Projekt

By Fabian Schütz, Ronja Deisel, Tobias Kloht, Lukas Grätz

September 17, 2016

Contents

theory *Rushby3*
imports *Main*
begin

typedecl u — Ubeings

consts $g :: u \Rightarrow u \Rightarrow bool$ (**infixr** $>$ *54*)
consts $k :: u \Rightarrow u \Rightarrow bool$ (**infixr** $<$ *54*)
consts $e :: u \Rightarrow u \Rightarrow bool$ (**infixr** $=$ *54*)

abbreviation *Greater0* **where**
$Greater0 \equiv \forall\ x\ y.\ x > y \vee y > x \vee x = y$

abbreviation *God* :: $u \Rightarrow bool$ (G) **where**
$G \equiv \lambda x.\ \neg(\exists\ y.\ (y > x))$

consts $re :: u \Rightarrow bool$

abbreviation *ExUnd* **where**
$ExUnd \equiv \exists\ x\ .\ God\ x$

abbreviation *Greater1* **where**
$Greater1 \equiv \forall\ x\ .\ (\neg re\ x) \longrightarrow (\exists\ y\ .\ y > x)$

theorem *God*!:
assumes *ExUnd*
assumes *Greater1*
shows $\exists\ x.\ (G\ x \wedge re\ x)$

using *assms* **by** *blast*

abbreviation *Greater2* **where**
$Greater2 \equiv (\forall\ x\ y.\ (re\ x \wedge \neg\ re\ y) \longrightarrow (x > y))$

abbreviation *Ex-re* **where**
$Ex\text{-}re \equiv \exists\ x.\ re\ x$

theorem *God!2*:
  assumes *ExUnd*
  assumes *Greater2* **and** *Ex-re*
  shows $\exists\ x.\ (G\ x \wedge re\ x)$

using *assms* **by** *blast*

abbreviation *P1* **where**
  $P1 \equiv \exists\ x.\ G\ x$

theorem *P1*!:
  assumes *ExUnd*
  shows *P1* **using** *assms* **by** −

consts $P :: (u \Rightarrow bool) \Rightarrow bool$ — greater making property

abbreviation *P-re* **where**
  $P\text{-}re \equiv P\ re$

abbreviation *subsetP* **where**
  $subsetP \equiv \lambda\ FF.\ \forall x.\ FF\ x \longrightarrow P\ x$

abbreviation *Greater3* **where**
  $Greater3 \equiv \forall\ x\ y\ .\ x > y \longleftrightarrow (\forall F.\ P\ F \longrightarrow (F\ y \longrightarrow F\ x)) \wedge (\exists F.\ P\ F \wedge (F\ x \wedge \neg F\ y))$

abbreviation *Realization* **where**
  $Realization \equiv \forall\ FF.\ subsetP\ FF \longrightarrow (\exists x.\ \forall f.\ P\ f \longrightarrow (f\ x \longleftrightarrow FF\ f))$

