

Arrow and Gibbard-Satterthwaite

Tobias Nipkow

December 17, 2009

Abstract

This article formalizes two proofs of Arrow's impossibility theorem due to Geanakoplos and derives the Gibbard-Satterthwaite theorem as a corollary. One formalization is based on utility functions, the other one on strict partial orders.

For an article about these proofs see <http://www.in.tum.de/~nipkow/pubs/arrow.pdf>.

1 Arrow's Theorem for Utility Functions

theory *Arrow-Utility* **imports** *Complex-Main*
begin

This theory formalizes the first proof due to Geanakoplos [1]. In contrast to the standard model of preferences as linear orders, we model preferences as *utility functions* mapping each alternative to a real number. The type of alternatives and voters is assumed to be finite.

typedecl *alt*
typedecl *indi*

axioms
alt3: $\exists a b c :: alt. distinct[a,b,c]$
finite-alt: *finite*(UNIV :: *alt set*)

finite-indi: *finite*(UNIV :: *indi set*)

lemma *third-alt*: $a \neq b \implies \exists c :: alt. distinct[a,b,c]$
using *alt3* **by** *simpmetis*

lemma *alt2*: $\exists b :: alt. b \neq a$
using *alt3* **by** *simpmetis*

types *pref* = *alt* \implies *real*
pref = *indi* \implies *pref*

definition

top :: *pref* \Rightarrow *alt* \Rightarrow *bool* (**infixr** $<\cdot$ 60) **where**
p $<\cdot$ *b* $\equiv \forall a. a \neq b \longrightarrow p\ a < p\ b$

definition

bot :: *alt* \Rightarrow *pref* \Rightarrow *bool* (**infixr** $\cdot <$ 60) **where**
b $\cdot <$ *p* $\equiv \forall a. a \neq b \longrightarrow p\ b < p\ a$

definition

extreme :: *pref* \Rightarrow *alt* \Rightarrow *bool* **where**
extreme *p* *b* $\equiv b \cdot < p \vee p < \cdot b$

abbreviation

Extreme *P* *b* $\equiv \forall i. \text{extreme } (P\ i)\ b$

lemma [*simp*]: $r \leq s \Longrightarrow r < s+(1::\text{real})$

by *arith*

lemma [*simp*]: $r < s \Longrightarrow r < s+(1::\text{real})$

by *arith*

lemma [*simp*]: $r \leq s \Longrightarrow \neg s+(1::\text{real}) < r$

by *arith*

lemma [*simp*]: $(r < s-(1::\text{real})) = (r+1 < s)$

by *arith*

lemma [*simp*]: $(s-(1::\text{real}) < r) = (s < r+1)$

by *arith*

lemma *less-if-bot*[*simp*]: $\llbracket b \cdot < p; x \neq b \rrbracket \Longrightarrow p\ b < p\ x$

by(*simp* *add:bot-def*)

lemma [*simp*]: $\llbracket p < \cdot b; x \neq b \rrbracket \Longrightarrow p\ x < p\ b$

by(*simp* *add:top-def*)

lemma [*simp*]: **assumes** *top*: $p < \cdot b$ **shows** $\neg p\ b < p\ c$

proof (*cases*)

assume $b = c$ **thus** *?thesis* **by** *simp*

next

assume $b \neq c$

with *top* **have** $p\ c < p\ b$ **by** (*simp* *add:eq-sym-conv*)

thus *?thesis* **by** *simp*

qed

lemma *not-less-if-bot*[*simp*]:

assumes *bot*: $b \cdot < p$ **shows** $\neg p\ c < p\ b$

proof (*cases*)

assume $b = c$ **thus** *?thesis* **by** *simp*

next

assume $b \neq c$

with *bot* **have** $p\ b < p\ c$ **by** (*simp* *add:eq-sym-conv*)

thus *?thesis* **by** *simp*

qed

lemma *top-impl-not-bot*[simp]: $p < \cdot b \implies \neg b \cdot < p$
by(*unfold bot-def, simp add:alt2*)

lemma [simp]: *extreme* $p b \implies (\neg p < \cdot b) = (b \cdot < p)$
apply(*unfold extreme-def*)
apply(*fastsimp dest:top-impl-not-bot*)
done

lemma [simp]: *extreme* $p b \implies (\neg b \cdot < p) = (p < \cdot b)$
apply(*unfold extreme-def*)
apply(*fastsimp dest:top-impl-not-bot*)
done

Auxiliary construction to hide details of preference model.

definition

mktop :: $pref \Rightarrow alt \Rightarrow pref$ **where**
mktop $p b \equiv p(b := Max(range p) + 1)$

definition

mkbot :: $pref \Rightarrow alt \Rightarrow pref$ **where**
mkbot $p b \equiv p(b := Min(range p) - 1)$

definition

between :: $pref \Rightarrow alt \Rightarrow alt \Rightarrow alt \Rightarrow pref$ **where**
between $p a b c \equiv p(b := (p a + p c)/2)$

To make things simpler:

declare *between-def*[simp]

lemma [simp]: $a \neq b \implies mktop p b a = p a$
by(*simp add:mktop-def*)

lemma [simp]: $a \neq b \implies mkbot p b a = p a$
by(*simp add:mkbot-def*)

lemma [simp]: $a \neq b \implies p a < mktop p b b$
by(*simp add:mktop-def finite-alt*)

lemma [simp]: $a \neq b \implies mkbot p b b < p a$
by(*simp add:mkbot-def finite-alt*)

lemma [simp]: $mktop p b < \cdot b$
by(*simp add:mktop-def top-def finite-alt*)

lemma [simp]: $\neg b \cdot < mktop p b$
by(*simp add:mktop-def bot-def alt2 finite-alt*)

lemma [simp]: $a \neq b \implies \neg P p a < mkb\text{ot } (P p) b b$
proof (simp add:mkb\text{ot-def finite-alt)
 have $\neg P p a + 1 < P p a$ by simp
 thus EX $x. \neg P p a + 1 < P p x$..
qed

The proof starts here.

