Miscellaneous FOL Examples

October 1, 2005

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
$Trueprop :: o \Rightarrow prop \quad (\text{-}\ 5)$

1.2 Propositional logic

consts
$false :: o \quad (\bot)$

$imp :: o \Rightarrow o \Rightarrow o$ (**infixr** $\longrightarrow$ *25*)
$conj :: o \Rightarrow o \Rightarrow o$ (**infixr** $\wedge$ *35*)
$disj :: o \Rightarrow o \Rightarrow o$ (**infixr** $\vee$ *30*)

axioms
falseE [*elim*]: $\bot \Longrightarrow A$

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

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

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

theorem *conjE* [*elim*]: $A \wedge B \Longrightarrow (A \Longrightarrow B \Longrightarrow C) \Longrightarrow C$
⟨*proof*⟩

constdefs
true :: o ($\top$)
$\top \equiv \bot \longrightarrow \bot$
not :: $o \Rightarrow o$ ($\neg$ - [*40*] *40*)
$\neg A \equiv A \longrightarrow \bot$
iff :: $o \Rightarrow o \Rightarrow o$ (**infixr** $\longleftrightarrow$ *25*)
$A \longleftrightarrow B \equiv (A \longrightarrow B) \wedge (B \longrightarrow A)$

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

consts
  $equal :: i \Rightarrow i \Rightarrow o$ (**infixl** = *50*)

axioms
  refl [*intro*]: $x = x$
  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

consts
  $All :: (i \Rightarrow o) \Rightarrow o$ (**binder** $\forall$ *10*)
  $Ex :: (i \Rightarrow o) \Rightarrow o$ (**binder** $\exists$ *10*)

axioms
  allI [*intro*]: $(\bigwedge x.\ P(x)) \Longrightarrow \forall x.\ P(x)$
  allD [*dest*]: $\forall x.\ P(x) \Longrightarrow P(a)$

  exI [*intro*]: $P(a) \Longrightarrow \exists x.\ P(x)$
  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*

consts
  Zero :: *nat* (*0*)

Suc :: *nat* => *nat*
rec :: [*nat*, *'a*, [*nat*, *'a*] => *'a*] => *'a*

axioms
induct [*case-names 0 Suc*, *induct type*: *nat*]:
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))*

lemma *Suc-n-not-n*: *Suc(k)* ≠ *k*
⟨*proof*⟩

constdefs
add :: [*nat*, *nat*] => *nat* (**infixl** + *60*)
m + *n* == *rec(m*, *n*, λ*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 (!!*n*. *f(Suc(n))* = *Suc(f(n)))* ==> *f(i* + *j)* = *i* + *f(j)*
⟨*proof*⟩

end

3 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*))
add-def: *m*+*n* == *rec*(*m*, *n*, %*x y*. *Suc*(*y*))

⟨*ML*⟩

end

4 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*)

⟨*ML*⟩

end

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

5.1 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 $\sim (EX\ x.\ ALL\ y.\ mem(y,x) <-> \sim mem(x,x))$
⟨*proof*⟩

Problem 4.1: hopeless!

lemma $(ALL\ x.\ p(x) --> p(h(x)) \mid p(g(x)))\ \&\ (EX\ x.\ p(x))\ \&\ (ALL\ x.\ \sim p(h(x)))$
$--> (EX\ x.\ p(g(g(g(g(g(x)))))))$
⟨*proof*⟩

5.2 Intuitionistic FOL: propositional problems based on Pelletier.

1

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

2

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

3

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

4

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

5

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

6

lemma $\sim\sim(P \mid \sim P)$
⟨*proof*⟩

7

lemma $\sim\sim(P \mid \sim\sim\sim P)$
⟨*proof*⟩

8. Peirce's law

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

9

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

10

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

5.3 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*⟩

5.4 ****Examples with quantifiers****

5.5 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 $(ALL\ x.\ P(x)) \mid Q \;\; --> \;\; (ALL\ x.\ P(x) \mid Q)$
⟨*proof*⟩

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

5.6 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 $(P \dashrightarrow (EX\ x.\ Q(x))) \dashrightarrow (EX\ x.\ P \dashrightarrow Q(x))$
⟨*proof*⟩

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

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

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

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

5.7 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 $\sim\sim(EX\ y.\ ALL\ x.\ P(y) \dashrightarrow P(x))$
⟨*proof*⟩

19

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

20

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

21

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

22

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

23

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

24

lemma $\sim(EX\ x.\ S(x) \& Q(x)) \ \&\ (ALL\ x.\ P(x) \dashrightarrow Q(x) | R(x)) \ \&$
$(\sim(EX\ x.\ P(x)) \dashrightarrow (EX\ x.\ Q(x))) \ \&\ (ALL\ x.\ Q(x) | R(x) \dashrightarrow S(x))$
$\dashrightarrow \sim\sim(EX\ x.\ P(x) \& R(x))$⟨*proof*⟩

25

lemma $(EX\ x.\ P(x))\ \&$
$\quad (ALL\ x.\ L(x) \dashrightarrow \sim (M(x)\ \&\ R(x)))\ \&$
$\quad (ALL\ x.\ P(x) \dashrightarrow (M(x)\ \&\ L(x)))\ \&$
$\quad ((ALL\ x.\ P(x) \dashrightarrow Q(x)) \mid (EX\ x.\ P(x) \& R(x)))$
$\dashrightarrow (EX\ x.\ Q(x) \& P(x))$
⟨*proof*⟩

26

lemma $(\sim\sim(EX\ x.\ p(x)) <\!\!\dashrightarrow \sim\sim(EX\ x.\ q(x)))\ \&$
$\quad (ALL\ x.\ ALL\ y.\ p(x)\ \&\ q(y) \dashrightarrow (r(x) <\!\!\dashrightarrow s(y)))$
$\dashrightarrow ((ALL\ x.\ p(x) \dashrightarrow r(x)) <\!\!\dashrightarrow (ALL\ x.\ q(x) \dashrightarrow s(x)))$
⟨*proof*⟩

