] \implies R$   
*<proof>*

**lemma** *not-to-imp*:  
 **assumes**  $\sim P$   
 **and**  $r: P \longrightarrow \text{False} \implies Q$   
 **shows**  $Q$   
*<proof>*

**lemma** *rev-mp*:  $[[ P; P \longrightarrow Q ]] \implies Q$   
*<proof>*

**lemma** *contrapos*:  
 **assumes** *major*:  $\sim Q$   
 **and** *minor*:  $P \implies Q$   
 **shows**  $\sim P$   
*<proof>*

*<ML>*

**lemma** *iffI*:  $[[ P \implies Q; Q \implies P ]] \implies P \longleftrightarrow Q$   
*<proof>*

**lemma** *iffE*:  
 **assumes** *major*:  $P \longleftrightarrow Q$   
 **and**  $r: P \longrightarrow Q \implies Q \longrightarrow P \implies R$   
 **shows**  $R$   
*<proof>*

**lemma** *iffD1*:  $[[ P \longleftrightarrow Q; P ]] \implies Q$   
*<proof>*

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

*<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** *ex1I*:  
 $P(a) \implies (!x. P(x) \implies x=a) \implies EX! x. P(x)$   
*<proof>*

**lemma** *ex-ex1I*:  
 $EX x. P(x) \implies (!x y. [[ P(x); P(y) ]] \implies x=y) \implies EX! x. P(x)$   
*<proof>*

**lemma** *ex1E*:  
 $EX! x. P(x) \implies (!x. [[ P(x); ALL y. P(y) \longrightarrow y=x ]] \implies R) \implies R$   
*<proof>*

*<ML>*

**lemma** *conj-cong*:  
assumes  $P \leftrightarrow P'$   
and  $P' \implies Q \leftrightarrow Q'$   
shows  $(P \& Q) \leftrightarrow (P' \& Q')$   
*<proof>*

**lemma** *conj-cong2*:  
assumes  $P \leftrightarrow P'$   
and  $P' \implies Q \leftrightarrow Q'$

**shows**  $(Q \& P) \leftrightarrow (Q' \& P')$   
*<proof>*

**lemma** *disj-cong*:  
**assumes**  $P \leftrightarrow P'$  **and**  $Q \leftrightarrow Q'$   
**shows**  $(P | Q) \leftrightarrow (P' | Q')$   
*<proof>*

**lemma** *imp-cong*:  
**assumes**  $P \leftrightarrow P'$   
**and**  $P' \implies Q \leftrightarrow Q'$   
**shows**  $(P \dashrightarrow Q) \leftrightarrow (P' \dashrightarrow Q')$   
*<proof>*

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

**lemma** *not-cong*:  $P \leftrightarrow P' \implies \sim P \leftrightarrow \sim P'$   
*<proof>*

**lemma** *all-cong*:  
**assumes**  $\forall x. P(x) \leftrightarrow Q(x)$   
**shows**  $(\text{ALL } x. P(x)) \leftrightarrow (\text{ALL } x. Q(x))$   
*<proof>*

**lemma** *ex-cong*:  
**assumes**  $\forall x. P(x) \leftrightarrow Q(x)$   
**shows**  $(\text{EX } x. P(x)) \leftrightarrow (\text{EX } x. Q(x))$   
*<proof>*

**lemma** *ex1-cong*:  
**assumes**  $\forall x. P(x) \leftrightarrow Q(x)$   
**shows**  $(\text{EX! } x. P(x)) \leftrightarrow (\text{EX! } x. Q(x))$   
*<proof>*

**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*:  $(A == B) ==> A <-> B$   
*<proof>*

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

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

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

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

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

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

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

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

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

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

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

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

**lemma** *iff-impE*:  
assumes *major*:  $(P \leftrightarrow Q) \longrightarrow S$   
and *r1*:  $[[ P; Q \longrightarrow S ]] \implies Q$   
and *r2*:  $[[ Q; P \longrightarrow S ]] \implies P$   
and *r3*:  $S \implies R$   
shows *R*  
*<proof>*

**lemma** *all-impE*:  
assumes *major*:  $(\text{ALL } x. P(x)) \longrightarrow S$   
and *r1*:  $!!x. P(x)$   
and *r2*:  $S \implies R$   
shows *R*  
*<proof>*

**lemma** *ex-impE*:  
**assumes** *major*:  $(\exists x. P(x)) \dashrightarrow S$   
**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-reft*:  $[[x=x; PROP W]] \implies PROP W \langle proof \rangle$

$\langle ML \rangle$

### 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*:  $ALL x. P(x)$   
**and** *2*:  $P(x) \implies ALL x. P(x) \implies Q$   
**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*:  $\sim (Q \leftrightarrow P) \implies \sim (P \leftrightarrow Q)$   
*<proof>*