locale arrow =
fixes $F :: \text{prof} \Rightarrow \text{pref}$
assumes unanimity: $(\bigwedge i. P i a < P i b) \implies F P a < F P b$
and IIA:
 $(\bigwedge i. (P i a < P i b) = (P' i a < P' i b)) \implies$
 $(F P a < F P b) = (F P' a < F P' b)$
begin

lemmas IIA' = IIA[THEN iffD1]

definition

$\text{dictates} :: \text{indi} \Rightarrow \text{alt} \Rightarrow \text{alt} \Rightarrow \text{bool } (- \text{dictates } - < -)$ **where**
 $(i \text{dictates } a < b) \equiv \forall P. P i a < P i b \longrightarrow F P a < F P b$

definition

$\text{dictates2} :: \text{indi} \Rightarrow \text{alt} \Rightarrow \text{alt} \Rightarrow \text{bool } (- \text{dictates } -, -)$ **where**
 $(i \text{dictates } a, b) \equiv (i \text{dictates } a < b) \wedge (i \text{dictates } b < a)$

definition

$\text{dictates}' :: \text{indi} \Rightarrow \text{alt} \Rightarrow \text{bool } (- \text{dictates}'\text{-except } -)$ **where**
 $(i \text{dictates}'\text{-except } c) \equiv \forall a b. c \notin \{a, b\} \longrightarrow (i \text{dictates } a < b)$

definition

$\text{dictator} :: \text{indi} \Rightarrow \text{bool}$ **where**
 $\text{dictator } i \equiv \forall a b. (i \text{dictates } a < b)$

definition

$\text{pivotal} :: \text{indi} \Rightarrow \text{alt} \Rightarrow \text{bool}$ **where**
 $\text{pivotal } i b \equiv$
 $\exists P. \text{Extreme } P b \wedge b \cdot < P i \wedge b \cdot < F P \wedge$
 $F (P(i := mktop (P i) b)) < b$

lemma all-top[simp]: $\forall i. P i < b \implies F P < b$
by (unfold top-def) (simp add: unanimity)

lemma not-extreme:

assumes nex: $\neg \text{extreme } p b$
shows $\exists a c. \text{distinct}[a, b, c] \wedge \neg p a < p b \wedge \neg p b < p c$
proof –
obtain $a c$ **where** $abc: a \neq b \wedge \neg p a < p b \wedge b \neq c \wedge \neg p b < p c$
using nex **by** (unfold extreme-def top-def bot-def) fastsimp
show ?thesis
proof (cases $a = c$)
 assume $a \neq c$ **thus** ?thesis **using** abc **by** simp blast
next

```

assume ac:  $a = c$ 
obtain d where  $d$ : distinct[ $a, b, d$ ] using abc third-alt by blast
show ?thesis
proof (cases  $p\ b < p\ d$ )
  case False thus ?thesis using abc d by blast
next
  case True
  hence db:  $\neg p\ d < p\ b$  by arith
  from d have distinct[ $d, b, c$ ] by(simp add:ac eq-sym-conv)
  thus ?thesis using abc db by blast
qed
qed
qed

```

lemma *extremal*:

```

assumes extremes: Extreme P b shows extreme (F P) b
proof (rule ccontr)
assume nec:  $\neg$  extreme (F P) b
hence  $\exists a\ c.$  distinct[ $a, b, c$ ]  $\wedge \neg F\ P\ a < F\ P\ b \wedge \neg F\ P\ b < F\ P\ c$ 
  by(rule not-extreme)
then obtain a c where  $d$ : distinct[ $a, b, c$ ] and
  ab:  $\neg F\ P\ a < F\ P\ b$  and bc:  $\neg F\ P\ b < F\ P\ c$  by blast
let ?P =  $\lambda i.$  if  $P\ i < \cdot b$  then between (P i) a c b
  else  $(P\ i)(c := P\ i\ a + 1)$ 
have  $\neg F\ ?P\ a < F\ ?P\ b$ 
  using extremes d by(simp add:IIA[of - - - P] ab)
moreover have  $\neg F\ ?P\ b < F\ ?P\ c$ 
  using extremes d by(simp add:IIA[of - - - P] bc eq-sym-conv)
moreover have  $F\ ?P\ a < F\ ?P\ c$  by(rule unanimity)(insert d, simp)
ultimately show False by arith
qed

```

lemma *pivotal-ind*: **assumes** *fin*: *finite D*

```

shows  $\bigwedge P.$   $\llbracket D = \{i. b \cdot < P\ i\}; \text{Extreme } P\ b; b \cdot < F\ P \rrbracket$ 
   $\implies \exists i.$  pivotal i b (is  $\bigwedge P.$   $?D\ D\ P \implies ?E\ P \implies ?B\ P \implies -$ )
using fin
proof (induct)
  case (empty P)
  from empty(1,2) have  $\forall i.$   $P\ i < \cdot b$  by simp
  hence  $F\ P < \cdot b$  by simp
  hence False using empty by(blast dest:top-impl-not-bot)
  thus ?case ..
next
  fix  $D\ i\ P$ 
  assume IH:  $\bigwedge P.$   $?D\ D\ P \implies ?E\ P \implies ?B\ P \implies \exists i.$  pivotal i b
  and  $?E\ P$  and  $?B\ P$  and insert: insert i D = {i. b · < P i} and  $i \notin D$ 
  from insert have  $b \cdot < P\ i$  by blast
  let  $?P = P(i := \text{mktop } (P\ i)\ b)$ 

```

```

show  $\exists i. \text{pivotal } i \ b$ 
proof (cases  $F \ ?P < \cdot \ b$ )
  case True
  have  $\text{pivotal } i \ b$ 
  proof -
    from  $\langle ?E \ P \rangle \langle ?B \ P \rangle \langle b \cdot < P \ i \rangle \ \text{True}$ 
    show  $?thesis$  by (unfold  $\text{pivotal-def}$ , blast)
  qed
  thus  $?thesis \ ..$ 
next
case False
have  $D = \{i. b \cdot < ?P \ i\}$ 
  by (rule  $\text{set-ext}$ ) (simp add:  $\langle i \notin D \rangle$ , insert insert, blast)
moreover have  $\text{Extreme } ?P \ b$ 
  using  $\langle ?E \ P \rangle$  by (simp add:  $\text{extreme-def}$ )
moreover have  $b \cdot < F \ ?P$ 
  using  $\text{extremal}[OF \ \langle \text{Extreme } ?P \ b \rangle] \ \text{False}$  by (simp del:  $\text{fun-upd-apply}$ )
ultimately show  $?thesis$  by (rule  $\text{IH}$ )
qed
qed

```

```

lemma  $\text{pivotal-exists}: \exists i. \text{pivotal } i \ b$ 
proof -
  let  $?P = (\lambda a. \text{if } a=b \ \text{then } 0 \ \text{else } 1)::\text{prof}$ 
  have  $\text{Extreme } ?P \ b$  by (simp add:  $\text{extreme-def bot-def}$ )
  moreover have  $b \cdot < F \ ?P$ 
    by (simp add:  $\text{bot-def unanimity del: less-if-bot not-less-if-bot}$ )
  ultimately show  $\exists i. \text{pivotal } i \ b$ 
    by (rule  $\text{pivotal-ind}[OF \ \text{finite-subset}[OF \ \text{subset-UNIV finite-indi}] \ \text{refl}]$ )
qed