27

lemma $(EX\ x.\ P(x)\ \&\ \sim Q(x))\ \&$
$\quad (ALL\ x.\ P(x) \dashrightarrow R(x))\ \&$
$\quad (ALL\ x.\ M(x)\ \&\ L(x) \dashrightarrow P(x))\ \&$
$\quad ((EX\ x.\ R(x)\ \&\ \sim Q(x)) \dashrightarrow (ALL\ x.\ L(x) \dashrightarrow \sim R(x)))$
$\dashrightarrow (ALL\ x.\ M(x) \dashrightarrow \sim L(x))$
⟨*proof*⟩

28. AMENDED

lemma $(ALL\ x.\ P(x) \dashrightarrow (ALL\ x.\ Q(x)))\ \&$
$\quad (\sim\sim(ALL\ x.\ Q(x) \mid R(x)) \dashrightarrow (EX\ x.\ Q(x) \& S(x)))\ \&$
$\quad (\sim\sim(EX\ x.\ S(x)) \dashrightarrow (ALL\ x.\ L(x) \dashrightarrow M(x)))$
$\dashrightarrow (ALL\ x.\ P(x)\ \&\ L(x) \dashrightarrow M(x))$
⟨*proof*⟩

29. Essentially the same as Principia Mathematica *11.71

lemma $(EX\ x.\ P(x))\ \&\ (EX\ y.\ Q(y))$
$\dashrightarrow ((ALL\ x.\ P(x) \dashrightarrow R(x))\ \&\ (ALL\ y.\ Q(y) \dashrightarrow S(y)) \quad <\!\!\dashrightarrow$
$\quad (ALL\ x\ y.\ P(x)\ \&\ Q(y) \dashrightarrow R(x)\ \&\ S(y)))$
⟨*proof*⟩

30

lemma $(ALL\ x.\ (P(x) \mid Q(x)) \dashrightarrow \sim R(x))\ \&$
$\quad (ALL\ x.\ (Q(x) \dashrightarrow \sim S(x)) \dashrightarrow P(x)\ \&\ R(x))$
$\dashrightarrow (ALL\ x.\ \sim\sim S(x))$
⟨*proof*⟩

31

lemma $\sim(EX\ x.\ P(x)\ \&\ (Q(x) \mid R(x)))\ \&$
$\quad (EX\ x.\ L(x)\ \&\ P(x))\ \&$
$\quad (ALL\ x.\ \sim R(x) \dashrightarrow M(x))$
$\dashrightarrow (EX\ x.\ L(x)\ \&\ M(x))$
⟨*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 $(ALL\ x.\ \sim\sim(P(a)\ \&\ (P(x)-->P(b))-->P(c)))\ <->$
$(ALL\ x.\ \sim\sim((\sim P(a)\ |\ P(x)\ |\ P(c))\ \&\ (\sim P(a)\ |\ \sim P(b)\ |\ P(c))))$
⟨*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 $(ALL\ z.\ EX\ w.\ ALL\ x.\ EX\ y.$
$\sim\sim(P(x,z)-->P(y,w))\ \&\ P(y,z)\ \&\ (P(y,w)\ -->\ (EX\ u.\ Q(u,w))))\ \&$
$(ALL\ x\ z.\ \sim P(x,z)\ -->\ (EX\ y.\ Q(y,z)))\ \&$
$(\sim\sim(EX\ x\ y.\ Q(x,y))\ -->\ (ALL\ x.\ R(x,x)))$
$-->\ \sim\sim(ALL\ x.\ EX\ y.\ R(x,y))$
⟨*proof*⟩

39

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

40. AMENDED

lemma $(EX\ y.\ ALL\ x.\ F(x,y)\ <->\ F(x,x))\ -->$
$\sim(ALL\ x.\ EX\ y.\ ALL\ z.\ F(z,y)\ <->\ \sim\ F(z,x))$
⟨*proof*⟩

44

lemma $(ALL\ x.\ f(x)\ -->$
$(EX\ y.\ g(y)\ \&\ h(x,y)\ \&\ (EX\ y.\ g(y)\ \&\ \sim\ h(x,y))))\ \&$
$(EX\ x.\ j(x)\ \&\ (ALL\ y.\ g(y)\ -->\ h(x,y)))$
$-->\ (EX\ x.\ j(x)\ \&\ \sim f(x))$
⟨*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

6 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.

6.1 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 $P \mid \sim P$
⟨*proof*⟩

7

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

8. Peirce's law

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

9

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

6.2 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*⟩

6.3 Problems requiring quantifier duplication

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

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

Needs multiple instantiation of ALL.

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

Needs double instantiation of the quantifier

lemma $\exists x.\ P(x) --> P(a)\ \&\ P(b)$
⟨*proof*⟩

lemma $\exists z.\ P(z) --> (\forall x.\ P(x))$
⟨*proof*⟩

lemma $\exists x.\ (\exists y.\ P(y)) --> P(x)$
⟨*proof*⟩

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

lemma $\exists x\ x'.\ \forall y.\ \exists z\ z'.$
$(\sim P(y,y) \mid P(x,x) \mid \sim S(z,x))\ \&$
$(S(x,y) \mid \sim S(y,z) \mid Q(z',z'))\ \&$
$(Q(x',y) \mid \sim Q(y,z') \mid S(x',x'))$
⟨*proof*⟩

6.4 Hard examples with quantifiers

18

lemma $\exists y.\ \forall x.\ P(y)-->P(x)$
⟨*proof*⟩

19

lemma $\exists x.\ \forall y\ z.\ (P(y)-->Q(z)) --> (P(x)-->Q(x))$
⟨*proof*⟩

20

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

21

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

⟨*proof*⟩

22

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

23

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

24

lemma $\sim(\exists x.\ S(x)\&Q(x)) \ \&\ (\forall x.\ P(x) --> Q(x)|R(x)) \ \&$
$(\sim(\exists x.\ P(x)) --> (\exists x.\ Q(x))) \ \&\ (\forall x.\ Q(x)|R(x) --> S(x))$
$--> (\exists x.\ P(x)\&R(x))$
⟨*proof*⟩

25

lemma $(\exists x.\ P(x)) \ \&$
$(\forall x.\ L(x) --> \sim (M(x) \ \&\ R(x))) \ \&$
$(\forall x.\ P(x) --> (M(x) \ \&\ L(x))) \ \&$
$((\forall x.\ P(x)-->Q(x)) \mid (\exists x.\ P(x)\&R(x)))$
$--> (\exists x.\ Q(x)\&P(x))$
⟨*proof*⟩