abbreviation *Allah* **where**
  $Allah \equiv \lambda x\ .\ \forall y.\ x > y$

abbreviation *ExUndAllah* **where**
  $ExUndAllah \equiv \exists x.\ Allah\ x$

theorem *Allah!3*:
  assumes *P-re*
  assumes *Greater3*
  assumes *Realization*
  assumes *ExUndAllah*
  shows $\exists\ x.\ (Allah\ x \wedge re\ x)$
proof −
    from *assms(4)* **have** $\exists x.\ Allah\ x$ **by** −
    moreover {
      fix *x*
      assume *gx*: *Allah x*
      then have $\forall\ y.\ (x > y)$ **by** −

```
    from this assms(2) have ∀y.(∀F. P F ⟶ (F y ⟶ F x)) ∧ (∃F. P F ∧
(F x ∧ ¬F y)) by blast
    from this assms(1) have rx: re x by blast
    from gx rx have Allah x ∧ re x by (rule conjI)
    hence ∃ x. (Allah x ∧ re x) by (rule exI)
  }
  ultimately show ?thesis by (rule exE)
qed

theorem AllahCanDoItAll:
 assumes P-re
 assumes Greater3
 assumes Realization
 assumes ExUndAllah
 shows ∀f. P f ⟶ (∃x. Allah x ∧ f x)
proof −
 {
  fix f
  assume P f
  from assms(4) have ∃x. Allah x by −
  moreover {
   fix x
   assume gx: Allah x
   from this assms(2) assms(3) have ∀ y. x > y by blast
    from this assms(2) have ∀y.(∀F. P F ⟶ (F y ⟶ F x)) ∧ (∃F. P F ∧
(F x ∧ ¬F y)) by blast
   from this assms(1) have rx: f x by blast
   from gx rx have Allah x ∧ f x by (rule conjI)
   hence ∃ x. (Allah x ∧ f x) by (rule exI)
  }
  ultimately have (∃x. Allah x ∧ f x) by (rule exE)
 }
 thus ?thesis by blast
qed


theorem God!3:
 assumes P-re:
     P-re
 assumes Greater3:
     Greater3
 assumes Realization:
     Realization
 assumes ExUnd:
     ExUnd
 shows ∃ x. (God x ∧ re x)
proof −
 {
  fix x
```

```
    assume 0: God x
    have re x proof (cases re x)
      assume re x
      thus ?thesis by -
    next
      assume cas: ¬(re x)
      have ∃ff. (∀ f. P f ⟶ (f x ⟶ ff f)) ∧ ff re by blast
      moreover {
        fix ff
        assume ass:(∀ f. P f ⟶ (f x ⟶ ff f)) ∧ ff re
        from this have 1: ∀ f. P f ⟶ (f x ⟶ ff f) by simp
        from ass have 3: ff re by simp
         from 0 1 Greater3 Realization have ∃ y. ∀ f. P f ⟶ (f y ⟷ ff f)  by
metis
        moreover {
          fix y
          assume 4: ∀ f. P f ⟶ (f y ⟷ ff f)
          from this 3 P-re have re y by blast
          from 4 0 have a:¬ (y > x) by blast
          from 1 4 have vor1: ∀ F. P F ⟶ (F x ⟶ F y) by blast
          from P-re 3 4 cas have P re ∧ re y ∧ ¬(re x) by blast
          hence vor2: ∃ F. P F ∧ (F y ∧ ¬F x) by blast
          from vor1 vor2 Greater3 have y > x by blast
          from this a have False by simp
        }
        ultimately have False by (rule exE)
      }
      ultimately have False by (rule exE)
      thus ?thesis by simp
    qed
    from 0 this have God x ∧ re x by (rule conjI)
    hence ∃ x. God x ∧ re x by (rule exI)
  }
  from ExUnd this show ?thesis by (rule exE)
qed


theorem God!3Ver:
  assumes P-re:
          P-re
  assumes Greater3:
          Greater3
  assumes Realization:
          Realization
  assumes ExUnd:
          ExUnd
  shows ∃ x. (God x ∧ re x)
proof -
  {
```

```
    fix x
    assume 0: God x
    {
      assume cas: ¬(re x)
      from Realization P-re have ∀ f. P f ⟶ (f x ⟶ P f) ∧  P re by simp
      from Realization have 1: ∀ f. P f ⟶ (f x ⟶ P f) by simp
      from Realization have ∃ y. ∀ f. P f ⟶ (f y ⟷ P f) by blast
      moreover {
        fix y
        assume ∀ f. P f ⟶ (f y ⟷ P f)
        hence 4: ∀ f. P f ⟶ f y by simp
        hence P re ⟶ re y by (rule allE)
        from this P-re have re:re y by (rule mp)
        from 4 0 have a:¬ (y > x) by simp
        from 4 have vor1: ∀ f. P f ⟶ (f x ⟶ f y) by blast
        from P-re re cas have P re ∧ re y ∧ ¬(re x) by simp
        hence vor2: ∃ F. (P F ∧ F y ∧ ¬(F x)) by blast
        from vor1 vor2 Greater3 have y > x by blast
        from this a have False by simp
      }
      ultimately have False by (rule exE)
    }
    hence re x by (rule ccontr)
    from 0 this have God x ∧ re x by (rule conjI)
    hence ∃ x. God x ∧ re x by (rule exI)
  }
  from ExUnd this show ?thesis by (rule exE)
qed

theorem GodCanDoItAll:
  assumes P-re
  assumes Greater3
  assumes Realization
  assumes ExUnd
  shows ∀ f. P f ⟶ (∃ x. God x ∧ f x)
proof −

oops


consts D :: (u ⇒ bool) ⇒ bool

axiomatization where
  dsubstP: ∀ f. D f ⟶ P f


abbreviation quasi-id where
  quasi-id ≡ λx y. ∀ f. P f ∧ ¬D f ⟶ f x = f y
```

abbreviation *Realization-W* **where**
$Realization\text{-}W \equiv \forall\ FF.\ subsetP\ FF \longrightarrow (\exists\, x.\ \forall\, f.\ FF\, f \longrightarrow f\, x)$

end