Miscellaneous FOL Examples

June 8, 2008

Contents

1 A simple formulation of First-Order Logic

theory *First-Order-Logic* **imports** *Pure* **begin**

The subsequent theory development illustrates single-sorted intuitionistic first-order logic with equality, formulated within the Pure framework. Actually this is not an example of Isabelle/FOL, but of Isabelle/Pure.

1.1 Syntax

typedecl *i*
typedecl *o*

judgment
  Trueprop :: $o \Rightarrow prop$ (- *5*)

1.2 Propositional logic

axiomatization
  false :: o ($\bot$) **and**
  imp :: $o \Rightarrow o \Rightarrow o$ (**infixr** $\longrightarrow$ *25*) **and**
  conj :: $o \Rightarrow o \Rightarrow o$ (**infixr** $\wedge$ *35*) **and**
  disj :: $o \Rightarrow o \Rightarrow o$ (**infixr** $\vee$ *30*)
where
  falseE [*elim*]: $\bot \Longrightarrow A$ **and**

  impI [*intro*]: $(A \Longrightarrow B) \Longrightarrow A \longrightarrow B$ **and**
  mp [*dest*]: $A \longrightarrow B \Longrightarrow A \Longrightarrow B$ **and**

  conjI [*intro*]: $A \Longrightarrow B \Longrightarrow A \wedge B$ **and**
  conjD1: $A \wedge B \Longrightarrow A$ **and**
  conjD2: $A \wedge B \Longrightarrow B$ **and**

  disjE [*elim*]: $A \vee B \Longrightarrow (A \Longrightarrow C) \Longrightarrow (B \Longrightarrow C) \Longrightarrow C$ **and**
  disjI1 [*intro*]: $A \Longrightarrow A \vee B$ **and**
  disjI2 [*intro*]: $B \Longrightarrow A \vee B$

theorem *conjE* [*elim*]:
  assumes $A \wedge B$
  obtains A **and** B
⟨*proof*⟩

definition
  true :: o ($\top$) **where**
  $\top \equiv \bot \longrightarrow \bot$

definition
  not :: $o \Rightarrow o$ ($\neg$ - [*40*] *40*) **where**
  $\neg A \equiv A \longrightarrow \bot$

definition
  iff :: $o \Rightarrow o \Rightarrow o$ (**infixr** $\longleftrightarrow$ *25*) **where**
  $A \longleftrightarrow B \equiv (A \longrightarrow B) \wedge (B \longrightarrow A)$

theorem *trueI* [*intro*]: $\top$
⟨*proof*⟩

theorem *notI* [*intro*]: $(A \Longrightarrow \bot) \Longrightarrow \neg A$
⟨*proof*⟩

theorem *notE* [*elim*]: $\neg A \Longrightarrow A \Longrightarrow B$
⟨*proof*⟩

theorem *iffI* [*intro*]: $(A \Longrightarrow B) \Longrightarrow (B \Longrightarrow A) \Longrightarrow A \longleftrightarrow B$
⟨*proof*⟩

theorem *iff1* [*elim*]: $A \longleftrightarrow B \Longrightarrow A \Longrightarrow B$
⟨*proof*⟩

theorem *iff2* [*elim*]: $A \longleftrightarrow B \Longrightarrow B \Longrightarrow A$
⟨*proof*⟩

1.3 Equality

axiomatization
  equal :: $i \Rightarrow i \Rightarrow o$ (**infixl** = *50*)
where
  refl [*intro*]: $x = x$ **and**
  subst: $x = y \Longrightarrow P\ x \Longrightarrow P\ y$

theorem *trans* [*trans*]: $x = y \Longrightarrow y = z \Longrightarrow x = z$
  ⟨*proof*⟩

theorem *sym* [*sym*]: $x = y \Longrightarrow y = x$
⟨*proof*⟩

1.4 Quantifiers

axiomatization
  All :: $(i \Rightarrow o) \Rightarrow o$ (**binder** ∀ *10*) **and**
  Ex :: $(i \Rightarrow o) \Rightarrow o$ (**binder** ∃ *10*)
where
  allI [*intro*]: $(\bigwedge x.\ P\ x) \Longrightarrow \forall x.\ P\ x$ **and**
  allD [*dest*]: $\forall x.\ P\ x \Longrightarrow P\ a$ **and**
  exI [*intro*]: $P\ a \Longrightarrow \exists x.\ P\ x$ **and**
  exE [*elim*]: $\exists x.\ P\ x \Longrightarrow (\bigwedge x.\ P\ x \Longrightarrow C) \Longrightarrow C$

lemma $(\exists x.\ P\ (f\ x)) \longrightarrow (\exists y.\ P\ y)$
⟨*proof*⟩

lemma $(\exists x.\ \forall y.\ R\ x\ y) \longrightarrow (\forall y.\ \exists x.\ R\ x\ y)$
⟨*proof*⟩

end

2 Natural numbers

theory *Natural-Numbers* **imports** *FOL* **begin**

Theory of the natural numbers: Peano's axioms, primitive recursion. (Modernized version of Larry Paulson's theory "Nat".)

typedecl *nat*

arities *nat* :: *term*

axiomatization
Zero :: *nat* (0) **and**
Suc :: *nat* => *nat* **and**
rec :: [*nat*, *'a*, [*nat*, *'a*] => *'a*] => *'a*
where
induct [*case-names 0 Suc*, *induct type*: *nat*]:
$P(0)$ ==> (!!x. $P(x)$ ==> $P(Suc(x))$) ==> $P(n)$ **and**
Suc-inject: $Suc(m) = Suc(n)$ ==> $m = n$ **and**
Suc-neq-0: $Suc(m) = 0$ ==> R **and**
rec-0: $rec(0, a, f) = a$ **and**
rec-Suc: $rec(Suc(m), a, f) = f(m, rec(m, a, f))$

lemma *Suc-n-not-n*: $Suc(k) \neq k$
⟨*proof*⟩

definition
add :: [*nat*, *nat*] => *nat* (**infixl** + *60*) **where**
$m + n = rec(m, n, \lambda x\ y.\ Suc(y))$

lemma *add-0* [*simp*]: $0 + n = n$
⟨*proof*⟩

lemma *add-Suc* [*simp*]: $Suc(m) + n = Suc(m + n)$
⟨*proof*⟩

lemma *add-assoc*: $(k + m) + n = k + (m + n)$
⟨*proof*⟩

lemma *add-0-right*: $m + 0 = m$
⟨*proof*⟩

lemma *add-Suc-right*: $m + Suc(n) = Suc(m + n)$
⟨*proof*⟩

lemma
assumes !!n. $f(Suc(n)) = Suc(f(n))$
shows $f(i + j) = i + f(j)$
⟨*proof*⟩

end

3 Examples for the manual “Introduction to Isabelle”

theory *Intro*
imports *FOL*
begin

3.0.1 Some simple backward proofs

lemma *mythm*: $P|P$ −−> P
⟨*proof*⟩

lemma (P & Q) | R −−> (P | R)
⟨*proof*⟩

lemma (*ALL* x y. $P(x,y)$) −−> (*ALL* z w. $P(w,z)$)
⟨*proof*⟩

3.0.2 Demonstration of *fast*

lemma (*EX* y. *ALL* x. $J(y,x)$ <−> ~$J(x,x)$)
−−> ~ (*ALL* x. *EX* y. *ALL* z. $J(z,y)$ <−> ~ $J(z,x)$)
⟨*proof*⟩

lemma *ALL* x. $P(x,f(x))$ <−>
(*EX* y. (*ALL* z. $P(z,y)$ −−> $P(z,f(x))$) & $P(x,y)$)
⟨*proof*⟩

3.0.3 Derivation of conjunction elimination rule

lemma
assumes *major*: P&Q
and *minor*: [| P; Q |] ==> R
shows R
⟨*proof*⟩

3.1 Derived rules involving definitions

Derivation of negation introduction

lemma
assumes P ==> *False*
shows ~ P
⟨*proof*⟩

lemma
assumes *major*: ~P
and *minor*: P

shows *R*
⟨*proof*⟩

Alternative proof of the result above

```
lemma
  assumes major: ~P
    and minor: P
  shows R
⟨proof⟩

end
```

4 Theory of the natural numbers: Peano's axioms, primitive recursion

```
theory Nat
imports FOL
begin

typedecl nat
arities nat :: term

consts
  0 :: nat    (0)
  Suc :: nat => nat
  rec :: [nat, 'a, [nat, 'a] => 'a] => 'a
  add :: [nat, nat] => nat    (infixl + 60)

axioms
  induct:      [| P(0);  !!x. P(x) ==> P(Suc(x)) |]  ==> P(n)
  Suc-inject:  Suc(m)=Suc(n) ==> m=n
  Suc-neq-0:   Suc(m)=0      ==> R
  rec-0:       rec(0,a,f) = a
  rec-Suc:     rec(Suc(m), a, f) = f(m, rec(m,a,f))

defs
  add-def:     m+n == rec(m, n, %x y. Suc(y))
```

4.1 Proofs about the natural numbers

```
lemma Suc-n-not-n: Suc(k) ~= k
⟨proof⟩

lemma (k+m)+n = k+(m+n)
⟨proof⟩

lemma add-0 [simp]: 0+n = n
```

⟨*proof*⟩

lemma *add-Suc* [*simp*]: $Suc(m)+n = Suc(m+n)$
⟨*proof*⟩

lemma *add-assoc*: $(k+m)+n = k+(m+n)$
⟨*proof*⟩

lemma *add-0-right*: $m+0 = m$
⟨*proof*⟩

lemma *add-Suc-right*: $m+Suc(n) = Suc(m+n)$
⟨*proof*⟩

lemma
  assumes *prem*: !!n. $f(Suc(n)) = Suc(f(n))$
  shows $f(i+j) = i+f(j)$
⟨*proof*⟩

end

5 Intuitionistic FOL: Examples from The Foundation of a Generic Theorem Prover

theory *Foundation*
imports *IFOL*
begin

lemma $A\&B \ \ {-}{-}{>} \ (C{-}{-}{>}A\&C)$
⟨*proof*⟩

A form of conj-elimination

lemma
  assumes $A \ \& \ B$
    and A ==> B ==> C
  shows C
⟨*proof*⟩

lemma
  assumes !!A. ~ ~A ==> A
  shows B | ~B
⟨*proof*⟩

lemma
  assumes !!A. ~ ~A ==> A
  shows B | ~B
⟨*proof*⟩

lemma
  assumes *A* | ~*A*
    and ~ ~*A*
  shows *A*
⟨*proof*⟩

5.1 Examples with quantifiers

lemma
  assumes *ALL z. G(z)*
  shows *ALL z. G(z)|H(z)*
⟨*proof*⟩

lemma *ALL x. EX y. x=y*
⟨*proof*⟩

lemma *EX y. ALL x. x=y*
⟨*proof*⟩

Parallel lifting example.