26

lemma $((\exists x.\ p(x)) <-> (\exists x.\ q(x))) \ \&$
$(\forall x.\ \forall y.\ p(x) \ \&\ q(y) --> (r(x) <-> s(y)))$
$--> ((\forall x.\ p(x)-->r(x)) <-> (\forall x.\ q(x)-->s(x)))$
⟨*proof*⟩

27

lemma $(\exists x.\ P(x) \ \&\ \sim Q(x)) \ \&$
$(\forall x.\ P(x) --> R(x)) \ \&$
$(\forall x.\ M(x) \ \&\ L(x) --> P(x)) \ \&$
$((\exists x.\ R(x) \ \&\ \sim Q(x)) --> (\forall x.\ L(x) --> \sim R(x)))$
$--> (\forall x.\ M(x) --> \sim L(x))$
⟨*proof*⟩

28. AMENDED

lemma $(\forall x.\ P(x) --> (\forall x.\ Q(x))) \ \&$
$((\forall x.\ Q(x)|R(x)) --> (\exists x.\ Q(x)\&S(x))) \ \&$
$((\exists x.\ S(x)) --> (\forall x.\ L(x) --> M(x)))$
$--> (\forall x.\ P(x) \ \&\ L(x) --> M(x))$
⟨*proof*⟩

29. Essentially the same as Principia Mathematica *11.71

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

$$--> ((\forall x.\ P(x)-->R(x)) \ \&\ (\forall y.\ Q(y)-->S(y)) \quad <-> \\ (\forall x\ y.\ P(x)\ \&\ Q(y)\ --> R(x)\ \&\ S(y)))$$

⟨*proof*⟩

30

lemma $(\forall x.\ P(x) \mid Q(x)\ --> {\sim} R(x)) \ \&$
$(\forall x.\ (Q(x)\ --> {\sim} S(x))\ --> P(x)\ \&\ R(x))$
$--> (\forall x.\ S(x))$

⟨*proof*⟩

31

lemma ${\sim}(\exists x.\ P(x)\ \&\ (Q(x) \mid R(x)))\ \&$
$(\exists x.\ L(x)\ \&\ P(x))\ \&$
$(\forall x.\ {\sim} R(x)\ --> M(x))$
$--> (\exists x.\ L(x)\ \&\ M(x))$

⟨*proof*⟩

32

lemma $(\forall x.\ P(x)\ \&\ (Q(x)|R(x))-->S(x))\ \&$
$(\forall x.\ S(x)\ \&\ R(x)\ --> L(x))\ \&$
$(\forall x.\ M(x)\ --> R(x))$
$--> (\forall x.\ P(x)\ \&\ M(x)\ --> L(x))$

⟨*proof*⟩

33

lemma $(\forall x.\ P(a)\ \&\ (P(x)-->P(b))-->P(c)) \ <->$
$(\forall x.\ ({\sim}P(a) \mid P(x) \mid P(c))\ \&\ ({\sim}P(a) \mid {\sim}P(b) \mid P(c)))$

⟨*proof*⟩

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

lemma $((\exists x.\ \forall y.\ p(x) <-> p(y)) \ <->$
$((\exists x.\ q(x)) <-> (\forall y.\ p(y)))) \quad <->$
$((\exists x.\ \forall y.\ q(x) <-> q(y)) \ <->$
$((\exists x.\ p(x)) <-> (\forall y.\ q(y))))$

⟨*proof*⟩

35

lemma $\exists x\ y.\ P(x,y)\ --> \ (\forall u\ v.\ P(u,v))$

⟨*proof*⟩

36

lemma $(\forall x.\ \exists y.\ J(x,y))\ \&$
$(\forall x.\ \exists y.\ G(x,y))\ \&$
$(\forall x\ y.\ J(x,y) \mid G(x,y)\ --> (\forall z.\ J(y,z) \mid G(y,z)\ --> H(x,z)))$
$--> (\forall x.\ \exists y.\ H(x,y))$

⟨*proof*⟩

37

lemma $(\forall z.\ \exists w.\ \forall x.\ \exists y.$
$(P(x,z) \dashrightarrow P(y,w))\ \&\ P(y,z)\ \&\ (P(y,w) \dashrightarrow (\exists u.\ Q(u,w)))) \ \&$
$(\forall x\ z.\ \sim P(x,z) \dashrightarrow (\exists y.\ Q(y,z)))\ \&$
$((\exists x\ y.\ Q(x,y)) \dashrightarrow (\forall x.\ R(x,x)))$
$\dashrightarrow (\forall x.\ \exists y.\ R(x,y))$
⟨*proof*⟩

38

lemma $(\forall x.\ p(a)\ \&\ (p(x) \dashrightarrow (\exists y.\ p(y)\ \&\ r(x,y))) \dashrightarrow$
$(\exists z.\ \exists w.\ p(z)\ \&\ r(x,w)\ \&\ r(w,z)))\ <\!\!\dashrightarrow$
$(\forall x.\ (\sim p(a) \mid p(x) \mid (\exists z.\ \exists w.\ p(z)\ \&\ r(x,w)\ \&\ r(w,z)))\ \&$
$(\sim p(a) \mid \sim(\exists y.\ p(y)\ \&\ r(x,y)) \mid$
$(\exists z.\ \exists w.\ p(z)\ \&\ r(x,w)\ \&\ r(w,z))))$
⟨*proof*⟩

39

lemma $\sim (\exists x.\ \forall y.\ F(y,x) <\!\!\dashrightarrow \sim F(y,y))$
⟨*proof*⟩

40. AMENDED

lemma $(\exists y.\ \forall x.\ F(x,y) <\!\!\dashrightarrow F(x,x)) \dashrightarrow$
$\sim(\forall x.\ \exists y.\ \forall z.\ F(z,y) <\!\!\dashrightarrow \sim F(z,x))$
⟨*proof*⟩

41

lemma $(\forall z.\ \exists y.\ \forall x.\ f(x,y) <\!\!\dashrightarrow f(x,z)\ \&\ \sim f(x,x))$
$\dashrightarrow \sim (\exists z.\ \forall x.\ f(x,z))$
⟨*proof*⟩

42

lemma $\sim (\exists y.\ \forall x.\ p(x,y) <\!\!\dashrightarrow \sim (\exists z.\ p(x,z)\ \&\ p(z,x)))$
⟨*proof*⟩

43

lemma $(\forall x.\ \forall y.\ q(x,y) <\!\!\dashrightarrow (\forall z.\ p(z,x) <\!\!\dashrightarrow p(z,y)))$
$\dashrightarrow (\forall x.\ \forall y.\ q(x,y) <\!\!\dashrightarrow q(y,x))$
⟨*proof*⟩

44

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

45

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

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

46

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

6.5 Problems (mainly) involving equality or functions

48

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

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

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

51

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

52

Almost the same as 51.