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

**lemma** *eq-commute*:  $a=b \leftrightarrow b=a$   
*<proof>*

## 1.4 Atomizing meta-level rules

**lemma** *atomize-all* [*atomize*]:  $(!!x. P(x)) \implies \text{Trueprop } (ALL\ x. P(x))$   
*<proof>*

**lemma** *atomize-imp* [*atomize*]:  $(A \implies B) \implies \text{Trueprop } (A \dashrightarrow B)$   
*<proof>*

**lemma** *atomize-eq* [*atomize*]:  $(x == y) \implies \text{Trueprop } (x = y)$   
*<proof>*

**lemma** *atomize-iff* [*atomize*]:  $(A == B) \implies \text{Trueprop } (A \leftrightarrow B)$   
*<proof>*

**lemma** *atomize-conj* [*atomize*]:  $(A \&\&\& B) \implies \text{Trueprop } (A \& B)$   
*<proof>*

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

## 1.5 Atomizing elimination rules

*<ML>*

**lemma** *atomize-exL*[*atomize-elim*]:  $(!!x. P(x) \implies Q) \implies ((EX\ x. P(x)) \implies Q)$   
*<proof>*

**lemma** *atomize-conjL*[*atomize-elim*]:  $(A \implies B \implies C) \implies (A \& B \implies C)$   
*<proof>*

**lemma** *atomize-disjL*[*atomize-elim*]:  $((A \implies C) \implies (B \implies C) \implies C) \implies ((A \mid B \implies C) \implies C)$   
*<proof>*

**lemma** *atomize-elimL*[*atomize-elim*]:  $(!!B. (A \implies B) \implies B) \implies \text{Trueprop}(A)$   
*<proof>*

## 1.6 Calculational rules

**lemma** *forw-subst*:  $a = b \implies P(b) \implies P(a)$   
*<proof>*

**lemma** *back-subst*:  $P(a) \implies a = b \implies P(b)$   
*<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*::{*'b*}] => [*'b*::{*'c*}] **where**  
 $Let(s, f) == f(s)$

**syntax**

*-bind* :: [*p*trn, *'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**  $P(\text{let } x=t \text{ in } u(x))$   
*<proof>*

## 1.8 Intuitionistic simplification rules

**lemma** *conj-simps*:

$P \ \& \ True \ \leftrightarrow \ P$   
 $True \ \& \ P \ \leftrightarrow \ P$   
 $P \ \& \ False \ \leftrightarrow \ False$   
 $False \ \& \ P \ \leftrightarrow \ False$   
 $P \ \& \ P \ \leftrightarrow \ P$   
 $P \ \& \ P \ \& \ Q \ \leftrightarrow \ P \ \& \ Q$   
 $P \ \& \ \sim P \ \leftrightarrow \ False$   
 $\sim P \ \& \ P \ \leftrightarrow \ False$

$(P \ \& \ Q) \ \& \ R \ \leftrightarrow \ P \ \& \ (Q \ \& \ R)$   
*<proof>*

**lemma** *disj-simps*:

$P \ | \ True \ \leftrightarrow \ True$   
 $True \ | \ P \ \leftrightarrow \ True$   
 $P \ | \ False \ \leftrightarrow \ P$   
 $False \ | \ P \ \leftrightarrow \ P$   
 $P \ | \ P \ \leftrightarrow \ P$   
 $P \ | \ P \ | \ Q \ \leftrightarrow \ P \ | \ Q$   
 $(P \ | \ Q) \ | \ R \ \leftrightarrow \ P \ | \ (Q \ | \ R)$   
*<proof>*

**lemma** *not-simps*:

$\sim(P \ | \ Q) \ \leftrightarrow \ \sim P \ \& \ \sim Q$   
 $\sim False \ \leftrightarrow \ True$   
 $\sim True \ \leftrightarrow \ False$   
*<proof>*

**lemma** *imp-simps*:

$(P \ \rightarrow \ False) \ \leftrightarrow \ \sim P$   
 $(P \ \rightarrow \ True) \ \leftrightarrow \ True$   
 $(False \ \rightarrow \ P) \ \leftrightarrow \ True$   
 $(True \ \rightarrow \ P) \ \leftrightarrow \ P$   
 $(P \ \rightarrow \ P) \ \leftrightarrow \ True$   
 $(P \ \rightarrow \ \sim P) \ \leftrightarrow \ \sim P$   
*<proof>*

**lemma** *iff-simps*:

$(True \ \leftrightarrow \ P) \ \leftrightarrow \ P$   
 $(P \ \leftrightarrow \ True) \ \leftrightarrow \ P$   
 $(P \ \leftrightarrow \ P) \ \leftrightarrow \ True$   
 $(False \ \leftrightarrow \ P) \ \leftrightarrow \ \sim P$   
 $(P \ \leftrightarrow \ False) \ \leftrightarrow \ \sim P$   
*<proof>*