lemma *EX u. ALL x. EX v. ALL y. EX w. P(u,x,v,y,w)*
⟨*proof*⟩

lemma
  assumes *(EX z. F(z)) & B*
  shows *EX z. F(z) & B*
⟨*proof*⟩

A bigger demonstration of quantifiers – not in the paper.

lemma *(EX y. ALL x. Q(x,y)) --> (ALL x. EX y. Q(x,y))*
⟨*proof*⟩

end

6 First-Order Logic: PROLOG examples

theory *Prolog*
imports *FOL*
begin

typedecl *'a list*
arities *list* :: (*term*) *term*
consts
  Nil   :: *'a list*
  Cons :: [*'a*, *'a list*]=> *'a list*   (**infixr** : *60*)

app :: ['*a list*, '*a list*, '*a list*] => *o*
rev :: ['*a list*, '*a list*] => *o*

axioms
appNil: *app*(*Nil*,*ys*,*ys*)
appCons: *app*(*xs*,*ys*,*zs*) ==> *app*(*x*:*xs*, *ys*, *x*:*zs*)
revNil: *rev*(*Nil*,*Nil*)
revCons: [| *rev*(*xs*,*ys*); *app*(*ys*, *x*:*Nil*, *zs*) |] ==> *rev*(*x*:*xs*, *zs*)

lemma *app*(*a*:*b*:*c*:*Nil*, *d*:*e*:*Nil*, *?x*)
⟨*proof*⟩

lemma *app*(*?x*, *c*:*d*:*Nil*, *a*:*b*:*c*:*d*:*Nil*)
⟨*proof*⟩

lemma *app*(*?x*, *?y*, *a*:*b*:*c*:*d*:*Nil*)
⟨*proof*⟩

lemmas *rules* = *appNil appCons revNil revCons*

lemma *rev*(*a*:*b*:*c*:*d*:*Nil*, *?x*)
⟨*proof*⟩

lemma *rev*(*a*:*b*:*c*:*d*:*e*:*f*:*g*:*h*:*i*:*j*:*k*:*l*:*m*:*n*:*Nil*, *?w*)
⟨*proof*⟩

lemma *rev*(*?x*, *a*:*b*:*c*:*Nil*)
⟨*proof*⟩

⟨*ML*⟩

lemma *rev*(*?x*, *a*:*b*:*c*:*Nil*)
⟨*proof*⟩

lemma *rev*(*a*:*?x*:*c*:*?y*:*Nil*, *d*:*?z*:*b*:*?u*)
⟨*proof*⟩

lemma *rev*(*a*:*b*:*c*:*d*:*e*:*f*:*g*:*h*:*i*:*j*:*k*:*l*:*m*:*n*:*o*:*p*:*Nil*, *?w*)
⟨*proof*⟩

lemma *a*:*b*:*c*:*d*:*e*:*f*:*g*:*h*:*i*:*j*:*k*:*l*:*m*:*n*:*o*:*p*:*Nil* = *?x* & *app*(*?x*,*?x*,*?y*) & *rev*(*?y*,*?w*)
⟨*proof*⟩

end

7 Intuitionistic First-Order Logic

theory *Intuitionistic* **imports** *IFOL* **begin**

Metatheorem (for *propositional* formulae): P is classically provable iff $\neg\neg P$ is intuitionistically provable. Therefore $\neg P$ is classically provable iff it is intuitionistically provable.

Proof: Let Q be the conjuction of the propositions $A \vee \neg A$, one for each atom A in P. Now $\neg\neg Q$ is intuitionistically provable because $\neg\neg(A \vee \neg A)$ is and because double-negation distributes over conjunction. If P is provable classically, then clearly $Q \rightarrow P$ is provable intuitionistically, so $\neg\neg(Q \rightarrow P)$ is also provable intuitionistically. The latter is intuitionistically equivalent to $\neg\neg Q \rightarrow \neg\neg P$, hence to $\neg\neg P$, since $\neg\neg Q$ is intuitionistically provable. Finally, if P is a negation then $\neg\neg P$ is intuitionstically equivalent to P. [Andy Pitts]

lemma `~~(P&Q) <-> ~~P & ~~Q`
⟨*proof*⟩

lemma `~~ ((~P --> Q) --> (~P --> ~Q) --> P)`
⟨*proof*⟩

Double-negation does NOT distribute over disjunction

lemma `~~(P-->Q)  <-> (~~P --> ~~Q)`
⟨*proof*⟩

lemma `~~~P <-> ~P`
⟨*proof*⟩

lemma `~~((P --> Q | R)  -->  (P-->Q) | (P-->R))`
⟨*proof*⟩

lemma `(P<->Q) <-> (Q<->P)`
⟨*proof*⟩

lemma `((P --> (Q | (Q-->R))) --> R) --> R`
⟨*proof*⟩

lemma `(((G-->A) --> J) --> D --> E) --> (((H-->B)-->I)-->C-->J) --> (A-->H) --> F --> G --> (((C-->B)-->I)-->D)-->(A-->C) --> (((F-->A)-->B) --> I) --> E`
⟨*proof*⟩

Lemmas for the propositional double-negation translation

lemma `P --> ~~P`
⟨*proof*⟩

lemma ~~(~~*P* --> *P*)
⟨*proof*⟩

lemma ~~*P* & ~~(*P* --> *Q*) --> ~~*Q*
⟨*proof*⟩

The following are classically but not constructively valid. The attempt to prove them terminates quickly!

lemma ((*P*-->*Q*) --> *P*) --> *P*
⟨*proof*⟩

lemma (*P*&*Q*-->*R*) --> (*P*-->*R*) | (*Q*-->*R*)
⟨*proof*⟩

7.1 de Bruijn formulae

de Bruijn formula with three predicates

lemma ((*P*<->*Q*) --> *P*&*Q*&*R*) &
((*Q*<->*R*) --> *P*&*Q*&*R*) &
((*R*<->*P*) --> *P*&*Q*&*R*) --> *P*&*Q*&*R*
⟨*proof*⟩

de Bruijn formula with five predicates

lemma ((*P*<->*Q*) --> *P*&*Q*&*R*&*S*&*T*) &
((*Q*<->*R*) --> *P*&*Q*&*R*&*S*&*T*) &
((*R*<->*S*) --> *P*&*Q*&*R*&*S*&*T*) &
((*S*<->*T*) --> *P*&*Q*&*R*&*S*&*T*) &
((*T*<->*P*) --> *P*&*Q*&*R*&*S*&*T*) --> *P*&*Q*&*R*&*S*&*T*
⟨*proof*⟩

Problem 1.1

lemma (*ALL x. EX y. ALL z. p*(*x*) & *q*(*y*) & *r*(*z*)) <->
(*ALL z. EX y. ALL x. p*(*x*) & *q*(*y*) & *r*(*z*))
⟨*proof*⟩

Problem 3.1

lemma ~ (*EX x. ALL y. mem*(*y*,*x*) <-> ~ *mem*(*x*,*x*))
⟨*proof*⟩

Problem 4.1: hopeless!

lemma (*ALL x. p*(*x*) --> *p*(*h*(*x*)) | *p*(*g*(*x*))) & (*EX x. p*(*x*)) & (*ALL x.* ~*p*(*h*(*x*)))
--> (*EX x. p*(*g*(*g*(*g*(*g*(*g*(*x*)))))))
⟨*proof*⟩

7.2 Intuitionistic FOL: propositional problems based on Pelletier.

1

lemma ~~((P-->Q) <-> (~Q --> ~P))
⟨*proof*⟩

2

lemma ~~(~~P <-> P)
⟨*proof*⟩

3

lemma ~(P-->Q) --> (Q-->P)
⟨*proof*⟩

4

lemma ~~((~P-->Q) <-> (~Q --> P))
⟨*proof*⟩

5

lemma ~~((P|Q-->P|R) --> P|(Q-->R))
⟨*proof*⟩

6

lemma ~~(P | ~P)
⟨*proof*⟩

7

lemma ~~(P | ~~~P)
⟨*proof*⟩

8. Peirce's law

lemma ~~(((P-->Q) --> P) --> P)
⟨*proof*⟩

9

lemma ((P|Q) & (~P|Q) & (P| ~Q)) --> ~ (~P | ~Q)
⟨*proof*⟩

10

lemma (Q-->R) --> (R-->P&Q) --> (P-->(Q|R)) --> (P<->Q)
⟨*proof*⟩

7.3 11. Proved in each direction (incorrectly, says Pelletier!!)

lemma $P{<}{-}{>}P$
⟨*proof*⟩

12. Dijkstra's law

lemma $\sim\sim(((P <-> Q) <-> R)\ \ <->\ \ (P <-> (Q <-> R)))$
⟨*proof*⟩

lemma $((P <-> Q) <-> R)\ \ -->\ \ \sim\sim(P <-> (Q <-> R))$
⟨*proof*⟩

13. Distributive law

lemma $P \mid (Q \ \& \ R)\ \ <-> (P \mid Q) \ \& \ (P \mid R)$
⟨*proof*⟩

14

lemma $\sim\sim((P <-> Q) <-> ((Q \mid \sim P) \ \& \ (\sim Q|P)))$
⟨*proof*⟩

15

lemma $\sim\sim((P --> Q) <-> (\sim P \mid Q))$
⟨*proof*⟩

16

lemma $\sim\sim((P{-}{-}{>}Q) \mid (Q{-}{-}{>}P))$
⟨*proof*⟩

17

lemma $\sim\sim(((P \ \& \ (Q{-}{-}{>}R)){-}{-}{>}S) <-> ((\sim P \mid Q \mid S) \ \& \ (\sim P \mid \sim R \mid S)))$
⟨*proof*⟩