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

55

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

lemma $lives(agatha)$ & $lives(butler)$ & $lives(charles)$ &
$(killed(agatha,agatha) \mid killed(butler,agatha) \mid killed(charles,agatha))$ &
$(\forall x\ y.\ killed(x,y) \longrightarrow hates(x,y) \ \&\ \sim richer(x,y))$ &
$(\forall x.\ hates(agatha,x) \longrightarrow \sim hates(charles,x))$ &
$(hates(agatha,agatha) \ \&\ hates(agatha,charles))$ &
$(\forall x.\ lives(x) \ \&\ \sim richer(x,agatha) \longrightarrow hates(butler,x))$ &
$(\forall x.\ hates(agatha,x) \longrightarrow hates(butler,x))$ &
$(\forall x.\ \sim hates(x,agatha) \mid \sim hates(x,butler) \mid \sim hates(x,charles)) \longrightarrow$
$killed(?who,agatha)$
⟨*proof*⟩

56

lemma $(\forall x.\ (\exists y.\ P(y) \ \&\ x{=}f(y)) \longrightarrow P(x)) \longleftrightarrow (\forall x.\ P(x) \longrightarrow P(f(x)))$
⟨*proof*⟩

57

lemma $P(f(a,b),\ f(b,c)) \ \&\ P(f(b,c),\ f(a,c)) \ \&$
$(\forall x\ y\ z.\ P(x,y) \ \&\ P(y,z) \longrightarrow P(x,z)) \longrightarrow P(f(a,b),\ f(a,c))$
⟨*proof*⟩

58 NOT PROVED AUTOMATICALLY

lemma $(\forall x\ y.\ f(x){=}g(y)) \longrightarrow (\forall x\ y.\ f(f(x)){=}f(g(y)))$
⟨*proof*⟩

59

lemma $(\forall x.\ P(x) \longleftrightarrow \sim P(f(x))) \longrightarrow (\exists x.\ P(x) \ \&\ \sim P(f(x)))$
⟨*proof*⟩

60

lemma $\forall x.\ P(x,f(x)) \longleftrightarrow (\exists y.\ (\forall z.\ P(z,y) \longrightarrow P(z,f(x))) \ \&\ P(x,y))$
⟨*proof*⟩

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

lemma $(\forall x.\ p(a) \ \&\ (p(x) \longrightarrow p(f(x))) \longrightarrow p(f(f(x)))) \longleftrightarrow$
$(\forall x.\ (\sim p(a) \mid p(x) \mid p(f(f(x)))) \ \&$
$(\sim p(a) \mid \sim p(f(x)) \mid p(f(f(x)))))$
⟨*proof*⟩

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

lemma $(\forall x.\ F(x) \ \&\ \sim G(x) \longrightarrow (\exists y.\ H(x,y) \ \&\ J(y))) \ \&$
$(\exists x.\ K(x) \ \&\ F(x) \ \&\ (\forall y.\ H(x,y) \longrightarrow K(y))) \ \&$
$(\forall x.\ K(x) \longrightarrow \sim G(x)) \longrightarrow (\exists x.\ K(x) \ \&\ J(x))$
⟨*proof*⟩

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

lemma $(\forall x.\ F(x)\ \&\ \sim G(x)\ \text{--->}\ (\exists y.\ H(x,y)\ \&\ J(y)))\ \&$
$\quad(\exists x.\ K(x)\ \&\ F(x)\ \&\ (\forall y.\ H(x,y)\ \text{--->}\ K(y)))\ \&$
$\quad(\forall x.\ K(x)\ \text{--->}\ \sim G(x))\ \ \text{--->}\ \ (\exists x.\ K(x)\ \text{--->}\ \sim G(x))$
⟨*proof*⟩

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

lemma $((\exists x.\ A(x)\ \&\ (\forall y.\ C(y)\ \text{--->}\ (\forall z.\ D(x,y,z))))\ \text{--->}$
$\quad(\exists w.\ C(w)\ \&\ (\forall y.\ C(y)\ \text{--->}\ (\forall z.\ D(w,y,z)))))$
$\quad\&$
$\quad(\forall w.\ C(w)\ \&\ (\forall u.\ C(u)\ \text{--->}\ (\forall v.\ D(w,u,v)))\ \text{--->}$
$\qquad(\forall y\ z.$
$\qquad\quad(C(y)\ \&\ \ P(y,z)\ \text{--->}\ Q(w,y,z)\ \&\ OO(w,g))\ \&$
$\qquad\quad(C(y)\ \&\ \sim P(y,z)\ \text{--->}\ Q(w,y,z)\ \&\ OO(w,b))))$
$\quad\&$
$\quad(\forall w.\ C(w)\ \&$
$\qquad(\forall y\ z.$
$\qquad\quad(C(y)\ \&\ P(y,z)\ \text{--->}\ Q(w,y,z)\ \&\ OO(w,g))\ \&$
$\qquad\quad(C(y)\ \&\ \sim P(y,z)\ \text{--->}\ Q(w,y,z)\ \&\ OO(w,b)))\ \text{--->}$
$\qquad(\exists v.\ C(v)\ \&$
$\qquad\quad(\forall y.\ ((C(y)\ \&\ Q(w,y,y))\ \&\ OO(w,g)\ \text{--->}\ \sim P(v,y))\ \&$
$\qquad\qquad((C(y)\ \&\ Q(w,y,y))\ \&\ OO(w,b)\ \text{--->}\ P(v,y)\ \&\ OO(v,b)))))$
$\quad\text{--->}$
$\quad\sim\ (\exists x.\ A(x)\ \&\ (\forall y.\ C(y)\ \text{--->}\ (\forall z.\ D(x,y,z))))$
⟨*proof*⟩