```

```

lemma  $\text{pivotal-xdictates}: \text{assumes } \text{pivo}: \text{pivotal } i \ b$ 
  shows  $i \ \text{dictates-except } b$ 
proof -
  have  $\bigwedge a \ c. \llbracket a \neq b; b \neq c \rrbracket \implies i \ \text{dictates } a < c$ 
  proof (unfold  $\text{dictates-def}$ , intro allI impI)
    fix  $a \ c$  and  $P::\text{prof}$ 
    assume  $abc: a \neq b \ b \neq c$  and
       $ac: P \ i \ a < P \ i \ c$ 
    show  $F \ P \ a < F \ P \ c$ 
    proof -
      obtain  $P1 \ P2$  where
         $\text{Extreme } P1 \ b$  and  $b \cdot < F \ P1$  and  $b \cdot < P1 \ i$  and  $F \ P2 < \cdot \ b$  and
         $[\text{simp}]: P2 = P1(i := \text{mktop } (P1 \ i) \ b)$ 
      using  $\text{pivo}$  by (unfold  $\text{pivotal-def}$ ) fast
      let  $?P = \lambda j. \text{if } j=i \ \text{then between } (P \ j) \ a \ b \ c$ 
        else if  $P1 \ j < \cdot \ b$  then  $\text{mktop } (P \ j) \ b$  else  $\text{mkbot } (P \ j) \ b$ 
      have  $\text{eq}: (F \ P \ a < F \ P \ c) = (F \ ?P \ a < F \ ?P \ c)$ 

```

```

    using abc by - (rule IIA, auto)
  have F ?P a < F ?P b
  proof (rule IIA')
    fix j show (P2 j a < P2 j b) = (?P j a < ?P j b)
      using ⟨Extreme P1 b⟩ by(simp add: ac)
  next
    show F P2 a < F P2 b
      using ⟨F P2 <· b⟩ abc by(simp add: eq-sym-conv)
  qed
  also have ... < F ?P c
  proof (rule IIA')
    fix j show (P1 j b < P1 j c) = (?P j b < ?P j c)
      using ⟨Extreme P1 b⟩ ⟨b ·< P1 i⟩ by(simp add: ac)
  next
    show F P1 b < F P1 c
      using ⟨b ·< F P1⟩ abc by(simp add: eq-sym-conv)
  qed
  finally show ?thesis by(simp add: eq)
qed
qed
thus ?thesis by(unfold dictatesx-def) fast
qed

lemma pivotal-is-dictator:
  assumes pivo: pivotal i b and ab: a ≠ b and d: j dictates a,b
  shows i = j
  proof (rule ccontr)
    assume pd: i ≠ j
    obtain P1 P2 where Extreme P1 b and b ·< F P1 and F P2 <· b and
      P2: P2 = P1(i := mktop (P1 i) b)
      using pivo by (unfold pivotal-def) fast
    have ~ (P1 j a < P1 j b) (is ~ ?ab)
    proof
      assume ?ab
      hence F P1 a < F P1 b using d by(simp add: dictates-def dictates2-def)
      with ⟨b ·< F P1⟩ show False by simp
    qed
    hence P1 j b < P1 j a using ⟨Extreme P1 b⟩[THEN spec, of j] ab
      unfolding extreme-def top-def bot-def by metis
    hence P2 j b < P2 j a using pd by (simp add:P2)
    hence F P2 b < F P2 a using d by(simp add: dictates-def dictates2-def)
    with ⟨F P2 <· b⟩ show False by simp
  qed

theorem dictator: ∃ i. dictator i
  proof -
    from pivotal-exists[of b] obtain i where pivo: pivotal i b ..
    { fix a assume neg: a ≠ b have i dictates a,b

```

```

proof –
  obtain  $c$  where  $dist: distinct[a,b,c]$ 
    using  $neq\ third\text{-}alt$  by  $blast$ 
  obtain  $j$  where  $pivotal\ j\ c$  using  $pivotal\text{-}exists$  by  $fast$ 
  hence  $j\ dictates\text{-}except\ c$  by( $rule\ pivotal\text{-}xdictates$ )
  hence  $b: j\ dictates\ a,b$ 
    using  $dist$  by( $simp\ add:dictatesx\text{-}def\ dictates2\text{-}def\ eq\text{-}sym\text{-}conv$ )
  with  $pivo\ neq$  have  $i = j$  by( $rule\ pivotal\text{-}is\text{-}dictator$ )
  thus  $?thesis$  using  $b$  by  $simp$ 
qed
}
with  $pivotal\text{-}xdictates[OF\ pivo]$  have  $dictator\ i$ 
  by( $simp\ add: dictates\text{-}def\ dictatesx\text{-}def\ dictates2\text{-}def\ dictator\text{-}def$ )
  ( $metis\ real\text{-}less\text{-}def$ )
thus  $?thesis\ ..$ 
qed

end

end

```

2 Arrow’s Theorem for Strict Linear Orders

```

theory Arrow-Order imports Main FuncSet Zorn
begin

```

This theory formalizes the third proof due to Geanakoplos [1]. Preferences are modeled as strict linear orderings. The set of alternatives need not be finite.

Individuals are assumed to be finite but are not a priori identified with an initial segment of the naturals. In retrospect this generality appears gratuitous and complicates some of the low-level reasoning where we use a bijection with such an initial segment.

```

typedecl  $alt$ 
typedecl  $indi$ 

```

```

abbreviation  $I == (UNIV::indi\ set)$ 

```

```

axioms
 $alt3: \exists a\ b\ c::alt.\ distinct[a,b,c]$ 
 $finite\text{-}indi: finite\ I$ 

```

```

abbreviation  $N == card\ I$ 

```

```

lemma  $third\text{-}alt: a \neq b \implies \exists c::alt.\ distinct[a,b,c]$ 
using  $alt3$  by  $simp\ metis$ 

```

```

lemma  $alt2: \exists b::alt.\ b \neq a$ 

```

using *alt3* **by** *simp metis*

types *pref* = (*alt* * *alt*)*set*

definition *Lin* == {*L*::*pref*. *strict-linear-order* *L*}

lemmas *slo-defs* = *Lin-def strict-linear-order-on-def total-on-def irrefl-def*

lemma *notin-Lin-iff*: $L : Lin \implies x \neq y \implies (x,y) \notin L \iff (y,x) : L$

apply (*auto simp add: slo-defs*)

apply (*metis trans-def*)

done

lemma *converse-in-Lin*[*simp*]: $L^{-1} : Lin \iff L : Lin$

apply (*simp add: slo-defs*)

apply *blast*

done

lemma *Lin-irrefl*: $L : Lin \implies (a,b) : L \implies (b,a) : L \implies False$

by (*simp add: slo-defs*) (*metis trans-def*)

corollary *linear-alt*: $\exists L :: pref. L : Lin$

using *well-order-on* [**where** '*a* = *alt*, of *UNIV*]

apply (*auto simp: well-order-on-def Lin-def*)

apply (*metis strict-linear-order-on-diff-Id*)