Dijkstra's "Golden Rule"

lemma $(P\&Q) <-> P <-> Q <-> (P|Q)$
⟨*proof*⟩

7.4 ****Examples with quantifiers****

7.5 The converse is classical in the following implications...

lemma $(EX\ x.\ P(x){-}{-}{>}Q)\ \ -->\ \ (ALL\ x.\ P(x)) --> Q$
⟨*proof*⟩

lemma $((ALL\ x.\ P(x)){-}{-}{>}Q) --> \sim\ (ALL\ x.\ P(x) \ \& \sim Q)$
⟨*proof*⟩

lemma $((ALL\ x.\ \sim P(x)){-}{-}{>}Q)\ \ -->\ \ \sim\ (ALL\ x.\ \sim\ (P(x)|Q))$
⟨*proof*⟩

lemma $(ALL\ x.\ P(x))\ |\ Q\ \ -->\ \ (ALL\ x.\ P(x)\ |\ Q)$
⟨*proof*⟩

lemma $(EX\ x.\ P\ -->\ Q(x))\ -->\ (P\ -->\ (EX\ x.\ Q(x)))$
⟨*proof*⟩

7.6 The following are not constructively valid!

The attempt to prove them terminates quickly!

lemma $((ALL\ x.\ P(x))-->Q)\ -->\ (EX\ x.\ P(x)-->Q)$
⟨*proof*⟩

lemma $(P\ -->\ (EX\ x.\ Q(x)))\ -->\ (EX\ x.\ P-->Q(x))$
⟨*proof*⟩

lemma $(ALL\ x.\ P(x)\ |\ Q)\ -->\ ((ALL\ x.\ P(x))\ |\ Q)$
⟨*proof*⟩

lemma $(ALL\ x.\ \sim\sim P(x))\ -->\ \sim\sim(ALL\ x.\ P(x))$
⟨*proof*⟩

Classically but not intuitionistically valid. Proved by a bug in 1986!

lemma $EX\ x.\ Q(x)\ -->\ (ALL\ x.\ Q(x))$
⟨*proof*⟩

7.7 Hard examples with quantifiers

The ones that have not been proved are not known to be valid! Some will require quantifier duplication – not currently available

18

lemma $\sim\sim(EX\ y.\ ALL\ x.\ P(y)-->P(x))$
⟨*proof*⟩

19

lemma $\sim\sim(EX\ x.\ ALL\ y\ z.\ (P(y)-->Q(z))\ -->\ (P(x)-->Q(x)))$
⟨*proof*⟩

20

lemma $(ALL\ x\ y.\ EX\ z.\ ALL\ w.\ (P(x)\&Q(y)-->R(z)\&S(w)))$
$\quad -->\ (EX\ x\ y.\ P(x)\ \&\ Q(y))\ -->\ (EX\ z.\ R(z))$
⟨*proof*⟩

21

lemma $(EX\ x.\ P-->Q(x))\ \&\ (EX\ x.\ Q(x)-->P)\ -->\ \sim\sim(EX\ x.\ P<->Q(x))$
⟨*proof*⟩

22

lemma (ALL x. P <-> Q(x)) --> (P <-> (ALL x. Q(x)))
⟨*proof*⟩

23

lemma ~~ ((ALL x. P | Q(x)) <-> (P | (ALL x. Q(x))))
⟨*proof*⟩

24

lemma ~(EX x. S(x)&Q(x)) & (ALL x. P(x) --> Q(x)|R(x)) &
(~(EX x. P(x)) --> (EX x. Q(x))) & (ALL x. Q(x)|R(x) --> S(x))
--> ~~(EX x. P(x)&R(x))⟨*proof*⟩

25

lemma (EX x. P(x)) &
(ALL x. L(x) --> ~ (M(x) & R(x))) &
(ALL x. P(x) --> (M(x) & L(x))) &
((ALL x. P(x)-->Q(x)) | (EX x. P(x)&R(x)))
--> (EX x. Q(x)&P(x))
⟨*proof*⟩

26

lemma (~~(EX x. p(x)) <-> ~~(EX x. q(x))) &
(ALL x. ALL y. p(x) & q(y) --> (r(x) <-> s(y)))
--> ((ALL x. p(x)-->r(x)) <-> (ALL x. q(x)-->s(x)))
⟨*proof*⟩

27

lemma (EX x. P(x) & ~Q(x)) &
(ALL x. P(x) --> R(x)) &
(ALL x. M(x) & L(x) --> P(x)) &
((EX x. R(x) & ~ Q(x)) --> (ALL x. L(x) --> ~ R(x)))
--> (ALL x. M(x) --> ~L(x))
⟨*proof*⟩

28. AMENDED

lemma (ALL x. P(x) --> (ALL x. Q(x))) &
(~~(ALL x. Q(x)|R(x)) --> (EX x. Q(x)&S(x))) &
(~~(EX x. S(x)) --> (ALL x. L(x) --> M(x)))
--> (ALL x. P(x) & L(x) --> M(x))
⟨*proof*⟩

29. Essentially the same as Principia Mathematica *11.71

lemma (EX x. P(x)) & (EX y. Q(y))
--> ((ALL x. P(x)-->R(x)) & (ALL y. Q(y)-->S(y)) <->
(ALL x y. P(x) & Q(y) --> R(x) & S(y)))
⟨*proof*⟩

30

lemma (ALL x. (P(x) | Q(x)) --> ~ R(x)) &
    (ALL x. (Q(x) --> ~ S(x)) --> P(x) & R(x))
  --> (ALL x. ~~S(x))
⟨proof⟩

31

lemma ~(EX x. P(x) & (Q(x) | R(x))) &
    (EX x. L(x) & P(x)) &
    (ALL x. ~ R(x) --> M(x))
  --> (EX x. L(x) & M(x))
⟨proof⟩

32

lemma (ALL x. P(x) & (Q(x)|R(x))-->S(x)) &
    (ALL x. S(x) & R(x) --> L(x)) &
    (ALL x. M(x) --> R(x))
  --> (ALL x. P(x) & M(x) --> L(x))
⟨proof⟩

33

lemma (ALL x. ~~(P(a) & (P(x)-->P(b))-->P(c))) <->
   (ALL x. ~~((~P(a) | P(x) | P(c)) & (~P(a) | ~P(b) | P(c))))
⟨proof⟩

36

lemma (ALL x. EX y. J(x,y)) &
   (ALL x. EX y. G(x,y)) &
   (ALL x y. J(x,y) | G(x,y) --> (ALL z. J(y,z) | G(y,z) --> H(x,z)))
 --> (ALL x. EX y. H(x,y))
⟨proof⟩

37

lemma (ALL z. EX w. ALL x. EX y.
      ~~(P(x,z)-->P(y,w)) & P(y,z) & (P(y,w) --> (EX u. Q(u,w)))) &
    (ALL x z. ~P(x,z) --> (EX y. Q(y,z))) &
    (~~(EX x y. Q(x,y)) --> (ALL x. R(x,x)))
  --> ~~(ALL x. EX y. R(x,y))
⟨proof⟩

39

lemma ~ (EX x. ALL y. F(y,x) <-> ~F(y,y))
⟨proof⟩

40. AMENDED

lemma (EX y. ALL x. F(x,y) <-> F(x,x)) -->
      ~(ALL x. EX y. ALL z. F(z,y) <-> ~ F(z,x))

⟨*proof*⟩

44

lemma (ALL x. f(x) -->
(EX y. g(y) & h(x,y) & (EX y. g(y) & ~ h(x,y)))) &
(EX x. j(x) & (ALL y. g(y) --> h(x,y)))
--> (EX x. j(x) & ~f(x))
⟨*proof*⟩

48

lemma (a=b | c=d) & (a=c | b=d) --> a=d | b=c
⟨*proof*⟩

51

lemma (EX z w. ALL x y. P(x,y) <-> (x=z & y=w)) -->
(EX z. ALL x. EX w. (ALL y. P(x,y) <-> y=w) <-> x=z)
⟨*proof*⟩

52

Almost the same as 51.

lemma (EX z w. ALL x y. P(x,y) <-> (x=z & y=w)) -->
(EX w. ALL y. EX z. (ALL x. P(x,y) <-> x=z) <-> y=w)
⟨*proof*⟩

56

lemma (ALL x. (EX y. P(y) & x=f(y)) --> P(x)) <-> (ALL x. P(x) --> P(f(x)))
⟨*proof*⟩

57

lemma P(f(a,b), f(b,c)) & P(f(b,c), f(a,c)) &
(ALL x y z. P(x,y) & P(y,z) --> P(x,z)) --> P(f(a,b), f(a,c))
⟨*proof*⟩