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

lemma $((\exists x.\ A(x)\ \&\ (\forall y.\ C(y)\ \text{--->}\ (\forall z.\ D(x,y,z))))\ \text{--->}$
$\quad(\exists w.\ C(w)\ \&\ (\forall y.\ C(y)\ \text{--->}\ (\forall z.\ D(w,y,z)))))$
$\quad\&$
$\quad(\forall w.\ C(w)\ \&\ (\forall u.\ C(u)\ \text{--->}\ (\forall v.\ D(w,u,v)))\ \text{--->}$
$\qquad(\forall y\ z.$
$\qquad\quad(C(y)\ \&\ \ P(y,z)\ \text{--->}\ Q(w,y,z)\ \&\ OO(w,g))\ \&$
$\qquad\quad(C(y)\ \&\ \sim P(y,z)\ \text{--->}\ Q(w,y,z)\ \&\ OO(w,b))))$
$\quad\&$
$\quad((\exists w.\ C(w)\ \&\ (\forall y.\ (C(y)\ \&\ \ P(y,y)\ \text{--->}\ Q(w,y,y)\ \&\ OO(w,g))\ \&$
$\qquad\qquad(C(y)\ \&\ \sim P(y,y)\ \text{--->}\ Q(w,y,y)\ \&\ OO(w,b))))$
$\quad\text{--->}$
$\quad(\exists v.\ C(v)\ \&\ (\forall y.\ (C(y)\ \&\ \ P(y,y)\ \text{--->}\ P(v,y)\ \&\ OO(v,g))\ \&$
$\qquad\qquad(C(y)\ \&\ \sim P(y,y)\ \text{--->}\ P(v,y)\ \&\ OO(v,b)))))$
$\quad\text{--->}$
$\quad((\exists v.\ C(v)\ \&\ (\forall y.\ (C(y)\ \&\ \ P(y,y)\ \text{--->}\ P(v,y)\ \&\ OO(v,g))\ \&$
$\qquad\qquad(C(y)\ \&\ \sim P(y,y)\ \text{--->}\ P(v,y)\ \&\ OO(v,b))))$
$\quad\text{--->}$
$\quad(\exists u.\ C(u)\ \&\ (\forall y.\ (C(y)\ \&\ \ P(y,y)\ \text{--->}\ \sim P(u,y))\ \&$
$\qquad\qquad(C(y)\ \&\ \sim P(y,y)\ \text{--->}\ P(u,y)\ \&\ OO(u,b)))))$
$\quad\text{--->}$
$\quad\sim\ (\exists x.\ A(x)\ \&\ (\forall y.\ C(y)\ \text{--->}\ (\forall z.\ D(x,y,z))))$
⟨*proof*⟩

Challenge found on info-hol

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

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

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

end

7 First-Order Logic: the 'if' example

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

constdefs
$if :: [o,o,o]\text{=>}o$
$if(P,Q,R) == P\&Q \mid \sim P\&R$

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

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

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

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

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

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

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

⟨*ML*⟩

end

8 Theory for examples of simplification and induction on the natural numbers

theory *Nat2*
imports *FOL*
begin

typedecl *nat*
arities *nat* :: *term*

consts
succ :: *nat* => *nat*
pred :: *nat* => *nat*
0 :: *nat* (*0*)
add :: [*nat*,*nat*] => *nat* (**infixr** + *90*)
lt :: [*nat*,*nat*] => *o* (**infixr** < *70*)
leq :: [*nat*,*nat*] => *o* (**infixr** <= *70*)

axioms
pred-0: *pred*(*0*) = *0*
pred-succ: *pred*(*succ*(*m*)) = *m*

plus-0: *0*+*n* = *n*
plus-succ: *succ*(*m*)+*n* = *succ*(*m*+*n*)

nat-distinct1: ~ *0*=*succ*(*n*)
nat-distinct2: ~ *succ*(*m*)=*0*
succ-inject: *succ*(*m*)=*succ*(*n*) <-> *m*=*n*

leq-0: *0* <= *n*
leq-succ-succ: *succ*(*m*)<=*succ*(*n*) <-> *m*<=*n*
leq-succ-0: ~ *succ*(*m*) <= *0*

lt-0-succ: *0* < *succ*(*n*)
lt-succ-succ: *succ*(*m*)<*succ*(*n*) <-> *m*<*n*
lt-0: ~ *m* < *0*

nat-ind: [| *P*(*0*); *ALL n*. *P*(*n*)-->*P*(*succ*(*n*)) |] ==> *All*(*P*)

⟨*ML*⟩

end

9 Examples of simplification and induction on lists

theory *List*
imports *Nat2*

begin

typedecl *'a list*
arities *list* :: (*term*) *term*

consts
hd :: *'a list => 'a*
tl :: *'a list => 'a list*
forall :: *['a list, 'a => o] => o*
len :: *'a list => nat*
at :: *['a list, nat] => 'a*
Nil :: *'a list* ([])
Cons :: *['a, 'a list] => 'a list* (**infixr** . *80*)
app :: *['a list, 'a list] => 'a list* (**infixr** ++ *70*)

axioms
list-ind: *[| P([]); ALL x l. P(l)-->P(x . l) |] ==> All(P)*

forall-cong:
[| l = l'; !!x. P(x)<->P'(x) |] ==> forall(l,P) <-> forall(l',P')

list-distinct1: *~[] = x . l*
list-distinct2: *~x . l = []*

list-free: *x . l = x' . l' <-> x=x' & l=l'*

app-nil: *[]++l = l*
app-cons: *(x . l)++l' = x .(l++l')*
tl-eq: *tl(m . q) = q*
hd-eq: *hd(m . q) = m*