done

abbreviation

rem :: *pref* \Rightarrow *alt* \Rightarrow *pref* **where**

rem *L* *a* \equiv {(*x*,*y*). (*x*,*y*) \in *L* \wedge *x* \neq *a* \wedge *y* \neq *a*}

definition

mktop :: *pref* \Rightarrow *alt* \Rightarrow *pref* **where**

mktop *L* *b* \equiv *rem* *L* *b* \cup {(*x*,*b*) | *x*. *x* \neq *b*}

definition

mkbot :: *pref* \Rightarrow *alt* \Rightarrow *pref* **where**

mkbot *L* *b* \equiv *rem* *L* *b* \cup {(*b*,*y*) | *y*. *y* \neq *b*}

definition

below :: *pref* \Rightarrow *alt* \Rightarrow *alt* \Rightarrow *pref* **where**

below *L* *a* *b* \equiv *rem* *L* *a* \cup

{(*a*,*b*)} \cup {(*x*,*a*) | *x*. (*x*,*b*) : *L* \wedge *x* \neq *a*} \cup {(*a*,*y*) | *y*. (*b*,*y*) : *L* \wedge *y* \neq *a*}

definition

above :: *pref* \Rightarrow *alt* \Rightarrow *alt* \Rightarrow *pref* **where**

above *L* *a* *b* \equiv *rem* *L* *b* \cup

{(*a*,*b*)} \cup {(*x*,*b*) | *x*. (*x*,*a*) : *L* \wedge *x* \neq *b*} \cup {(*b*,*y*) | *y*. (*a*,*y*) : *L* \wedge *y* \neq *b*}

lemma *in-mktop*: $(x,y) \in mktop\ L\ z \iff x \neq z \wedge (\text{if } y=z \text{ then } x \neq y \text{ else } (x,y) \in L)$

by (*auto simp: mktop-def*)

lemma *in-mkbot*: $(x,y) \in \text{mkbot } L \ z \iff y \neq z \wedge (\text{if } x=z \text{ then } x \neq y \text{ else } (x,y) \in L)$
by(*auto simp:mkbot-def*)

lemma *in-above*: $a \neq b \implies L:\text{Lin} \implies$
 $(x,y) : \text{above } L \ a \ b \iff x \neq y \wedge$
 $(\text{if } x=b \text{ then } (a,y) : L \ \text{else}$
 $\text{if } y=b \text{ then } x=a \mid (x,a) : L \ \text{else } (x,y) : L)$
by(*auto simp:above-def slo-defs*)

lemma *in-below*: $a \neq b \implies L:\text{Lin} \implies$
 $(x,y) : \text{below } L \ a \ b \iff x \neq y \wedge$
 $(\text{if } y=a \text{ then } (x,b) : L \ \text{else}$
 $\text{if } x=a \text{ then } y=b \mid (b,y) : L \ \text{else } (x,y) : L)$
by(*auto simp:below-def slo-defs*)

declare $[[\text{simp-depth-limit} = 2]]$

lemma *mktop-Lin*: $L : \text{Lin} \implies \text{mktop } L \ x : \text{Lin}$
by(*auto simp add:slo-defs mktop-def trans-def*)
lemma *mkbot-Lin*: $L : \text{Lin} \implies \text{mkbot } L \ x : \text{Lin}$
by(*auto simp add:slo-defs trans-def mkbot-def*)

lemma *below-Lin*: $x \neq y \implies L : \text{Lin} \implies \text{below } L \ x \ y : \text{Lin}$
unfolding *slo-defs below-def trans-def*
apply(*simp*)
apply *blast*
done

lemma *above-Lin*: $x \neq y \implies L : \text{Lin} \implies \text{above } L \ x \ y : \text{Lin}$
unfolding *slo-defs above-def trans-def*
apply(*simp*)
apply *blast*
done

declare $[[\text{simp-depth-limit} = 50]]$

abbreviation *lessLin* :: $\text{alt} \Rightarrow \text{pref} \Rightarrow \text{alt} \Rightarrow \text{bool} \ ((- <_ -) [51, 51] 50)$
where $a <_L b == (a,b) : L$

definition *Prof* = $I \rightarrow \text{Lin}$

abbreviation *SWF* == $\text{Prof} \rightarrow \text{Lin}$

definition *unanimity* $F == \forall P \in \text{Prof} . \forall a \ b . (\forall i . a <_{P \ i} b) \longrightarrow a <_F P \ b$

definition *IIA* $F == \forall P \in \text{Prof} . \forall P' \in \text{Prof} . \forall a \ b .$
 $(\forall i . a <_{P \ i} b \iff a <_{P' \ i} b) \longrightarrow (a <_F P \ b \iff a <_F P' \ b)$

definition *dictator* $F \ i == \forall P \in \text{Prof} . F \ P = P \ i$

```

lemma dictatorI:  $F : SWF \implies$ 
   $\forall P \in Prof. \forall a b. a \neq b \implies (a,b) : P i \implies (a,b) : F P \implies$  dictator  $F i$ 
apply(simp add:dictator-def Prof-def Pi-def Lin-def strict-linear-order-on-def)
apply safe
apply(erule-tac x=P in allE)
apply(erule-tac x=P in allE)
apply(simp add:total-on-def irreft-def)
apply (metis trans-def)
apply (metis irreft-def)
done

```

```

lemma const-Lin-Prof:  $L:Lin \implies (\%p. L) : Prof$ 
by(simp add:Prof-def Pi-def)

```

```

lemma complete-Lin: assumes  $a \neq b$  shows  $EX L:Lin. (a,b) : L$ 
proof –
  from linear-alt obtain  $R$  where  $R:Lin$  by auto
  thus ?thesis by (metis assms in-mkbot mkbot-Lin)
qed

```

```

declare Let-def[simp]

```

```

theorem Arrow: assumes  $F : SWF$  and  $u$ : unanimity  $F$  and IIA  $F$ 
shows  $EX i. dictator F i$ 
proof –