**lemma** *quant-simps*:

$!!P. (ALL \ x. \ P) \ \leftrightarrow \ P$   
 $(ALL \ x. \ x=t \ \rightarrow \ P(x)) \ \leftrightarrow \ P(t)$   
 $(ALL \ x. \ t=x \ \rightarrow \ P(x)) \ \leftrightarrow \ P(t)$   
 $!!P. (EX \ x. \ P) \ \leftrightarrow \ P$   
 $EX \ x. \ x=t$   
 $EX \ x. \ t=x$   
 $(EX \ x. \ x=t \ \& \ P(x)) \ \leftrightarrow \ P(t)$   
 $(EX \ x. \ t=x \ \& \ P(x)) \ \leftrightarrow \ P(t)$   
*<proof>*

**lemma** *distrib-simps*:

$P \& (Q \mid R) \leftrightarrow P \& Q \mid P \& R$   
 $(Q \mid R) \& P \leftrightarrow Q \& P \mid R \& P$   
 $(P \mid Q \dashrightarrow R) \leftrightarrow (P \dashrightarrow R) \& (Q \dashrightarrow R)$   
*<proof>*

Conversion into rewrite rules

**lemma** *P-iff-F*:  $\sim P \implies (P \leftrightarrow \text{False})$  *<proof>*

**lemma** *iff-reflection-F*:  $\sim P \implies (P == \text{False})$  *<proof>*

**lemma** *P-iff-T*:  $P \implies (P \leftrightarrow \text{True})$  *<proof>*

**lemma** *iff-reflection-T*:  $P \implies (P == \text{True})$  *<proof>*

More rewrite rules

**lemma** *conj-commute*:  $P \& Q \leftrightarrow Q \& P$  *<proof>*

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

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

**lemma** *disj-commute*:  $P \mid Q \leftrightarrow Q \mid P$  *<proof>*

**lemma** *disj-left-commute*:  $P \mid (Q \mid R) \leftrightarrow Q \mid (P \mid R)$  *<proof>*

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

**lemma** *conj-disj-distribL*:  $P \& (Q \mid R) \leftrightarrow (P \& Q) \mid (P \& R)$  *<proof>*

**lemma** *conj-disj-distribR*:  $(P \mid Q) \& R \leftrightarrow (P \& R) \mid (Q \& R)$  *<proof>*

**lemma** *disj-conj-distribL*:  $P \mid (Q \& R) \leftrightarrow (P \mid Q) \& (P \mid R)$  *<proof>*

**lemma** *disj-conj-distribR*:  $(P \& Q) \mid R \leftrightarrow (P \mid R) \& (Q \mid R)$  *<proof>*

**lemma** *imp-conj-distrib*:  $(P \dashrightarrow (Q \& R)) \leftrightarrow (P \dashrightarrow Q) \& (P \dashrightarrow R)$  *<proof>*

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

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

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

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

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

**lemma** *ex-disj-distrib*:

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

**lemma** *all-conj-distrib*:

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

**end**

## 2 Classical first-order logic

**theory** *FOL*

```

imports IFOL
keywords print-claset print-induct-rules :: diag
uses
  ~~/src/Provers/classical.ML
  ~~/src/Provers/blast.ML
  ~~/src/Provers/clasimp.ML
  ~~/src/Tools/induct.ML
  ~~/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: ( $\neg P \implies \text{False}$ )  $\implies P$ 
  <proof>

```

```

lemma disjCI: ( $\sim Q \implies P$ )  $\implies P \mid Q$ 
  <proof>

```

```

lemma ex-classical:
  assumes r:  $\sim(\text{EX } x. P(x)) \implies P(a)$ 
  shows  $\text{EX } x. P(x)$ 
  <proof>