forall-nil: *forall([],P)*
forall-cons: *forall(x . l,P) <-> P(x) & forall(l,P)*

len-nil: *len([]) = 0*
len-cons: *len(m . q) = succ(len(q))*

at-0: *at(m . q,0) = m*
at-succ: *at(m . q,succ(n)) = at(q,n)*

⟨*ML*⟩

end

10 Example of Declaring an Oracle

theory *IffOracle*
imports *FOL*

begin

10.1 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*⟩

10.2 Oracle as low-level rule

⟨*ML*⟩

These oracle calls had better fail

⟨*ML*⟩

10.3 Oracle as proof method

⟨*ML*⟩

lemma $A <-> A$
⟨*proof*⟩

lemma $A <-> A <-> A <-> A <-> A <-> A <-> A <-> A <-> A <-> A$
⟨*proof*⟩

lemma $A <-> A <-> A <-> A <-> A$
⟨*proof*⟩

lemma A
⟨*proof*⟩

end

11 Test of Locale Interpretation

theory *LocaleTest*
imports *FOL*
begin

⟨*ML*⟩

12 Context Elements and Locale Expressions

Naming convention for global objects: prefixes L and l

12.1 Renaming with Syntax

locale (**open**) *LS* = *var mult* +
  assumes *mult*(*x*, *y*) = *mult*(*y*, *x*)

print-locale *LS*

locale *LS′* = *LS mult* (**infixl** ∗∗ *60*)

print-locale *LS′*

locale *LT* = *var mult* (**infixl** ∗∗ *60*) +
  assumes *x* ∗∗ *y* = *y* ∗∗ *x*

locale *LU* = *LT mult* (**infixl** ∗∗ *60*) + *LT add* (**infixl** ++ *55*) + *var h* +
  assumes *hom*: *h*(*x* ∗∗ *y*) = *h*(*x*) ++ *h*(*y*)

locale *LV* = *LU* - *add*

12.2 Constrains

locale *LZ* = **fixes** *a* (**structure**)
locale *LZ′* = *LZ* +
  constrains *a* :: *′a* => *′b*
  assumes *a* (*x* :: *′a*) = *a* (*y*)
print-locale *LZ′*

13 Interpretation

Naming convention for global objects: prefixes I and i

interpretation input syntax

locale *IL*
locale *IM* = **fixes** *a* **and** *b* **and** *c*

interpretation *test* [*simp*]: *IL* + *IM a b c* [*x y z*] ⟨*proof*⟩

print-interps *IL*
print-interps *IM*

interpretation *test* [*simp*]: *IL* **print-interps** *IM* ⟨*proof*⟩

interpretation *IL* ⟨*proof*⟩

Processing of locale expression

locale *IA* = **fixes** *a* **assumes** *asm-A*: *a* = *a*

locale (**open**) *IB* = **fixes** *b* **assumes** *asm-B* [*simp*]: *b* = *b*

locale $IC = IA + IB$ **+ assumes** *asm-C*: $c = c$

locale $ID = IA + IB$ **+ fixes** d **defines** *def-D*: $d == (a = b)$

theorem (**in** *IA*)
includes *ID*
shows *True* ⟨*proof*⟩

theorem (**in** *ID*) *True* ⟨*proof*⟩

typedecl i
arities $i :: term$

interpretation *i1*: *IC* [X::i Y::i] ⟨*proof*⟩

print-interps *IA*

thm *i1.a.asm-A* **thm** *LocaleTest.i1.a.asm-A*
thm *i1.asm-A* **thm** *LocaleTest.i1.asm-A*

⟨*ML*⟩

interpretation *IC* [W::i Z::i] ⟨*proof*⟩
interpretation *IC* [W::$'a$ Z::i] ⟨*proof*⟩

print-interps *IA*

thm *asm-C* **thm** *a-b.asm-C* **thm** *LocaleTest.a-b.asm-C* **thm** *LocaleTest.a-b.asm-C*

⟨*ML*⟩

interpretation *i2*: *ID* [X Y::i $Y = X$] ⟨*proof*⟩

print-interps *IA*
print-interps *ID*

interpretation *i3*: *ID* [X Y::i] ⟨*proof*⟩

print-interps *IA*
print-interps *IB*
print-interps *IC*
print-interps *ID*

interpretation *i10*: *ID* + *ID* *a′* *b′* *d′* [*X* *Y*::*i* - *u* *v*::*i* -] ⟨*proof*⟩

corollary (**in** *ID*) *th-x*: *True* ⟨*proof*⟩

thm *i2.th-x* **thm** *i3.th-x*

⟨*ML*⟩

lemma (**in** *ID*) *th-y*: *d* == (*a* = *b*) ⟨*proof*⟩

thm *i2.th-y* **thm** *i3.th-y*

⟨*ML*⟩

lemmas (**in** *ID*) *th-z* = *th-y*

thm *i2.th-z*

⟨*ML*⟩

13.1 Interpretation in Proof Contexts

locale *IF* = **fixes** *f* **assumes** *asm-F*: *f* & *f* −−> *f*

theorem *True*
⟨*proof*⟩
  print-interps *IA*
  ⟨*proof*⟩
  print-interps *IA*
  print-interps *IC*
  ⟨*proof*⟩
  thm *i11.asm-F*
⟨*proof*⟩

theorem (**in** *IA*) *True*
⟨*proof*⟩

print-interps *IA*
⟨*proof*⟩

⟨*ML*⟩

locale *IE* = **fixes** *e* **defines** *e-def*: *e*(*x*) == *x* & *x*
notes *e-def2* = *e-def*

lemma (**in** *IE*) *True* **thm** *e-def* ⟨*proof*⟩

interpretation *i7*: *IE* [%*x*. *x*] ⟨*proof*⟩

thm *i7.e-def2*

⟨*ML*⟩

locale *IE'* = **fixes** *e* **defines** *e-def*: *e* == (%*x*. *x* & *x*)
notes *e-def2* = *e-def*

interpretation *i7'*: *IE'* [(%*x*. *x*)] ⟨*proof*⟩

thm *i7'.e-def2*

⟨*ML*⟩

locale (**open**) *IG* = **fixes** *g* **assumes** *asm-G*: *g* −−> *x*
notes *asm-G2* = *asm-G*

interpretation *i8*: *IG* [*False*] ⟨*proof*⟩

thm *i8.asm-G2*

⟨*ML*⟩

Locale without assumptions

locale *IL1* = **notes** *rev-conjI* [*intro*] = *conjI* [*THEN iffD1* [*OF conj-commute*]]