```

```

  { fix  $a b a' b'$  and  $P P'$ 
    assume  $d1: a \neq b a' \neq b'$  and  $d2: a \neq b' b \neq a'$  and
       $P : Prof P' : Prof$  and  $1: \forall i. (a,b) : P i \longleftrightarrow (a',b') : P' i$ 
    assume  $(a,b) : F P$ 
    def  $Q \equiv \%i. let L = (if a=a' then P i else below (P i) a' a)$ 
       $in if b=b' then L else above L b b'$ 
    have  $Q : Prof$  using  $\langle P : Prof \rangle$ 
      by(simp add:Q-def Prof-def Pi-def above-Lin below-Lin)
    hence  $F Q : Lin$  using  $\langle F : SWF \rangle$  by(simp add:Pi-def)
    have  $\forall i. (a,b) : P i \longleftrightarrow (a,b) : Q i$  using  $d1 d2 \langle P : Prof \rangle$ 
      by(simp add:in-above in-below Q-def Prof-def Pi-def below-Lin)
    hence  $(a,b) : F Q$  using  $\langle (a,b) : F P \rangle \langle IIA F \rangle \langle P:Prof \rangle \langle Q : Prof \rangle$ 
      unfolding IIA-def by blast
    moreover
    { assume  $a \neq a'$ 
      hence  $!!i. (a',a) : Q i$  using  $d1 d2 \langle P : Prof \rangle$ 
        by(simp add:in-above in-below Q-def Prof-def Pi-def below-Lin)
      hence  $(a',a) : F Q$  using  $u \langle Q : Prof \rangle$  by(simp add:unanimity-def)
    } moreover
    { assume  $b \neq b'$ 
      hence  $!!i. (b,b') : Q i$  using  $d1 d2 \langle P : Prof \rangle$ 
        by(simp add:in-above in-below Q-def Prof-def Pi-def below-Lin)
      hence  $(b,b') : F Q$  using  $u \langle Q : Prof \rangle$  by(simp add:unanimity-def)
    }
  }

```

```

}
ultimately have (a',b') : F Q using (F Q : Lin)
  unfolding slo-defs trans-def by blast
moreover
have  $\forall i. (a',b') : Q i \longleftrightarrow (a',b') : P' i$  using d1 d2 (P : Prof) 1
  by(simp add:Q-def in-below in-above Prof-def Pi-def below-Lin) blast
ultimately have (a',b') : F P'
  using (IIA F) (P' : Prof) (Q : Prof) unfolding IIA-def by blast
} note 1 = this
{ fix a b a' b' and P P'
  assume  $a \neq b$   $a' \neq b'$   $a \neq b'$   $b \neq a'$  P : Prof P' : Prof
     $\forall i. (a,b) : P i \longleftrightarrow (a',b') : P' i$ 
  hence (a,b) : F P  $\longleftrightarrow$  (a',b') : F P' using 1 by blast
} note 2 = this
{ fix a b P P'
  assume  $a \neq b$  P : Prof P' : Prof and
    iff:  $\forall i. (a,b) : P i \longleftrightarrow (b,a) : P' i$ 
  from (a  $\neq$  b) obtain c where dist: distinct[a,b,c] using third-alt by metis
  let ?Q = %p. below (P p) c b
  let ?R = %p. below (?Q p) b a
  let ?S = %p. below (?R p) a c
  have ?Q : Prof using (P : Prof) dist
    by(auto simp add:Prof-def Pi-def below-Lin)
  hence ?R : Prof using dist by(auto simp add:Prof-def Pi-def below-Lin)
  hence ?S : Prof using dist by(auto simp add:Prof-def Pi-def below-Lin)
  have  $\forall i. (a,b) : P i \longleftrightarrow (a,c) : ?Q i$  using (P : Prof) dist
    by(auto simp add:in-below Prof-def Pi-def)
  hence ab: (a,b) : F P  $\longleftrightarrow$  (a,c) : F ?Q
    using 2 (P : Prof) (?Q : Prof) dist[simplified] by (blast)
  have  $\forall i. (a,c) : ?Q i \longleftrightarrow (b,c) : ?R i$  using (P : Prof) dist
    by(auto simp add:in-below Prof-def Pi-def below-Lin)
  hence ac: (a,c) : F ?Q  $\longleftrightarrow$  (b,c) : F ?R
    using 2 (?Q : Prof) (?R : Prof) dist[simplified] by (blast)
  have  $\forall i. (b,c) : ?R i \longleftrightarrow (b,a) : ?S i$  using (P : Prof) dist
    by(auto simp add:in-below Prof-def Pi-def below-Lin)
  hence bc: (b,c) : F ?R  $\longleftrightarrow$  (b,a) : F ?S
    using (?R : Prof) (?S : Prof) dist[simplified]
  apply - apply(rule 2) by fast+
  have  $\forall i. (b,a) : ?S i \longleftrightarrow (a,b) : P i$  using (P : Prof) dist
    by(auto simp add:in-below Prof-def Pi-def below-Lin)
  hence  $\forall i. (b,a) : ?S i \longleftrightarrow (b,a) : P' i$  using iff by blast
  hence ba: (b,a) : F ?S  $\longleftrightarrow$  (b,a) : F P'
    using (IIA F) (P' : Prof) (?S : Prof) unfolding IIA-def by fast
  from ab ac bc ba have (a,b) : F P  $\longleftrightarrow$  (b,a) : F P' by simp
} note 3 = this
{ fix a b c P P'
  assume A:  $a \neq b$   $b \neq c$   $a \neq c$  P : Prof P' : Prof and
    iff:  $\forall i. (a,b) : P i \longleftrightarrow (b,c) : P' i$ 
  hence  $\forall i. (b,a) : (converse o P)i \longleftrightarrow (b,c) : P' i$  by simp