60

lemma ALL x. P(x,f(x)) <-> (EX y. (ALL z. P(z,y) --> P(z,f(x))) & P(x,y))
⟨*proof*⟩

end

8 First-Order Logic: propositional examples (intuitionistic version)

theory *Propositional-Int*

imports *IFOL*
begin

commutative laws of & and |

lemma *P & Q --> Q & P*
⟨*proof*⟩

lemma *P | Q --> Q | P*
⟨*proof*⟩

associative laws of & and |

lemma *(P & Q) & R --> P & (Q & R)*
⟨*proof*⟩

lemma *(P | Q) | R --> P | (Q | R)*
⟨*proof*⟩

distributive laws of & and |

lemma *(P & Q) | R --> (P | R) & (Q | R)*
⟨*proof*⟩

lemma *(P | R) & (Q | R) --> (P & Q) | R*
⟨*proof*⟩

lemma *(P | Q) & R --> (P & R) | (Q & R)*
⟨*proof*⟩

lemma *(P & R) | (Q & R) --> (P | Q) & R*
⟨*proof*⟩

Laws involving implication

lemma *(P-->R) & (Q-->R) <-> (P|Q --> R)*
⟨*proof*⟩

lemma *(P & Q --> R) <-> (P--> (Q-->R))*
⟨*proof*⟩

lemma *((P-->R)-->R) --> ((Q-->R)-->R) --> (P&Q-->R) --> R*
⟨*proof*⟩

lemma *~(P-->R) --> ~(Q-->R) --> ~(P&Q-->R)*
⟨*proof*⟩

lemma *(P --> Q & R) <-> (P-->Q) & (P-->R)*
⟨*proof*⟩

Propositions-as-types

— The combinator K

lemma $P \dashrightarrow (Q \dashrightarrow P)$
⟨*proof*⟩

lemma $(P\text{-->}Q\text{-->}R)$ --> $(P\text{-->}Q)$ --> $(P\text{-->}R)$
⟨*proof*⟩

lemma $(P\text{-->}Q) \mid (P\text{-->}R)$ --> $(P$ --> $Q \mid R)$
⟨*proof*⟩

lemma $(P\text{-->}Q)$ --> $(\sim Q$ --> $\sim P)$
⟨*proof*⟩

Schwichtenberg's examples (via T. Nipkow)

lemma *stab-imp*: (((Q-->R)-->R)-->Q) --> (((P-->Q)-->R)-->R)-->P-->Q
⟨*proof*⟩

lemma *stab-to-peirce*:
(((P --> R) --> R) --> P) --> (((Q --> R) --> R) --> Q)
--> ((P --> Q) --> P) --> P
⟨*proof*⟩

lemma *peirce-imp1*: (((Q --> R) --> Q) --> Q)
--> (((P --> Q) --> R) --> P --> Q) --> P --> Q
⟨*proof*⟩

lemma *peirce-imp2*: (((P --> R) --> P) --> P) --> ((P --> Q --> R) --> P) --> P
⟨*proof*⟩

lemma *mints*: ((((P --> Q) --> P) --> P) --> Q) --> Q
⟨*proof*⟩

lemma *mints-solovev*: (P --> (Q --> R) --> Q) --> ((P --> Q) --> R) --> R
⟨*proof*⟩

lemma *tatsuta*: (((P7 --> P1) --> P10) --> P4 --> P5)
--> (((P8 --> P2) --> P9) --> P3 --> P10)
--> (P1 --> P8) --> P6 --> P7
--> (((P3 --> P2) --> P9) --> P4)
--> (P1 --> P3) --> (((P6 --> P1) --> P2) --> P9) --> P5
⟨*proof*⟩

lemma *tatsuta1*: (((P8 --> P2) --> P9) --> P3 --> P10)
--> (((P3 --> P2) --> P9) --> P4)
--> (((P6 --> P1) --> P2) --> P9)
--> (((P7 --> P1) --> P10) --> P4 --> P5)
--> (P1 --> P3) --> (P1 --> P8) --> P6 --> P7 --> P5
⟨*proof*⟩

end

9 First-Order Logic: quantifier examples (intuitionistic version)

theory *Quantifiers-Int*
imports *IFOL*
begin

lemma *(ALL x y. P(x,y)) --> (ALL y x. P(x,y))*
⟨*proof*⟩

lemma *(EX x y. P(x,y)) --> (EX y x. P(x,y))*
⟨*proof*⟩
lemma *(ALL x. P(x)) | (ALL x. Q(x)) --> (ALL x. P(x) | Q(x))*
⟨*proof*⟩

lemma *(ALL x. P-->Q(x)) <-> (P--> (ALL x. Q(x)))*
⟨*proof*⟩

lemma *(ALL x. P(x)-->Q) <-> ((EX x. P(x)) --> Q)*
⟨*proof*⟩

Some harder ones

lemma *(EX x. P(x) | Q(x)) <-> (EX x. P(x)) | (EX x. Q(x))*
⟨*proof*⟩
lemma *(EX x. P(x)&Q(x)) --> (EX x. P(x)) & (EX x. Q(x))*
⟨*proof*⟩

Basic test of quantifier reasoning

— TRUE
lemma *(EX y. ALL x. Q(x,y)) --> (ALL x. EX y. Q(x,y))*
⟨*proof*⟩

lemma *(ALL x. Q(x)) --> (EX x. Q(x))*
⟨*proof*⟩

The following should fail, as they are false!

lemma *(ALL x. EX y. Q(x,y)) --> (EX y. ALL x. Q(x,y))*
⟨*proof*⟩

lemma *(EX x. Q(x)) --> (ALL x. Q(x))*
⟨*proof*⟩

lemma *P(?a) --> (ALL x. P(x))*
⟨*proof*⟩

lemma (P(?a) --> (ALL x. Q(x))) --> (ALL x. P(x) --> Q(x))
⟨proof⟩

Back to things that are provable ...

lemma (ALL x. P(x)-->Q(x)) & (EX x. P(x)) --> (EX x. Q(x))
⟨proof⟩
lemma (P --> (EX x. Q(x))) & P --> (EX x. Q(x))
⟨proof⟩

lemma (ALL x. P(x)-->Q(f(x))) & (ALL x. Q(x)-->R(g(x))) & P(d) --> R(?a)
⟨proof⟩

lemma (ALL x. Q(x)) --> (EX x. Q(x))
⟨proof⟩

Some slow ones

— Principia Mathematica *11.53
lemma (ALL x y. P(x) --> Q(y)) <-> ((EX x. P(x)) --> (ALL y. Q(y)))
⟨proof⟩

lemma (EX x y. P(x) & Q(x,y)) <-> (EX x. P(x) & (EX y. Q(x,y)))
⟨proof⟩

lemma (EX y. ALL x. P(x) --> Q(x,y)) --> (ALL x. P(x) --> (EX y. Q(x,y)))
⟨proof⟩

end

10 Classical Predicate Calculus Problems

theory *Classical* **imports** *FOL* **begin**

lemma (P --> Q | R) --> (P-->Q) | (P-->R)
⟨proof⟩

If and only if

lemma (P<->Q) <-> (Q<->P)
⟨proof⟩

lemma ~ (P <-> ~P)
⟨proof⟩

Sample problems from F. J. Pelletier, Seventy-Five Problems for Testing Automatic Theorem Provers, J. Automated Reasoning 2 (1986), 191-216. Errata, JAR 4 (1988), 236-236.

The hardest problems – judging by experience with several theorem provers, including matrix ones – are 34 and 43.

10.1 Pelletier's examples

1

lemma $(P-->Q)\ \ <->\ \ (\sim Q\ -->\ \sim P)$
⟨*proof*⟩

2

lemma $\sim\ \sim\ P\ \ <->\ \ P$
⟨*proof*⟩

3

lemma $\sim(P-->Q)\ -->\ (Q-->P)$
⟨*proof*⟩

4

lemma $(\sim P-->Q)\ \ <->\ \ (\sim Q\ -->\ P)$
⟨*proof*⟩

5

lemma $((P|Q)-->(P|R))\ -->\ (P|(Q-->R))$
⟨*proof*⟩

6

lemma $P\ |\ \sim\ P$
⟨*proof*⟩

7

lemma $P\ |\ \sim\ \sim\ \sim\ P$
⟨*proof*⟩

8. Peirce's law

lemma $((P-->Q)\ -->\ P)\ \ -->\ \ P$
⟨*proof*⟩

9

lemma $((P|Q)\ \&\ (\sim P|Q)\ \&\ (P|\ \sim Q))\ -->\ \sim\ (\sim P\ |\ \sim Q)$
⟨*proof*⟩

10

lemma $(Q{-}{-}{>}R)$ & $(R{-}{-}{>}P\&Q)$ & $(P{-}{-}{>}Q|R)$ $-{-}{>}$ $(P{<}{-}{>}Q)$
⟨*proof*⟩

11. Proved in each direction (incorrectly, says Pelletier!!)

lemma $P{<}{-}{>}P$
⟨*proof*⟩

12. "Dijkstra's law"

lemma $((P <-> Q) <-> R)\ \ <->\ \ (P <-> (Q <-> R))$
⟨*proof*⟩

13. Distributive law

lemma $P \mid (Q\ \&\ R)\ \ <-> (P \mid Q)\ \&\ (P \mid R)$
⟨*proof*⟩

14

lemma $(P <-> Q) <-> ((Q \mid {\sim}P)\ \&\ ({\sim}Q|P))$
⟨*proof*⟩

15

lemma $(P\ {-}{-}{>}\ Q) <-> ({\sim}P \mid Q)$
⟨*proof*⟩

16

lemma $(P{-}{-}{>}Q) \mid (Q{-}{-}{>}P)$
⟨*proof*⟩

17

lemma $((P\ \&\ (Q{-}{-}{>}R)){-}{-}{>}S) <-> (({\sim}P \mid Q \mid S)\ \&\ ({\sim}P \mid {\sim}R \mid S))$
⟨*proof*⟩

10.2 Classical Logic: examples with quantifiers

lemma $(\forall x.\ P(x)\ \&\ Q(x)) <-> (\forall x.\ P(x))\ \ \&\ \ (\forall x.\ Q(x))$
⟨*proof*⟩

lemma $(\exists x.\ P{-}{-}{>}Q(x))\ \ <->\ \ (P\ {-}{-}{>}\ (\exists x.\ Q(x)))$
⟨*proof*⟩

lemma $(\exists x.\ P(x){-}{-}{>}Q)\ \ <->\ \ (\forall x.\ P(x))\ {-}{-}{>}\ Q$
⟨*proof*⟩

lemma $(\forall x.\ P(x)) \mid Q\ \ <->\ \ (\forall x.\ P(x) \mid Q)$
⟨*proof*⟩

Discussed in Avron, Gentzen-Type Systems, Resolution and Tableaux, JAR 10 (265-281), 1993. Proof is trivial!

lemma ${\sim}((\exists x.{\sim}P(x))\ \&\ ((\exists x.\ P(x)) \mid (\exists x.\ P(x)\ \&\ Q(x)))\ \&\ {\sim}\ (\exists x.\ P(x)))$
⟨*proof*⟩

10.3 Problems requiring quantifier duplication

Theorem B of Peter Andrews, Theorem Proving via General Matings, JACM 28 (1981).