lemma [| *P*; *Q* |] ==> *P* & *Q*
⟨*proof*⟩

locale *IL11* = **notes** *rev-conjI* = *conjI* [*THEN iffD1* [*OF conj-commute*]]

lemma [| *P*; *Q* |] ==> *P* & *Q*
⟨*proof*⟩

13.2 Simple locale with assumptions

consts *ibin* :: [*i*, *i*] => *i* (**infixl** # *60*)

axioms *i-assoc*: (*x* # *y*) # *z* = *x* # (*y* # *z*)
i-comm: *x* # *y* = *y* # *x*

locale *IL2* =
fixes *OP* (**infixl** + *60*)
assumes *assoc*: (*x* + *y*) + *z* = *x* + (*y* + *z*)
and *comm*: *x* + *y* = *y* + *x*

lemma (**in** *IL2*) *lcomm*: *x* + (*y* + *z*) = *y* + (*x* + *z*)
⟨*proof*⟩

lemmas (**in** *IL2*) *AC* = *comm assoc lcomm*

lemma (*x*::*i*) # *y* # *z* # *w* = *y* # *x* # *w* # *z*
⟨*proof*⟩

13.3 Nested locale with assumptions

locale *IL3* =
fixes *OP* (**infixl** + *60*)
assumes *assoc*: (*x* + *y*) + *z* = *x* + (*y* + *z*)

locale *IL4* = *IL3* +
assumes *comm*: *x* + *y* = *y* + *x*

lemma (**in** *IL4*) *lcomm*: *x* + (*y* + *z*) = *y* + (*x* + *z*)
⟨*proof*⟩

lemmas (**in** *IL4*) *AC* = *comm assoc lcomm*

lemma (*x*::*i*) # *y* # *z* # *w* = *y* # *x* # *w* # *z*
⟨*proof*⟩

Locale with definition

This example is admittedly not very creative :-)

locale *IL5* = *IL4* + *var A* +
defines *A-def*: *A* == *True*

lemma (**in** *IL5*) *lem*: *A*
⟨*proof*⟩

lemma *IL5*(*op* #) ==> *True*
⟨*proof*⟩

Interpretation in a context with target

lemma (**in** *IL4*)
fixes *A* (**infixl** $ *60*)
assumes *A*: *IL4*(*A*)
shows (*x*::*i*) $ *y* $ *z* $ *w* = *y* $ *x* $ *w* $ *z*
⟨*proof*⟩

14 Interpretation in Locales

Naming convention for global objects: prefixes R and r

locale (**open**) *Rsemi* = *var prod* (**infixl** ** *65*) +
assumes *assoc*: (*x* ** *y*) ** *z* = *x* ** (*y* ** *z*)

locale (**open**) *Rlgrp* = *Rsemi* + *var one* + *var inv* +
assumes *lone*: *one* ** *x* = *x*
and *linv*: *inv*(*x*) ** *x* = *one*

lemma (**in** *Rlgrp*) *lcancel*:
x ** *y* = *x* ** *z* <-> *y* = *z*
⟨*proof*⟩

locale (**open**) *Rrgrp* = *Rsemi* + *var one* + *var inv* +
assumes *rone*: *x* ** *one* = *x*
and *rinv*: *x* ** *inv*(*x*) = *one*

lemma (**in** *Rrgrp*) *rcancel*:
y ** *x* = *z* ** *x* <-> *y* = *z*
⟨*proof*⟩

interpretation *Rlgrp* < *Rrgrp*
⟨*proof*⟩

print-locale *Rlgrp*

lemma (**in** *Rlgrp*)
y ** *x* = *z* ** *x* <-> *y* = *z*
⟨*proof*⟩
print-interps *Rrgrp* **thm** *lcancel rcancel*
⟨*proof*⟩

interpretation *Rrgrp* < *Rlgrp*
⟨*proof*⟩

print-locale! *Rrgrp*
print-locale! *Rlgrp*

locale *RA5* = *var A* + *var B* + *var C* + *var D* + *var E* +
  assumes *eq*: *A* <–> *B* <–> *C* <–> *D* <–> *E*

interpretation *RA5* < *RA5* - - *D E C*
print-facts
print-interps *RA5*
  ⟨*proof*⟩

interpretation *RA5* < *RA5 C* - *E* - *A*
print-facts
print-interps *RA5*
  ⟨*proof*⟩

interpretation *RA5* < *RA5 B C A* - -
print-facts
print-interps *RA5*
  ⟨*proof*⟩

lemma (**in** *RA5*) *True*
print-facts
print-interps *RA5*
  ⟨*proof*⟩

interpretation *RA5* < *RA5* - *C D B* - ⟨*proof*⟩

locale (**open**) *RA1* = *var A* + *var B* + **assumes** *p*: *A* <–> *B*
locale (**open**) *RA2* = *var A* + *var B* + **assumes** *q*: *A* & *B* | ~ *A* & ~ *B*
locale (**open**) *RA3* = *var A* + *var B* + **assumes** *r*: (*A* –-> *B*) & (*B* –-> *A*)

interpretation *RA1* < *RA2*
  print-facts
  ⟨*proof*⟩
interpretation *RA2* < *RA3*
  print-facts
  ⟨*proof*⟩
interpretation *RA3* < *RA1*
  print-facts
  ⟨*proof*⟩

locale (**open**) *RB1* = *var A* + *var B* + **assumes** *p*: *A* <−> *B*
locale (**open**) *RB2* = *var A* + *var B* + **assumes** *q*: *A* & *B* | ~ *A* & ~ *B*
locale (**open**) *RB3* = *var A* + *var B* + **assumes** *r*: (*A* −−> *B*) & (*B* −−> *A*)

interpretation *RB1* < *RB2*
print-facts
⟨*proof*⟩
interpretation *RB3* < *RB1*
print-facts
⟨*proof*⟩
interpretation *RB2* < *RB3*
print-facts
⟨*proof*⟩

lemma (**in** *RB1*) *True*
print-facts
⟨*proof*⟩

locale *Rpsemi* = *var prod* (**infixl** ** 65*) +
assumes *assoc*: (*x* ** *y*) ** *z* = *x* ** (*y* ** *z*)