```

```

moreover have  $cP$ :  $\text{converse } o P : \text{Prof}$ 
  using  $\langle P:\text{Prof} \rangle$  by( $\text{simp add}:\text{Prof-def } Pi\text{-def}$ )
ultimately have  $(b,a) : F(\text{converse } o P) \longleftrightarrow (b,c) : F P'$  using  $A$ 
  apply – apply( $\text{rule } 2$ ) by  $\text{fast+}$ 
moreover have  $(a,b) : F P \longleftrightarrow (b,a) : F(\text{converse } o P)$ 
  by ( $\text{rule } 3[\text{OF } \langle a \neq b \rangle \langle P:\text{Prof} \rangle cP]$ )  $\text{simp}$ 
ultimately have  $(a,b) : F P \longleftrightarrow (b,c) : F P'$  by  $\text{blast}$ 
} note  $4 = \text{this}$ 
{ fix  $a b a' b' :: \text{alt and } P P'$ 
  assume  $A: a \neq b a' \neq b' P : \text{Prof } P' : \text{Prof}$ 
     $\forall i. (a,b) : P i \longleftrightarrow (a',b') : P' i$ 
  have  $(a,b) : F P \longleftrightarrow (a',b') : F P'$ 
proof –
  { assume  $a \neq b' \ \& \ b \neq a'$  hence  $?thesis$  using  $2 A$  by  $\text{blast}$  }
  moreover
  { assume  $a = b' \ \& \ b \neq a'$  hence  $?thesis$  using  $4 A$  by  $\text{blast}$  }
  moreover
  { assume  $a \neq b' \ \& \ b = a'$  hence  $?thesis$  using  $4 A$  by  $\text{blast}$  }
  moreover
  { assume  $a = b' \ \& \ b = a'$  hence  $?thesis$  using  $3 A$  by  $\text{blast}$  }
  ultimately show  $?thesis$  by  $\text{blast}$ 
}
qed
} note  $\text{pairwise-neutrality} = \text{this}$ 
obtain  $h :: \text{indi} \Rightarrow \text{nat}$  where
   $\text{inj}h: \text{inj } h$  and  $\text{surj}h: h \text{ ' } I = \{0..<N\}$ 
  by( $\text{metis ex-bij-betw-finite-nat}[\text{OF } \text{finite-indi}] \text{bij-betw-def}$ )
obtain  $a b :: \text{alt}$  where  $a \neq b$  using  $\text{alt}3$  by  $\text{auto}$ 
obtain  $\text{Lab}$  where  $(a,b) : \text{Lab } \text{Lab}:\text{Lin}$  using  $\langle a \neq b \rangle$  by ( $\text{metis complete-Lin}$ )
hence  $(b,a) \notin \text{Lab}$  by( $\text{simp add}:\text{slo-defs trans-def}$ )  $\text{metis}$ 
obtain  $\text{Lba}$  where  $(b,a) : \text{Lba } \text{Lba}:\text{Lin}$  using  $\langle a \neq b \rangle$  by ( $\text{metis complete-Lin}$ )
hence  $(a,b) \notin \text{Lba}$  by( $\text{simp add}:\text{slo-defs trans-def}$ )  $\text{metis}$ 
let  $?Pi = \%n. \%i. \text{if } h \ i < n \ \text{then } \text{Lab} \ \text{else } \text{Lba}$ 
have  $Pi\text{Prof}: !!n. ?Pi \ n : \text{Prof}$  using  $\langle \text{Lab}:\text{Lin} \rangle \langle \text{Lba}:\text{Lin} \rangle$ 
  unfolding  $\text{Prof-def } Pi\text{-def}$  by  $\text{simp}$ 
have  $EX \ n < N. (\forall m \leq n. (b,a) : F(?Pi \ m)) \ \& \ (a,b) : F(?Pi \ (n+1))$ 
proof –
  have  $0: !!n. F(?Pi \ n) : \text{Lin}$  using  $\langle F : \text{SWF} \rangle Pi\text{Prof}$  by( $\text{simp add}:\text{Pi-def}$ )
  have  $F(\%i. \text{Lba}):\text{Lin}$  using  $\langle F:\text{SWF} \rangle \langle \text{Lba}:\text{Lin} \rangle$  by( $\text{simp add}:\text{Prof-def } Pi\text{-def}$ )
  hence  $1: (a,b) \notin F(?Pi \ 0)$  using  $u \ \langle (a,b) \notin \text{Lba} \rangle \langle \text{Lba}:\text{Lin} \rangle \langle \text{Lba}:\text{Lin} \rangle \langle a \neq b \rangle$ 
  by( $\text{simp add}:\text{unanimity-def notin-Lin-iff const-Lin-Prof}$ )
  have  $?Pi \ N = (\%p. \text{Lab})$  using  $\text{surjh}$ 
  by( $\text{auto simp}:\text{image-def expand-fun-eq Bex-def Collect-def}$ 
     $\text{atLeastLessThan-def lessThan-def}$ )
moreover
have  $F(\%i. \text{Lab}):\text{Lin}$  using  $\langle F:\text{SWF} \rangle \langle \text{Lab}:\text{Lin} \rangle$  by( $\text{simp add}:\text{Prof-def } Pi\text{-def}$ )
ultimately have  $2: (a,b) \in F(?Pi \ N)$  using  $u \ \langle (a,b) : \text{Lab} \rangle \langle \text{Lab}:\text{Lin} \rangle$ 
  by( $\text{simp add}:\text{unanimity-def const-Lin-Prof}$ )
with  $\text{ex-least-nat-less}[\%n. (a,b) : F(?Pi \ n), \text{OF } 1 \ 2]$ 
   $\text{notin-Lin-iff}[\text{OF } 0 \ \langle a \neq b \rangle]$ 

```

```

    show ?thesis by simp
qed
then obtain n where n: n < N  $\forall m \leq n. (b,a) : F(?Pi m) (a,b) : F(?Pi(n+1))$ 
  by blast
have dictator F (inv h n)
proof (rule dictatorI [OF ⟨F : SWF⟩], auto)
  fix P c d assume P ∈ Prof c ≠ d (c,d) ∈ P(inv h n)
  then obtain e where dist: distinct[c,d,e] using third-alt by metis
  let ?W = %i. if h i < n then mktop (P i) e else
    if h i = n then above (P i) c e else mkbtop (P i) e
  have ?W : Prof using ⟨P : Prof⟩ dist
    by (simp add: Pi-def Prof-def mkbtop-Lin mktop-Lin above-Lin)
  have  $\forall i. (c,d) : P i \longleftrightarrow (c,d) : ?W i$  using dist ⟨P : Prof⟩
    by (auto simp: in-above in-mkbtop in-mktop Prof-def Pi-def)
  hence PW: (c,d) : F P  $\longleftrightarrow (c,d) : F ?W$ 
    using ⟨IIA F⟩[unfolded IIA-def] ⟨P:Prof⟩ ⟨?W:Prof⟩ by fast
  have  $\forall i. (c,e) : ?W i \longleftrightarrow (a,b) : ?Pi (n+1) i$  using dist ⟨P : Prof⟩
    by (auto simp: ⟨(a,b):Lab⟩ ⟨(a,b)∉Lba⟩
      in-mkbtop in-mktop in-above Prof-def Pi-def)
  hence (c,e) : F ?W  $\longleftrightarrow (a,b) : F(?Pi(n+1))$ 
    using pairwise-neutrality[of c e a b ?W ?Pi(n+1)]
      ⟨a≠b⟩ dist ⟨?W : Prof⟩ PiProf by simp
  hence (c,e) : F ?W using n(3) by blast
  have  $\forall i. (e,d) : ?W i \longleftrightarrow (b,a) : ?Pi n i$ 
    using dist ⟨P : Prof⟩ ⟨(c,d) ∈ P(inv h n)⟩ ⟨inj h⟩
    by (auto simp: ⟨(b,a):Lba⟩ ⟨(b,a)∉Lab⟩
      in-mkbtop in-mktop in-above Prof-def Pi-def)
  hence (e,d) : F ?W  $\longleftrightarrow (b,a) : F(?Pi n)$ 
    using pairwise-neutrality[of e d b a ?W ?Pi n]
      ⟨a≠b⟩ dist ⟨?W : Prof⟩ PiProf by simp blast
  hence (e,d) : F ?W using n(2) by auto
  with ⟨(c,e) : F ?W⟩ ⟨?W : Prof⟩ ⟨F:SWF⟩
  have (c,d) ∈ F ?W unfolding Pi-def slo-defs trans-def by blast
  thus (c,d) ∈ F P using PW by blast
qed
thus ?thesis ..
qed
end