lemma $(\exists x.\ \forall y.\ P(x) <-> P(y)) --> ((\exists x.\ P(x)) <-> (\forall y.\ P(y)))$
$\langle \mathit{proof} \rangle$

Needs multiple instantiation of ALL.

lemma $(\forall x.\ P(x)-->P(f(x)))\ \ \&\ \ P(d)-->P(f(f(f(d))))$
$\langle \mathit{proof} \rangle$

Needs double instantiation of the quantifier

lemma $\exists x.\ P(x)\ -->\ P(a)\ \&\ P(b)$
$\langle \mathit{proof} \rangle$

lemma $\exists z.\ P(z)\ -->\ (\forall x.\ P(x))$
$\langle \mathit{proof} \rangle$

lemma $\exists x.\ (\exists y.\ P(y))\ -->\ P(x)$
$\langle \mathit{proof} \rangle$

V. Lifschitz, What Is the Inverse Method?, JAR 5 (1989), 1–23. NOT PROVED

lemma $\exists x\ x'.\ \forall y.\ \exists z\ z'.$
$\quad (\sim P(y,y)\ |\ P(x,x)\ |\ \sim S(z,x))\ \&$
$\quad (S(x,y)\ |\ \sim S(y,z)\ |\ Q(z',z'))\ \&$
$\quad (Q(x',y)\ |\ \sim Q(y,z')\ |\ S(x',x'))$
$\langle \mathit{proof} \rangle$

10.4 Hard examples with quantifiers

18

lemma $\exists y.\ \forall x.\ P(y)-->P(x)$
$\langle \mathit{proof} \rangle$

19

lemma $\exists x.\ \forall y\ z.\ (P(y)-->Q(z))\ -->\ (P(x)-->Q(x))$
$\langle \mathit{proof} \rangle$

20

lemma $(\forall x\ y.\ \exists z.\ \forall w.\ (P(x)\&Q(y)-->R(z)\&S(w)))$
$\quad -->\ (\exists x\ y.\ P(x)\ \&\ Q(y))\ -->\ (\exists z.\ R(z))$
$\langle \mathit{proof} \rangle$

21

lemma $(\exists x.\ P-->Q(x))\ \&\ (\exists x.\ Q(x)-->P)\ -->\ (\exists x.\ P<->Q(x))$

$\langle proof \rangle$

22

lemma $(\forall x.\ P <-> Q(x))\ \ -->\ \ (P <-> (\forall x.\ Q(x)))$
$\langle proof \rangle$

23

lemma $(\forall x.\ P \mid Q(x))\ \ <->\ \ (P \mid (\forall x.\ Q(x)))$
$\langle proof \rangle$

24

lemma $\tilde{}(\exists x.\ S(x) \& Q(x))\ \&\ (\forall x.\ P(x)\ -->\ Q(x)|R(x))\ \&$
$\quad (\tilde{}(\exists x.\ P(x))\ -->\ (\exists x.\ Q(x)))\ \&\ (\forall x.\ Q(x)|R(x)\ -->\ S(x))$
$\quad -->\ (\exists x.\ P(x) \& R(x))$
$\langle proof \rangle$

25

lemma $(\exists x.\ P(x))\ \&$
$\quad (\forall x.\ L(x)\ -->\ \tilde{}\ (M(x)\ \&\ R(x)))\ \&$
$\quad (\forall x.\ P(x)\ -->\ (M(x)\ \&\ L(x)))\ \&$
$\quad ((\forall x.\ P(x) --> Q(x))\ |\ (\exists x.\ P(x) \& R(x)))$
$\quad -->\ (\exists x.\ Q(x) \& P(x))$
$\langle proof \rangle$

26

lemma $((\exists x.\ p(x))\ <->\ (\exists x.\ q(x)))\ \&$
$\quad (\forall x.\ \forall y.\ p(x)\ \&\ q(y)\ -->\ (r(x)\ <->\ s(y)))$
$\quad -->\ ((\forall x.\ p(x) --> r(x))\ <->\ (\forall x.\ q(x) --> s(x)))$
$\langle proof \rangle$

27

lemma $(\exists x.\ P(x)\ \&\ \tilde{}Q(x))\ \&$
$\quad (\forall x.\ P(x)\ -->\ R(x))\ \&$
$\quad (\forall x.\ M(x)\ \&\ L(x)\ -->\ P(x))\ \&$
$\quad ((\exists x.\ R(x)\ \&\ \tilde{}\ Q(x))\ -->\ (\forall x.\ L(x)\ -->\ \tilde{}\ R(x)))$
$\quad -->\ (\forall x.\ M(x)\ -->\ \tilde{}L(x))$
$\langle proof \rangle$

28. AMENDED

lemma $(\forall x.\ P(x)\ -->\ (\forall x.\ Q(x)))\ \&$
$\quad ((\forall x.\ Q(x)|R(x))\ -->\ (\exists x.\ Q(x) \& S(x)))\ \&$
$\quad ((\exists x.\ S(x))\ -->\ (\forall x.\ L(x)\ -->\ M(x)))$
$\quad -->\ (\forall x.\ P(x)\ \&\ L(x)\ -->\ M(x))$
$\langle proof \rangle$

29. Essentially the same as Principia Mathematica *11.71

lemma $(\exists x.\ P(x))\ \&\ (\exists y.\ Q(y))$

--> ((∀x. P(x)-->R(x)) & (∀y. Q(y)-->S(y)) <->
(∀x y. P(x) & Q(y) --> R(x) & S(y)))
⟨*proof*⟩

30

lemma (∀x. P(x) | Q(x) --> ~ R(x)) &
(∀x. (Q(x) --> ~ S(x)) --> P(x) & R(x))
--> (∀x. S(x))
⟨*proof*⟩

31

lemma ~(∃x. P(x) & (Q(x) | R(x))) &
(∃x. L(x) & P(x)) &
(∀x. ~ R(x) --> M(x))
--> (∃x. L(x) & M(x))
⟨*proof*⟩

32

lemma (∀x. P(x) & (Q(x)|R(x))-->S(x)) &
(∀x. S(x) & R(x) --> L(x)) &
(∀x. M(x) --> R(x))
--> (∀x. P(x) & M(x) --> L(x))
⟨*proof*⟩

33

lemma (∀x. P(a) & (P(x)-->P(b))-->P(c)) <->
(∀x. (~P(a) | P(x) | P(c)) & (~P(a) | ~P(b) | P(c)))
⟨*proof*⟩

34 AMENDED (TWICE!!). Andrews's challenge

lemma ((∃x. ∀y. p(x) <-> p(y)) <->
((∃x. q(x)) <-> (∀y. p(y)))) <->
((∃x. ∀y. q(x) <-> q(y)) <->
((∃x. p(x)) <-> (∀y. q(y))))
⟨*proof*⟩

35

lemma ∃x y. P(x,y) --> (∀u v. P(u,v))
⟨*proof*⟩

36

lemma (∀x. ∃y. J(x,y)) &
(∀x. ∃y. G(x,y)) &
(∀x y. J(x,y) | G(x,y) --> (∀z. J(y,z) | G(y,z) --> H(x,z)))
--> (∀x. ∃y. H(x,y))
⟨*proof*⟩

37

lemma $(\forall z.\ \exists w.\ \forall x.\ \exists y.$
$\quad (P(x,z) --> P(y,w))\ \&\ P(y,z)\ \&\ (P(y,w)\ -->\ (\exists u.\ Q(u,w))))\ \&$
$\quad (\forall x\ z.\ {\sim}P(x,z)\ -->\ (\exists y.\ Q(y,z)))\ \&$
$\quad ((\exists x\ y.\ Q(x,y))\ -->\ (\forall x.\ R(x,x)))$
$\quad -->\ (\forall x.\ \exists y.\ R(x,y))$
⟨*proof*⟩

38

lemma $(\forall x.\ p(a)\ \&\ (p(x)\ -->\ (\exists y.\ p(y)\ \&\ r(x,y)))\ -->$
$\quad (\exists z.\ \exists w.\ p(z)\ \&\ r(x,w)\ \&\ r(w,z)))\ \ <->$
$\quad (\forall x.\ ({\sim}p(a)\ |\ p(x)\ |\ (\exists z.\ \exists w.\ p(z)\ \&\ r(x,w)\ \&\ r(w,z)))\ \&$
$\quad ({\sim}p(a)\ |\ {\sim}(\exists y.\ p(y)\ \&\ r(x,y))\ |$
$\quad (\exists z.\ \exists w.\ p(z)\ \&\ r(x,w)\ \&\ r(w,z))))$
⟨*proof*⟩

39

lemma ${\sim}\ (\exists x.\ \forall y.\ F(y,x)\ <->\ {\sim}F(y,y))$
⟨*proof*⟩

40. AMENDED

lemma $(\exists y.\ \forall x.\ F(x,y)\ <->\ F(x,x))\ -->$
$\quad {\sim}(\forall x.\ \exists y.\ \forall z.\ F(z,y)\ <->\ {\sim}\ F(z,x))$
⟨*proof*⟩

41

lemma $(\forall z.\ \exists y.\ \forall x.\ f(x,y)\ <->\ f(x,z)\ \&\ {\sim}\ f(x,x))$
$\quad -->\ {\sim}\ (\exists z.\ \forall x.\ f(x,z))$
⟨*proof*⟩

42

lemma ${\sim}\ (\exists y.\ \forall x.\ p(x,y)\ <->\ {\sim}\ (\exists z.\ p(x,z)\ \&\ p(z,x)))$
⟨*proof*⟩

43

lemma $(\forall x.\ \forall y.\ q(x,y)\ <->\ (\forall z.\ p(z,x)\ <->\ p(z,y)))$
$\quad -->\ (\forall x.\ \forall y.\ q(x,y)\ <->\ q(y,x))$
⟨*proof*⟩

44

lemma $(\forall x.\ f(x)\ -->\ (\exists y.\ g(y)\ \&\ h(x,y)\ \&\ (\exists y.\ g(y)\ \&\ {\sim}\ h(x,y))))\ \&$
$\quad (\exists x.\ j(x)\ \&\ (\forall y.\ g(y)\ -->\ h(x,y)))$
$\quad -->\ (\exists x.\ j(x)\ \&\ {\sim}f(x))$
⟨*proof*⟩