locale *Rplgrp* = *Rpsemi* + *var one* + *var inv* +
assumes *lone*: *one* ** *x* = *x*
and *linv*: *inv*(*x*) ** *x* = *one*

lemma (**in** *Rplgrp*) *lcancel*:
x ** *y* = *x* ** *z* <−> *y* = *z*
⟨*proof*⟩

locale *Rprgrp* = *Rpsemi* + *var one* + *var inv* +
assumes *rone*: *x* ** *one* = *x*
and *rinv*: *x* ** *inv*(*x*) = *one*

lemma (**in** *Rprgrp*) *rcancel*:
y ** *x* = *z* ** *x* <−> *y* = *z*
⟨*proof*⟩

interpretation *Rplgrp* < *Rprgrp*
⟨*proof*⟩

print-locale *Rplgrp*

lemma (**in** *Rplgrp*)
*y ** x = z ** x <-> y = z*
⟨*proof*⟩
print-interps *Rprgrp* **thm** *lcancel rcancel*
⟨*proof*⟩

interpretation *Rprgrp* < *Rplgrp*
⟨*proof*⟩

print-locale *Rprgrp*
print-locale *Rplgrp*

14.1 Interaction of Interpretation in Theories and Locales: in Locale, then in Theory

consts
rone :: *i*
rinv :: *i* => *i*

axioms
r-one : *rone # x = x*
r-inv : *rinv*(*x*) *# x = rone*

interpretation *Rbool*: *Rlgrp* [*op # rone rinv*]
⟨*proof*⟩

print-interps *Rrgrp*
print-interps *Rlgrp*

lemma *y # x = z # x <-> y = z* ⟨*proof*⟩

lemma (**in** *Rrgrp*) *new-cancel*:
*b ** a = c ** a <-> b = c*
⟨*proof*⟩

thm *Rbool.new-cancel*

⟨*ML*⟩

lemma $b \# a = c \# a <-> b = c$ ⟨*proof*⟩

14.2 Interaction of Interpretation in Theories and Locales: in Theory, then in Locale

locale *Rqsemi* = *var prod* (**infixl** $**$ *65*) +
  assumes *assoc*: $(x ** y) ** z = x ** (y ** z)$

locale *Rqlgrp* = *Rqsemi* + *var one* + *var inv* +
  assumes *lone*: $one ** x = x$
    and *linv*: $inv(x) ** x = one$

lemma (**in** *Rqlgrp*) *lcancel*:
  $x ** y = x ** z <-> y = z$
⟨*proof*⟩

locale *Rqrgrp* = *Rqsemi* + *var one* + *var inv* +
  assumes *rone*: $x ** one = x$
    and *rinv*: $x ** inv(x) = one$

lemma (**in** *Rqrgrp*) *rcancel*:
  $y ** x = z ** x <-> y = z$
⟨*proof*⟩

interpretation *Rqrgrp* < *Rprgrp*
⟨*proof*⟩

interpretation *R2*: *Rqlgrp* [*op* # *rone rinv*]
⟨*proof*⟩

print-interps *Rqsemi*
print-interps *Rqlgrp*
print-interps *Rplgrp*

interpretation *Rqlgrp* < *Rqrgrp*
  ⟨*proof*⟩

print-interps! *Rqrgrp*
print-interps! *Rpsemi*
print-interps! *Rprgrp*
print-interps! *Rplgrp*
thm *R2.rcancel*
thm *R2.lcancel*

⟨*ML*⟩

14.3 Generation of Witness Theorems for Transitive Interpretations

locale *Rtriv* = *var x* +
  assumes *x*: *x* = *x*

locale *Rtriv2* = *var x* + *var y* +
  assumes *x*: *x* = *x* **and** *y*: *y* = *y*

interpretation *Rtriv2* < *Rtriv x*
  ⟨*proof*⟩

interpretation *Rtriv2* < *Rtriv y*
  ⟨*proof*⟩

print-locale *Rtriv2*

locale *Rtriv3* = *var x* + *var y* + *var z* +
  assumes *x*: *x* = *x* **and** *y*: *y* = *y* **and** *z*: *z* = *z*

interpretation *Rtriv3* < *Rtriv2 x y*
  ⟨*proof*⟩

print-locale *Rtriv3*

interpretation *Rtriv3* < *Rtriv2 x z*
  ⟨*proof*⟩

⟨*ML*⟩

print-locale *Rtriv3*

14.4 Normalisation Replaces Assumed Element by Derived Element

typedecl ('*a*, '*b*) *pair*
arities *pair* :: (*term*, *term*) *term*

consts
  pair :: ['*a*, '*b*] => ('*a*, '*b*) *pair*
  fst :: ('*a*, '*b*) *pair* => '*a*
  snd :: ('*a*, '*b*) *pair* => '*b*

axioms
  fst [*simp*]: *fst*(*pair*(*x*, *y*)) = *x*
  snd [*simp*]: *snd*(*pair*(*x*, *y*)) = *y*

locale *Rpair* = *var prod* (**infixl** ** 65) + *var prodP* (**infixl** *** 65) +
  defines *P-def*: *x* *** *y* == *pair*(*fst*(*x*) ** *fst*(*y*), *snd*(*x*) ** *snd*(*y*))

locale *Rpair-semi* = *Rpair* + *Rpsemi*

interpretation *Rpair-semi* < *Rpsemi prodP* (**infixl** ∗∗∗ *65*)
⟨*proof*⟩

locale *Rsemi-rev* = *Rpsemi* + *var rprod* (**infixl** ++ *65*) +
defines *r-def*: *x* ++ *y* == *y* ∗∗ *x*

lemma (**in** *Rsemi-rev*) *r-assoc*:
(*x* ++ *y*) ++ *z* = *x* ++ (*y* ++ *z*)
⟨*proof*⟩

lemma (**in** *Rpair-semi*)
includes *Rsemi-rev prodP* (**infixl** ∗∗∗ *65*) *rprodP* (**infixl** +++ *65*)
constrains *prod* :: ['*a*, '*a*] => '*a*
and *rprodP* :: [('*a*, '*a*) *pair*, ('*a*, '*a*) *pair*] => ('*a*, '*a*) *pair*
shows (*x* +++ *y*) +++ *z* = *x* +++ (*y* +++ *z*)
⟨*proof*⟩

end