```

3 The Gibbard-Satterthwaite Theorem

theory GS imports Arrow-Order
begin

The Gibbard-Satterthwaite theorem as a corollary to Arrow's theorem.
The proof follows Nisan [2].

definition manipulable f == $\exists P \in \text{Prof}. \exists i. \exists L \in \text{Lin}. (f P, f(P(i:=L))) : P i$

definition $dict\ f\ i == \forall P \in Prof. \forall a. a \neq f\ P \longrightarrow (a, f\ P) : P\ i$

definition

$Top :: alt\ set \Rightarrow pref \Rightarrow pref$ **where**
 $Top\ S\ L \equiv \{(a,b). (a,b) \in L \wedge (a \in S \wedge b \in S \vee a \notin S \wedge b \notin S)\} \cup$
 $\{(a,b). a \notin S \wedge b \in S\}$

lemma $Top\text{-in-Lin}: L:Lin \Longrightarrow Top\ S\ L : Lin$

apply ($simp\ add:Top\text{-def}\ slo\text{-defs}\ Sigma\text{-def}$)

unfolding $trans\text{-def}$

apply $blast$

done

lemma $Top\text{-in-Prof}: P:Prof \Longrightarrow Top\ S\ o\ P : Prof$

by ($simp\ add:Prof\text{-def}\ Pi\text{-def}\ Top\text{-in-Lin}$)

lemma $not\text{-manipulable}: \neg\ manipulable\ f \longleftrightarrow$

$(\forall P \in Prof. \forall i. \forall L \in Lin. f\ P \neq f(P(i:=L)) \longrightarrow$
 $(f(P(i:=L)), f\ P) : P\ i \wedge (f\ P, f(P(i:=L))) : L) \text{ (is } ?A = ?B)$

proof

assume $?A$

show $?B$

proof ($clarsimp$)

fix $P\ i\ L$ **assume** $0: P \in Prof\ L \in Lin\ f\ P \neq f(P(i:=L))$

moreover **hence** $1: P\ i: Lin\ P(i:=L): Prof$ **by** ($simp\ add:Prof\text{-def}\ Pi\text{-def}$) $+$

ultimately **have** $(f(P(i:=L)), f\ P) \in P\ i$ **(is** $?L$)

using $\langle ?A \rangle$ **unfolding** $manipulable\text{-def}$ **by** ($metis\ notin\text{-Lin}\text{-iff}$)

moreover **have** $(f\ P, f(P(i:=L))) \in L$ **(is** $?R$)

using $0\ 1\ fun\text{-upd}\text{-upd}[of\ P]\ fun\text{-upd}\text{-triv}[of\ P]\ fun\text{-upd}\text{-same}[of\ P]$

using $\langle ?A \rangle$ **unfolding** $manipulable\text{-def}$ **by** ($metis\ notin\text{-Lin}\text{-iff}$)

ultimately **show** $?L \wedge ?R$ **..**

qed

next

assume $?B$

show $?A$

proof ($clarsimp\ simp:manipulable\text{-def}$)

fix $P\ i\ L$ **assume** $P \in Prof\ L \in Lin\ (f\ P, f(P(i:=L))) \in P\ i$

moreover **hence** $P\ i: Lin$ **by** ($simp\ add:Prof\text{-def}\ Pi\text{-def}$)

ultimately **show** $False$

using $\langle ?B \rangle$ **by** ($metis\ Lin\text{-irrefl}$)

qed

qed

definition $swf(f) \equiv \lambda P. \{(a,b). a \neq b \wedge f(Top\ \{a,b\}\ o\ P) = b\}$

locale $GS =$

fixes f

assumes $not\text{-manip}: \neg\ manipulable\ f$

and onto: $f \text{ ' } Prof = UNIV$
begin

lemma nonmanip:

$P:Prof \implies L:Lin \implies f(P(i := L)) \neq f P \implies$
 $(f(P(i := L)), f P) : P i \wedge (f P, f(P(i := L))) : L$
using not-manip by(metis not-manipulable)

lemma mono:

assumes $P \in Prof \ P' \in Prof \ \forall i \ a. (a, f P) : P i \longrightarrow (a, f P) : P' i$
shows $f P' = f P$
proof –

obtain $h :: indi \Rightarrow nat$ **where**

$inj h$ **and** $surjh: h \text{ ' } I = \{0..<N\}$

by(metis ex-bij-betw-finite-nat[OF finite-indi] bij-betw-def)

let $?M = \%n \ i. \text{ if } h \ i < n \text{ then } P' \ i \text{ else } P \ i$

have $N: !!i. h \ i < N$ **using** $surjh$ **by** *auto*

have $MProf: !!n. ?M \ n : Prof$ **and** $P'Lin: !!i. P' \ i : Lin$

using $\langle P:Prof \rangle \ \langle P':Prof \rangle$ **by**(simp add:Prof-def Pi-def)+

{ fix n **have** $n \leq N \implies f(?M \ n) = f P$

proof(*induct* n)

case 0 **show** $?case$ **by** *simp*

next

case (*Suc* n)

let $?up = (?M \ n)(inv \ h \ n := P' \ (inv \ h \ n))$

have $1: ?M(Suc \ n) = ?up$ **using** $surjh \ Suc(2)$

by(*simp* (*no-asm-simp*) add:expand-fun-eq f-inv-into-f)
(metis *inj* *inv-f-f* *less-antisym*)

show $?case$

proof(*rule ccontr*)

assume $\neg ?case$

with $\langle ?M(Suc \ n) = ?up \rangle \ Suc$ **have** $0: f \ ?up \neq f(?M \ n)$ **by** *simp*

from *nonmanip*[*OF* $MProf \ P'Lin \ 0$] *assms*(3) **show** *False*

using $N \ surjh \ Suc \ Lin-irrefl[*OF* \ P'Lin]$

by(*fastsimp* *simp: f-inv-into-f*)

qed

qed

}
from *this*[*of* N] N **show** $f P' = f P$ **by** *simp*

qed

lemma una-Top: **assumes** $P:Prof \ S \neq \{\}$ **shows** $f(Top \ S \ o \ P) : S$

proof –

obtain $h :: indi \Rightarrow nat$ **where**

$inj h$ **and** $surjh: h \text{ ' } I = \{0..<N\}$

by(metis ex-bij-betw-finite-nat[OF finite-indi] bij-betw-def)