45

lemma $(\forall x.\ f(x)\ \&\ (\forall y.\ g(y)\ \&\ h(x,y)\ -->\ j(x,y))$
$\quad -->\ (\forall y.\ g(y)\ \&\ h(x,y)\ -->\ k(y)))\ \&$

$\sim (\exists y.\ l(y)\ \&\ k(y))\ \&$
$(\exists x.\ f(x)\ \&\ (\forall y.\ h(x,y) \ {-}{-}{>}\ l(y))$
$\quad \&\ (\forall y.\ g(y)\ \&\ h(x,y) \ {-}{-}{>}\ j(x,y)))$
${-}{-}{>}\ (\exists x.\ f(x)\ \&\ \sim (\exists y.\ g(y)\ \&\ h(x,y)))$
⟨*proof*⟩

46

lemma $(\forall x.\ f(x)\ \&\ (\forall y.\ f(y)\ \&\ h(y,x) \ {-}{-}{>}\ g(y)) \ {-}{-}{>}\ g(x))\ \&$
$((\exists x.\ f(x)\ \&\ {\sim}g(x)) \ {-}{-}{>}$
$(\exists x.\ f(x)\ \&\ {\sim}g(x)\ \&\ (\forall y.\ f(y)\ \&\ {\sim}g(y) \ {-}{-}{>}\ j(x,y))))\ \&$
$(\forall x\ y.\ f(x)\ \&\ f(y)\ \&\ h(x,y) \ {-}{-}{>}\ {\sim}j(y,x))$
${-}{-}{>}\ (\forall x.\ f(x) \ {-}{-}{>}\ g(x))$
⟨*proof*⟩

10.5 Problems (mainly) involving equality or functions

48

lemma $(a{=}b \mid c{=}d)\ \&\ (a{=}c \mid b{=}d) \ {-}{-}{>}\ a{=}d \mid b{=}c$
⟨*proof*⟩

49 NOT PROVED AUTOMATICALLY. Hard because it involves substitution for Vars the type constraint ensures that x,y,z have the same type as a,b,u.

lemma $(\exists x\ y{::}'a.\ \forall z.\ z{=}x \mid z{=}y)\ \&\ P(a)\ \&\ P(b)\ \&\ a{\sim}{=}b$
${-}{-}{>}\ (\forall u{::}'a.\ P(u))$
⟨*proof*⟩

50. (What has this to do with equality?)

lemma $(\forall x.\ P(a,x) \mid (\forall y.\ P(x,y))) \ {-}{-}{>}\ (\exists x.\ \forall y.\ P(x,y))$
⟨*proof*⟩

51

lemma $(\exists z\ w.\ \forall x\ y.\ P(x,y) \ {<}{-}{>}\ (x{=}z\ \&\ y{=}w)) \ {-}{-}{>}$
$(\exists z.\ \forall x.\ \exists w.\ (\forall y.\ P(x,y) \ {<}{-}{>}\ y{=}w) \ {<}{-}{>}\ x{=}z)$
⟨*proof*⟩

52

Almost the same as 51.

lemma $(\exists z\ w.\ \forall x\ y.\ P(x,y) \ {<}{-}{>}\ (x{=}z\ \&\ y{=}w)) \ {-}{-}{>}$
$(\exists w.\ \forall y.\ \exists z.\ (\forall x.\ P(x,y) \ {<}{-}{>}\ x{=}z) \ {<}{-}{>}\ y{=}w)$
⟨*proof*⟩

55

Non-equational version, from Manthey and Bry, CADE-9 (Springer, 1988). fast DISCOVERS who killed Agatha.

lemma *lives*(*agatha*) & *lives*(*butler*) & *lives*(*charles*) &
(*killed*(*agatha*,*agatha*) | *killed*(*butler*,*agatha*) | *killed*(*charles*,*agatha*)) &
(∀ *x y*. *killed*(*x*,*y*) --> *hates*(*x*,*y*) & ~*richer*(*x*,*y*)) &
(∀ *x*. *hates*(*agatha*,*x*) --> ~*hates*(*charles*,*x*)) &
(*hates*(*agatha*,*agatha*) & *hates*(*agatha*,*charles*)) &
(∀ *x*. *lives*(*x*) & ~*richer*(*x*,*agatha*) --> *hates*(*butler*,*x*)) &
(∀ *x*. *hates*(*agatha*,*x*) --> *hates*(*butler*,*x*)) &
(∀ *x*. ~*hates*(*x*,*agatha*) | ~*hates*(*x*,*butler*) | ~*hates*(*x*,*charles*)) -->
killed(*?who*,*agatha*)
⟨*proof*⟩

56

lemma (∀ *x*. (∃ *y*. *P*(*y*) & *x*=*f*(*y*)) --> *P*(*x*)) <-> (∀ *x*. *P*(*x*) --> *P*(*f*(*x*)))
⟨*proof*⟩

57

lemma *P*(*f*(*a*,*b*), *f*(*b*,*c*)) & *P*(*f*(*b*,*c*), *f*(*a*,*c*)) &
(∀ *x y z*. *P*(*x*,*y*) & *P*(*y*,*z*) --> *P*(*x*,*z*)) --> *P*(*f*(*a*,*b*), *f*(*a*,*c*))
⟨*proof*⟩

58 NOT PROVED AUTOMATICALLY

lemma (∀ *x y*. *f*(*x*)=*g*(*y*)) --> (∀ *x y*. *f*(*f*(*x*))=*f*(*g*(*y*)))
⟨*proof*⟩

59

lemma (∀ *x*. *P*(*x*) <-> ~*P*(*f*(*x*))) --> (∃ *x*. *P*(*x*) & ~*P*(*f*(*x*)))
⟨*proof*⟩

60

lemma ∀ *x*. *P*(*x*,*f*(*x*)) <-> (∃ *y*. (∀ *z*. *P*(*z*,*y*) --> *P*(*z*,*f*(*x*))) & *P*(*x*,*y*))
⟨*proof*⟩

62 as corrected in JAR 18 (1997), page 135

lemma (∀ *x*. *p*(*a*) & (*p*(*x*) --> *p*(*f*(*x*))) --> *p*(*f*(*f*(*x*)))) <->
(∀ *x*. (~*p*(*a*) | *p*(*x*) | *p*(*f*(*f*(*x*)))) &
(~*p*(*a*) | ~*p*(*f*(*x*)) | *p*(*f*(*f*(*x*)))))
⟨*proof*⟩

From Davis, Obvious Logical Inferences, IJCAI-81, 530-531 fast indeed copes!

lemma (∀ *x*. *F*(*x*) & ~*G*(*x*) --> (∃ *y*. *H*(*x*,*y*) & *J*(*y*))) &
(∃ *x*. *K*(*x*) & *F*(*x*) & (∀ *y*. *H*(*x*,*y*) --> *K*(*y*))) &
(∀ *x*. *K*(*x*) --> ~*G*(*x*)) --> (∃ *x*. *K*(*x*) & *J*(*x*))
⟨*proof*⟩

From Rudnicki, Obvious Inferences, JAR 3 (1987), 383-393. It does seem obvious!

```
lemma (∀x. F(x) & ~G(x) --> (∃y. H(x,y) & J(y))) &
      (∃x. K(x) & F(x) & (∀y. H(x,y) --> K(y)))  &
      (∀x. K(x) --> ~G(x))   -->   (∃x. K(x) --> ~G(x))
⟨proof⟩
```

Halting problem: Formulation of Li Dafa (AAR Newsletter 27, Oct 1994.) author U. Egly

```
lemma ((∃x. A(x) & (∀y. C(y) --> (∀z. D(x,y,z)))) -->
   (∃w. C(w) & (∀y. C(y) --> (∀z. D(w,y,z)))))
  &
  (∀w. C(w) & (∀u. C(u) --> (∀v. D(w,u,v))) -->
        (∀y z.
            (C(y) &  P(y,z) --> Q(w,y,z) & OO(w,g)) &
            (C(y) & ~P(y,z) --> Q(w,y,z) & OO(w,b))))
  &
  (∀w. C(w) &
    (∀y z.
        (C(y) & P(y,z) --> Q(w,y,z) & OO(w,g)) &
        (C(y) & ~P(y,z) --> Q(w,y,z) & OO(w,b))) -->
    (∃v. C(v) &
          (∀y. ((C(y) & Q(w,y,y)) & OO(w,g) --> ~P(v,y)) &
                  ((C(y) & Q(w,y,y)) & OO(w,b) --> P(v,y) & OO(v,b)))))
   -->
   ~ (∃x. A(x) & (∀y. C(y) --> (∀z. D(x,y,z))))
⟨proof⟩
```

Halting problem II: credited to M. Bruschi by Li Dafa in JAR 18(1), p.105

```
lemma ((∃x. A(x) & (∀y. C(y) --> (∀z. D(x,y,z)))) -->
   (∃w. C(w) & (∀y. C(y) --> (∀z. D(w,y,z)))))
  &
  (∀w. C(w) & (∀u. C(u) --> (∀v. D(w,u,v))) -->
        (∀y z.
            (C(y) &  P(y,z) --> Q(w,y,z) & OO(w,g)) &
            (C(y) & ~P(y,z) --> Q(w,y,z) & OO(w,b))))
  &
  ((∃w. C(w) & (∀y. (C(y) &  P(y,y) --> Q(w,y,y) & OO(w,g)) &
                    (C(y) & ~P(y,y) --> Q(w,y,y) & OO(w,b))))
   -->
   (∃v. C(v) & (∀y. (C(y) &  P(y,y) --> P(v,y) & OO(v,g)) &
                    (C(y) & ~P(y,y) --> P(v,y) & OO(v,b)))))
  -->
  ((∃v. C(v) & (∀y. (C(y) &  P(y,y) --> P(v,y) & OO(v,g)) &
                    (C(y) & ~P(y,y) --> P(v,y) & OO(v,b))))
   -->
   (∃u. C(u) & (∀y. (C(y) &  P(y,y) --> ~P(u,y)) &
                    (C(y) & ~P(y,y) --> P(u,y) & OO(u,b)))))
   -->
   ~ (∃x. A(x) & (∀y. C(y) --> (∀z. D(x,y,z))))
⟨proof⟩
```