```

```

lemma exCI:
  assumes r:  $\text{ALL } x. \sim P(x) \implies P(a)$ 
  shows  $\text{EX } x. P(x)$ 
  <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**  $R$   
 $\langle$ *proof* $\rangle$

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

**lemma** *notnotD*:  $\sim\sim P \implies P$   
 $\langle$ *proof* $\rangle$

**lemma** *contrapos2*:  $[[ Q; \sim P \implies \sim Q ]] \implies P$   
 $\langle$ *proof* $\rangle$

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

**lemma** *alt-ex1E*:  
**assumes** *major*:  $EX! x. P(x)$   
**and** *r*:  $!!x. [[ P(x); ALL y y'. P(y) \ \& \ P(y') \dashrightarrow y=y' ]] \implies R$   
**shows**  $R$   
 $\langle$ *proof* $\rangle$

**lemma** *imp-elim*:  $P \dashrightarrow Q \implies (\sim R \implies P) \implies (Q \implies R) \implies R$   
 $\langle$ *proof* $\rangle$

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

### 3 Classical Reasoner

$\langle ML \rangle$

**lemmas**  $[intro!] = refl\ TrueI\ conjI\ disjCI\ impI\ notI\ iffI$   
**and**  $[elim!] = conjE\ disjE\ impCE\ FalseE\ iffCE$   
 $\langle ML \rangle$

**lemmas**  $[intro!] = allI\ ex-ex1I$   
**and**  $[intro] = exI$   
**and**  $[elim!] = exE\ alt-ex1E$   
**and**  $[elim] = allE$   
 $\langle ML \rangle$

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

**lemma** *True-implies-equals*:  $(True ==> PROP\ P) == PROP\ P$   
 $\langle proof \rangle$

**lemma** *uncurry*:  $P\ --> Q\ --> R ==> P\ \&\ Q\ --> R$   
 $\langle proof \rangle$

**lemma** *iff-allI*:  $(!\!x.\ P(x)\ <-> Q(x)) ==> (ALL\ x.\ P(x))\ <-> (ALL\ x.\ Q(x))$   
 $\langle proof \rangle$

**lemma** *iff-exI*:  $(!\!x.\ P(x)\ <-> Q(x)) ==> (EX\ x.\ P(x))\ <-> (EX\ x.\ Q(x))$   
 $\langle proof \rangle$

**lemma** *all-comm*:  $(ALL\ x\ y.\ P(x,y))\ <-> (ALL\ y\ x.\ P(x,y))\ \langle proof \rangle$

**lemma** *ex-comm*:  $(EX\ x\ y.\ P(x,y))\ <-> (EX\ y\ x.\ P(x,y))\ \langle proof \rangle$

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

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

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

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

$!!P Q. (EX x. P(x) \dashrightarrow Q) \leftrightarrow (ALL x. P(x)) \dashrightarrow Q$   
 $!!P Q. (EX x. P \dashrightarrow Q(x)) \leftrightarrow P \dashrightarrow (EX x. Q(x))$   
 <proof>

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

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

$!!P Q. (ALL x. P(x) \& Q) \leftrightarrow (ALL x. P(x)) \& Q$   
 $!!P Q. (ALL x. P \& Q(x)) \leftrightarrow P \& (ALL x. Q(x))$   
 $!!P Q. (ALL x. P(x) \dashrightarrow Q) \leftrightarrow (EX x. P(x)) \dashrightarrow Q$   
 $!!P Q. (ALL x. P \dashrightarrow Q(x)) \leftrightarrow P \dashrightarrow (ALL x. Q(x))$   
 <proof>

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

$!!P Q. (ALL x. P(x) | Q) \leftrightarrow (ALL x. P(x)) | Q$   
 $!!P Q. (ALL x. P | Q(x)) \leftrightarrow P | (ALL x. Q(x))$   
 <proof>

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

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

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

**lemma** *de-Morgan-conj*:  $(\sim(P \& Q)) \leftrightarrow (\sim P | \sim Q)$  <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*:  $(\sim (ALL x. P(x))) \leftrightarrow (EX x. \sim P(x))$  <proof>

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

**lemmas** *meta-simps =*

*triv-forall-equality*

*True-implies-equals*

**lemmas** *IFOL-simps* =  
*reft [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*:  $[ [ P1 \dashrightarrow Q1; P2 \dashrightarrow Q2 ] ] \implies (P1 \mid P2) \dashrightarrow (Q1 \mid Q2)$   
*<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*:  $(!!x. P(x) \dashrightarrow Q(x)) \implies (EX x. P(x)) \dashrightarrow (EX x. Q(x))$   
*<proof>*

**lemma** *all-mono*:  $(!!x. P(x) \dashrightarrow Q(x)) \implies (ALL\ x. P(x) \dashrightarrow (ALL\ x. Q(x)))$   
 $\langle proof \rangle$

### 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*:  $(!!x. P(x)) \implies Trueprop(induct-forall(\lambda x. P(x)))$   
 $\langle proof \rangle$

**lemma** *induct-implies-eq*:  $(A \implies B) \implies Trueprop(induct-implies(A, B))$   
 $\langle proof \rangle$

**lemma** *induct-equal-eq*:  $(x == y) \implies Trueprop(induct-equal(x, y))$   
 $\langle proof \rangle$

**lemma** *induct-conj-eq*:  $(A \&\&\& B) \implies Trueprop(induct-conj(A, B))$   
 $\langle proof \rangle$

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

**lemmas** *induct-rulify* [*symmetric*] = *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**