Challenge found on info-hol

lemma $\forall x.\ \exists v\ w.\ \forall y\ z.\ P(x)\ \&\ Q(y)\ \text{-->}\ (P(v)\ |\ R(w))\ \&\ (R(z)\ \text{-->}\ Q(v))$
⟨*proof*⟩

Attributed to Lewis Carroll by S. G. Pulman. The first or last assumption can be deleted.

lemma $(\forall x.\ \mathit{honest}(x)\ \&\ \mathit{industrious}(x)\ \text{-->}\ \mathit{healthy}(x))\ \&$
$\sim\ (\exists x.\ \mathit{grocer}(x)\ \&\ \mathit{healthy}(x))\ \&$
$(\forall x.\ \mathit{industrious}(x)\ \&\ \mathit{grocer}(x)\ \text{-->}\ \mathit{honest}(x))\ \&$
$(\forall x.\ \mathit{cyclist}(x)\ \text{-->}\ \mathit{industrious}(x))\ \&$
$(\forall x.\ {\sim}\mathit{healthy}(x)\ \&\ \mathit{cyclist}(x)\ \text{-->}\ {\sim}\mathit{honest}(x))$
$\text{-->}\ (\forall x.\ \mathit{grocer}(x)\ \text{-->}\ {\sim}\mathit{cyclist}(x))$
⟨*proof*⟩

end

11 First-Order Logic: propositional examples (classical version)

theory *Propositional-Cla*
imports *FOL*
begin

commutative laws of & and |

lemma $P\ \&\ Q\ \text{-->}\ Q\ \&\ P$
⟨*proof*⟩

lemma $P\ |\ Q\ \text{-->}\ Q\ |\ P$
⟨*proof*⟩

associative laws of & and |

lemma $(P\ \&\ Q)\ \&\ R\ \text{-->}\ P\ \&\ (Q\ \&\ R)$
⟨*proof*⟩

lemma $(P\ |\ Q)\ |\ R\ \text{-->}\ P\ |\ (Q\ |\ R)$
⟨*proof*⟩

distributive laws of & and |

lemma $(P\ \&\ Q)\ |\ R\ \text{-->}\ (P\ |\ R)\ \&\ (Q\ |\ R)$
⟨*proof*⟩

lemma $(P\ |\ R)\ \&\ (Q\ |\ R)\ \text{-->}\ (P\ \&\ Q)\ |\ R$

⟨*proof*⟩

lemma $(P \mid Q)\ \&\ R\ \ {-}{-}{>}\ (P\ \&\ R) \mid (Q\ \&\ R)$
⟨*proof*⟩

lemma $(P\ \&\ R) \mid (Q\ \&\ R)\ \ {-}{-}{>}\ (P \mid Q)\ \&\ R$
⟨*proof*⟩

Laws involving implication

lemma $(P{-}{-}{>}R)\ \&\ (Q{-}{-}{>}R)\ {<}{-}{>}\ (P|Q\ {-}{-}{>}\ R)$
⟨*proof*⟩

lemma $(P\ \&\ Q\ {-}{-}{>}\ R)\ {<}{-}{>}\ (P{-}{-}{>}\ (Q{-}{-}{>}R))$
⟨*proof*⟩

lemma $((P{-}{-}{>}R){-}{-}{>}R)\ {-}{-}{>}\ ((Q{-}{-}{>}R){-}{-}{>}R)\ {-}{-}{>}\ (P\&Q{-}{-}{>}R)\ {-}{-}{>}$
R
⟨*proof*⟩

lemma $\sim(P{-}{-}{>}R)\ {-}{-}{>}\ \sim(Q{-}{-}{>}R)\ {-}{-}{>}\ \sim(P\&Q{-}{-}{>}R)$
⟨*proof*⟩

lemma $(P\ {-}{-}{>}\ Q\ \&\ R)\ {<}{-}{>}\ (P{-}{-}{>}Q)\ \ \&\ \ (P{-}{-}{>}R)$
⟨*proof*⟩

Propositions-as-types

— The combinator K
lemma $P\ {-}{-}{>}\ (Q\ {-}{-}{>}\ P)$
⟨*proof*⟩
lemma $(P{-}{-}{>}Q{-}{-}{>}R)\ \ {-}{-}{>}\ (P{-}{-}{>}Q)\ {-}{-}{>}\ (P{-}{-}{>}R)$
⟨*proof*⟩
lemma $(P{-}{-}{>}Q) \mid (P{-}{-}{>}R)\ \ {-}{-}{>}\ \ (P\ {-}{-}{>}\ Q \mid R)$
⟨*proof*⟩

lemma $(P{-}{-}{>}Q)\ \ {-}{-}{>}\ \ (\sim Q\ {-}{-}{>}\ \sim P)$
⟨*proof*⟩

Schwichtenberg's examples (via T. Nipkow)

lemma *stab-imp*: $(((Q{-}{-}{>}R){-}{-}{>}R){-}{-}{>}Q)\ {-}{-}{>}\ (((P{-}{-}{>}Q){-}{-}{>}R){-}{-}{>}R){-}{-}{>}P{-}{-}{>}Q$
⟨*proof*⟩

lemma *stab-to-peirce*:
$(((P\ {-}{-}{>}\ R)\ {-}{-}{>}\ R)\ {-}{-}{>}\ P)\ {-}{-}{>}\ (((Q\ {-}{-}{>}\ R)\ {-}{-}{>}\ R)\ {-}{-}{>}\ Q)$
$\qquad {-}{-}{>}\ ((P\ {-}{-}{>}\ Q)\ {-}{-}{>}\ P)\ {-}{-}{>}\ P$
⟨*proof*⟩

lemma *peirce-imp1*: $(((Q\ {-}{-}{>}\ R)\ {-}{-}{>}\ Q)\ {-}{-}{>}\ Q)$
$\qquad {-}{-}{>}\ (((P\ {-}{-}{>}\ Q)\ {-}{-}{>}\ R)\ {-}{-}{>}\ P\ {-}{-}{>}\ Q)\ {-}{-}{>}\ P\ {-}{-}{>}\ Q$
⟨*proof*⟩

lemma *peirce-imp2*: $(((P \dashrightarrow R) \dashrightarrow P) \dashrightarrow P) \dashrightarrow ((P \dashrightarrow Q \dashrightarrow R) \dashrightarrow P) \dashrightarrow P$
⟨*proof*⟩

lemma *mints*: $((((P \dashrightarrow Q) \dashrightarrow P) \dashrightarrow P) \dashrightarrow Q) \dashrightarrow Q$
⟨*proof*⟩

lemma *mints-solovev*: $(P \dashrightarrow (Q \dashrightarrow R) \dashrightarrow Q) \dashrightarrow ((P \dashrightarrow Q) \dashrightarrow R) \dashrightarrow R$
⟨*proof*⟩

lemma *tatsuta*: $(((P7 \dashrightarrow P1) \dashrightarrow P10) \dashrightarrow P4 \dashrightarrow P5)$
$\dashrightarrow (((P8 \dashrightarrow P2) \dashrightarrow P9) \dashrightarrow P3 \dashrightarrow P10)$
$\dashrightarrow (P1 \dashrightarrow P8) \dashrightarrow P6 \dashrightarrow P7$
$\dashrightarrow (((P3 \dashrightarrow P2) \dashrightarrow P9) \dashrightarrow P4)$
$\dashrightarrow (P1 \dashrightarrow P3) \dashrightarrow (((P6 \dashrightarrow P1) \dashrightarrow P2) \dashrightarrow P9) \dashrightarrow P5$
⟨*proof*⟩

lemma *tatsuta1*: $(((P8 \dashrightarrow P2) \dashrightarrow P9) \dashrightarrow P3 \dashrightarrow P10)$
$\dashrightarrow (((P3 \dashrightarrow P2) \dashrightarrow P9) \dashrightarrow P4)$
$\dashrightarrow (((P6 \dashrightarrow P1) \dashrightarrow P2) \dashrightarrow P9)$
$\dashrightarrow (((P7 \dashrightarrow P1) \dashrightarrow P10) \dashrightarrow P4 \dashrightarrow P5)$
$\dashrightarrow (P1 \dashrightarrow P3) \dashrightarrow (P1 \dashrightarrow P8) \dashrightarrow P6 \dashrightarrow P7 \dashrightarrow P5$
⟨*proof*⟩

end

12 First-Order Logic: quantifier examples (classical version)

theory *Quantifiers-Cla*
imports *FOL*
begin

lemma $(ALL\ x\ y.\ P(x,y)) \dashrightarrow (ALL\ y\ x.\ P(x,y))$
⟨*proof*⟩

lemma $(EX\ x\ y.\ P(x,y)) \dashrightarrow (EX\ y\ x.\ P(x,y))$
⟨*proof*⟩
lemma $(ALL\ x.\ P(x)) \mid (ALL\ x.\ Q(x)) \dashrightarrow (ALL\ x.\ P(x) \mid Q(x))$
⟨*proof*⟩

lemma $(ALL\ x.\ P \dashrightarrow Q(x)) <\!\!-\!\!> (P \dashrightarrow (ALL\ x.\ Q(x)))$
⟨*proof*⟩

lemma $(ALL\ x.\ P(x){-}{-}{>}Q)\ \ {<}{-}{>}\ \ ((EX\ x.\ P(x))\ {-}{-}{>}\ Q)$
  ⟨*proof*⟩

Some harder ones

lemma $(EX\ x.\ P(x)\ |\ Q(x))\ {<}{-}{>}\ (EX\ x.\ P(x))\ |\ (EX\ x.\ Q(x))$
  ⟨*proof*⟩
lemma $(EX\ x.\ P(x)\&Q(x))\ {-}{-}{>}\ (EX\ x.\ P(x))\ \ \&\ \ (EX\ x.\ Q(x))$
  ⟨*proof*⟩

Basic test of quantifier reasoning

— TRUE
lemma $(EX\ y.\ ALL\ x.\ Q(x,y))\ {-}{-}{>}\ \ (ALL\ x.\ EX\ y.\ Q(x,y))$
  ⟨*proof*⟩

lemma $(ALL\ x.\ Q(x))\ \ {-}{-}{>}\ \ (EX\ x.\ Q(x))$
  ⟨*proof*⟩

The following should fail, as they are false!

lemma $(ALL\ x.\ EX\ y.\ Q(x,y))\ \ {-}{-}{>}\ \ (EX\ y.\ ALL\ x.\ Q(x,y))$
  ⟨*proof*⟩

lemma $(EX\ x.\ Q(x))\ \ {-}{-}{>}\ \ (ALL\ x.\ Q(x))$
  ⟨*proof*⟩

lemma $P(?a)\ {-}{-}{>}\ (ALL\ x.\ P(x))$
  ⟨*proof*⟩

lemma $(P(?a)\ {-}{-}{>}\ (ALL\ x.\ Q(x)))\ {-}{-}{>}\ (ALL\ x.\ P(x)\ {-}{-}{>}\ Q(x))$
  ⟨*proof*⟩

Back to things that are provable . . .

lemma $(ALL\ x.\ P(x){-}{-}{>}Q(x))\ \&\ (EX\ x.\ P(x))\ {-}{-}{>}\ (EX\ x.\ Q(x))$
  ⟨*proof*⟩
lemma $(P\ {-}{-}{>}\ (EX\ x.\ Q(x)))\ \&\ P\ {-}{-}{>}\ (EX\ x.\ Q(x))$
  ⟨*proof*⟩

lemma $(ALL\ x.\ P(x){-}{-}{>}Q(f(x)))\ \&\ (ALL\ x.\ Q(x){-}{-}{>}R(g(x)))\ \&\ P(d)\ {-}{-}{>}$
$R(?a)$
  ⟨*proof*⟩

lemma $(ALL\ x.\ Q(x))\ \ {-}{-}{>}\ \ (EX\ x.\ Q(x))$
  ⟨*proof*⟩

Some slow ones

— Principia Mathematica *11.53
lemma $(ALL\ x\ y.\ P(x)\ {-}{-}{>}\ Q(y))\ {<}{-}{>}\ ((EX\ x.\ P(x))\ {-}{-}{>}\ (ALL\ y.\ Q(y)))$
  ⟨*proof*⟩

lemma (*EX x y. P(x) & Q(x,y)*) <−> (*EX x. P(x)* & (*EX y. Q(x,y)*))
⟨*proof*⟩

lemma (*EX y. ALL x. P(x)* −−> *Q(x,y)*) −−> (*ALL x. P(x)* −−> (*EX y. Q(x,y)*))
⟨*proof*⟩

end

theory *Miniscope*
imports *FOL*
begin

lemmas *ccontr* = *FalseE* [*THEN classical*]

12.1 Negation Normal Form

12.1.1 de Morgan laws

lemma *demorgans*:
~(P&Q) <−> ~P | ~Q
~(P|Q) <−> ~P & ~Q
~~P <−> P
!!P. ~(ALL x. P(x)) <−> (EX x. ~P(x))
!!P. ~(EX x. P(x)) <−> (ALL x. ~P(x))
⟨*proof*⟩

lemma *nnf-simps*:
(P−−>Q) <−> (~P | Q)
~(P−−>Q) <−> (P & ~Q)
(P<−>Q) <−> (~P | Q) & (~Q | P)
~(P<−>Q) <−> (P | Q) & (~P | ~Q)
⟨*proof*⟩

12.1.2 Pushing in the existential quantifiers

lemma *ex-simps*:
(EX x. P) <−> P
!!P Q. (EX x. P(x) & Q) <−> (EX x. P(x)) & Q
!!P Q. (EX x. P & Q(x)) <−> P & (EX x. Q(x))
!!P Q. (EX x. P(x) | Q(x)) <−> (EX x. P(x)) | (EX x. Q(x))
!!P Q. (EX x. P(x) | Q) <−> (EX x. P(x)) | Q
!!P Q. (EX x. P | Q(x)) <−> P | (EX x. Q(x))

⟨*proof*⟩

12.1.3 Pushing in the universal quantifiers

lemma *all-simps*:
  $(ALL\ x.\ P) <-> P$
  $!!P\ Q.\ (ALL\ x.\ P(x)\ \&\ Q(x)) <-> (ALL\ x.\ P(x))\ \&\ (ALL\ x.\ Q(x))$
  $!!P\ Q.\ (ALL\ x.\ P(x)\ \&\ Q) <-> (ALL\ x.\ P(x))\ \&\ Q$
  $!!P\ Q.\ (ALL\ x.\ P\ \&\ Q(x)) <-> P\ \&\ (ALL\ x.\ Q(x))$
  $!!P\ Q.\ (ALL\ x.\ P(x)\ |\ Q) <-> (ALL\ x.\ P(x))\ |\ Q$
  $!!P\ Q.\ (ALL\ x.\ P\ |\ Q(x)) <-> P\ |\ (ALL\ x.\ Q(x))$
  ⟨*proof*⟩

lemmas *mini-simps* = *demorgans nnf-simps ex-simps all-simps*

⟨*ML*⟩

end

13 First-Order Logic: the 'if' example

theory *If* **imports** *FOL* **begin**

constdefs
  $if :: [o,o,o]=>o$
  $if(P,Q,R) == P\&Q\ |\ \tilde{}P\&R$

lemma *ifI*:
  $[|\ P ==> Q;\ \tilde{}P ==> R\ |] ==> if(P,Q,R)$
⟨*proof*⟩

lemma *ifE*:
  $[|\ if(P,Q,R);\ \ [|\ P;\ Q\ |] ==> S;\ [|\ \tilde{}P;\ R\ |] ==> S\ |] ==> S$
⟨*proof*⟩

lemma *if-commute*: $if(P,\ if(Q,A,B),\ if(Q,C,D)) <-> if(Q,\ if(P,A,C),\ if(P,B,D))$
⟨*proof*⟩

Trying again from the beginning in order to use *blast*

declare *ifI* [*intro*!]
declare *ifE* [*elim*!]

lemma *if-commute*: $if(P,\ if(Q,A,B),\ if(Q,C,D)) <-> if(Q,\ if(P,A,C),\ if(P,B,D))$
⟨*proof*⟩

lemma $if(if(P,Q,R),\ A,\ B) <-> if(P,\ if(Q,A,B),\ if(R,A,B))$
⟨*proof*⟩

Trying again from the beginning in order to prove from the definitions

lemma *if*(*if*(*P*,*Q*,*R*), *A*, *B*) <−> *if*(*P*, *if*(*Q*,*A*,*B*), *if*(*R*,*A*,*B*))
⟨*proof*⟩

An invalid formula. High-level rules permit a simpler diagnosis

lemma *if*(*if*(*P*,*Q*,*R*), *A*, *B*) <−> *if*(*P*, *if*(*Q*,*A*,*B*), *if*(*R*,*B*,*A*))
⟨*proof*⟩

Trying again from the beginning in order to prove from the definitions

lemma *if*(*if*(*P*,*Q*,*R*), *A*, *B*) <−> *if*(*P*, *if*(*Q*,*A*,*B*), *if*(*R*,*B*,*A*))
⟨*proof*⟩

end

theory *NatClass*
imports *FOL*
begin

This is an abstract version of theory *Nat*. Instead of axiomatizing a single type *nat* we define the class of all these types (up to isomorphism).

Note: The *rec* operator had to be made *monomorphic*, because class axioms may not contain more than one type variable.

consts
  0 :: *'a* (*0*)
  Suc :: *'a* => *'a*
  rec :: [*'a*, *'a*, [*'a*, *'a*] => *'a*] => *'a*

axclass
  nat < *term*
  induct: [| *P*(*0*); !!*x*. *P*(*x*) ==> *P*(*Suc*(*x*)) |] ==> *P*(*n*)
  Suc-inject: *Suc*(*m*) = *Suc*(*n*) ==> *m* = *n*
  Suc-neq-0: *Suc*(*m*) = *0* ==> *R*
  rec-0: *rec*(*0*, *a*, *f*) = *a*
  rec-Suc: *rec*(*Suc*(*m*), *a*, *f*) = *f*(*m*, *rec*(*m*, *a*, *f*))

definition
  add :: [*'a*::*nat*, *'a*] => *'a* (**infixl** + *60*) **where**
  m + *n* = *rec*(*m*, *n*, %*x* *y*. *Suc*(*y*))

lemma *Suc-n-not-n*: *Suc*(*k*) ~= (*k*::*'a*::*nat*)
⟨*proof*⟩

lemma (*k*+*m*)+*n* = *k*+(*m*+*n*)
⟨*proof*⟩

lemma *add-0* [*simp*]: *0*+*n* = *n*

⟨*proof*⟩

lemma *add-Suc* [*simp*]: $Suc(m)+n = Suc(m+n)$
⟨*proof*⟩

lemma *add-assoc*: $(k+m)+n = k+(m+n)$
⟨*proof*⟩

lemma *add-0-right*: $m+0 = m$
⟨*proof*⟩

lemma *add-Suc-right*: $m+Suc(n) = Suc(m+n)$
⟨*proof*⟩

lemma
  assumes *prem*: !!n. $f(Suc(n)) = Suc(f(n))$
  shows $f(i+j) = i+f(j)$
⟨*proof*⟩

end

14 Example of Declaring an Oracle

theory *IffOracle*
imports *FOL*
begin

14.1 Oracle declaration

This oracle makes tautologies of the form $P <-> P <-> P <-> P$. The length is specified by an integer, which is checked to be even and positive.

⟨*ML*⟩

14.2 Oracle as low-level rule

⟨*ML*⟩

These oracle calls had better fail.

⟨*ML*⟩

14.3 Oracle as proof method

⟨*ML*⟩

lemma $A <-> A$
  ⟨*proof*⟩

lemma $A <-> A <-> A <-> A <-> A <-> A <-> A <-> A <-> A$
$<-> A$
  $\langle proof \rangle$

lemma $A <-> A <-> A <-> A <-> A$
  $\langle proof \rangle$

lemma A
  $\langle proof \rangle$

end