from *assms* **obtain** a **where** $a:S$ **by** *blast*

from *onto* **obtain** Pa **where** $Pa:Prof \ f \ Pa = a$

by(metis *inv-into-into* *UNIV-I* *f-inv-into-f*)

```

let ?M = %n i. if h i < n then Top S (P i) else Pa i
have N: !!i. h i < N using surjh by auto
have MProf: !!n. ?M n : Prof using ⟨P:Prof⟩ ⟨Pa:Prof⟩
  by(simp add:Prof-def Pi-def Top-in-Lin mktop-Lin)
{ fix n have n<=N ==> f(?M n) : S
  proof(induct n)
    case 0 thus ?case using ⟨f Pa = a⟩ ⟨a:S⟩ by simp
  next
    case (Suc n)
    show ?case
    proof cases
      assume f(?M n) = f(?M(Suc n))
      thus ?case using Suc by simp
    next
      let ?up = (?M n)(inv h n := Top S (P(inv h n)))
      assume f(?M n) ≠ f(?M(Suc n))
      also have eq: ?M(Suc n) = ?up using surjh Suc
        by(simp (no-asm-simp) add:expand-fun-eq f-inv-into-f)
          (metis injh inv-f-eq less-antisym)
      finally have n: f(?M n) ≠ f(?up) .
      with nonmanip[OF MProf Top-in-Lin n[symmetric]] Suc eq ⟨P:Prof⟩
      show ?case by (simp add:Top-def Prof-def Pi-def)
    qed
  qed
}
from this[of N] N show ?thesis by(simp add:comp-def)
qed

```

```

lemma SWF-swf: swf f : SWF
proof (rule Pi-I)
  fix P assume P: Prof
  show swf f P : Lin
  proof(unfold Lin-def strict-linear-order-on-def, auto)
    show total(swf f P)
    proof(simp add: total-on-def, intro allI impI)
      fix a b :: alt assume a≠b
      thus (a,b) ∈ swf f P ∨ (b,a) ∈ swf f P
        unfolding swf-def using una-Top[of P {a,b}] ⟨P:Prof⟩
        by simp(metis insert-commute)
    qed
  show irrefl(swf f P) by(simp add: irrefl-def swf-def)
  show trans(swf f P)
  proof (clarsimp simp:trans-def swf-def insert-commute)
    fix a b c assume a≠b b≠c f(Top{a,b} ∘ P) = b f(Top{b,c} ∘ P) = c
    hence a≠c by(auto simp: insert-commute)
    note 3 = Top-in-Prof[OF ⟨P:Prof⟩, of {a,b,c}]
    { assume f (Top {a, b, c} ∘ P) = a
      hence f(Top{a,b} ∘ P) = a
        using mono[OF 3 Top-in-Prof[OF ⟨P:Prof⟩], of {a,b}]
    }
  qed

```

```

    by(auto simp:Top-def)
  with ⟨f(Top{a,b} ∘ P) = b⟩ ⟨a≠b⟩ have False by simp
} moreover
{ assume f (Top {a, b, c} ∘ P) = b
  hence f(Top{b,c} ∘ P) = b
    using mono[OF ∃ Top-in-Prof[OF ⟨P:Prof⟩], of {b,c}]
    by(auto simp:Top-def)
  with ⟨f(Top{b,c} ∘ P) = c⟩ ⟨b≠c⟩ have False by simp
}
ultimately have f (Top {a, b, c} ∘ P) = c
  using una-Top[OF ⟨P:Prof⟩, of {a,b,c}, simplified] by blast
hence f(Top{a,c} ∘ P) = c (is ?R)
  using mono[OF ∃ Top-in-Prof[OF ⟨P:Prof⟩], of {a,c}]
  by (auto simp:Top-def)
thus a≠c ∧ ?R using ⟨a≠c⟩ by blast
qed
qed
qed

```

```

lemma Top-top: L:Lin ⟹ (!!a. a≠b ⟹ (a,b) : L) ⟹ Top {b} L = L
apply(auto simp:Top-def slo-defs)
apply (metis trans-def)
apply (metis trans-def)
done

```

```

lemma una-swf: unanimity(swf f)
proof(clarsimp simp:swf-def unanimity-def)
  fix P a b
  assume P:Prof and abP: ∀ i. (a, b) ∈ P i
  hence a ≠ b by(fastsimp simp:Prof-def Pi-def slo-defs)
  let ?abP = Top {a,b} ∘ P
  have ?abP : Prof using ⟨P:Prof⟩ by(simp add:Prof-def Pi-def Top-in-Lin)
  have top: !!i c. b≠c ⟹ (c,b) : Top {a,b} (P i)
    using abP by(auto simp:Top-def)
  have Top {b} ∘ ?abP = ?abP using ⟨P:Prof⟩
    by (simp add:expand-fun-eq top Top-top Prof-def Pi-def Top-in-Lin)
  moreover have f(Top {b} ∘ ?abP) = b
    by (metis una-Top[OF ⟨?abP : Prof⟩] empty-not-insert singletonE)
  ultimately have f ?abP = b by simp
  thus a≠b ∧ f ?abP = b using ⟨a≠b⟩ by blast
qed

```

```

lemma IIA-swf: IIA(swf f)
proof(clarsimp simp:IIA-def)
  fix P P' a b
  assume P:Prof P':Prof and iff: ∀ i. ((a,b) ∈ P i) = ((a,b) ∈ P' i)
  hence [simp]: !!i x. (x,x) ∼: P i ∧ (x,x) ∼: P' i
    by(simp add:Prof-def Pi-def slo-defs)
  have iff': a≠b ⟹ (∀ i. ((b,a) ∈ P i) = ((b,a) ∈ P' i))

```

using *iff* $\langle P:\text{Prof} \rangle \langle P':\text{Prof} \rangle$ **unfolding** *Prof-def Pi-def*
by *simp (metis iff notin-Lin-iff)*
let $?abP = \text{Top } \{a,b\} \circ P$ **let** $?abP' = \text{Top } \{a,b\} \circ P'$
have $\forall i c. (c, f ?abP) : ?abP i \longrightarrow (c, f ?abP) : ?abP' i$
using *una-Top[of P {a,b}, OF $\langle P:\text{Prof} \rangle$] iff iff' by(auto simp add:Top-def)*
then have $f (\text{Top } \{a,b\} \circ P) = f (\text{Top } \{a,b\} \circ P')$
using *Top-in-Prof[OF $\langle P:\text{Prof} \rangle$] Top-in-Prof[OF $\langle P':\text{Prof} \rangle$]*
mono[of Top {a, b} \circ P] by metis
thus $(a <_{\text{swf } f} P b) = (a <_{\text{swf } f} P' b)$ **by** *(simp add: swf-def)*
qed

lemma *dict-swf*: **assumes** *dictator (swf f) i* **shows** *dict f i*
proof *(auto simp:dict-def)*
fix $P a$ **assume** $P:\text{Prof } a \neq f P$
have $f (\text{Top } \{a,f P\} \circ P) = f P$
using *mono[OF $\langle P:\text{Prof} \rangle$ Top-in-Prof[OF $\langle P:\text{Prof} \rangle$,of {a,f P}]]*
by *(auto simp:Top-def)*
moreover have $P i = \{(a,b). a \neq b \wedge f(\text{Top } \{a,b\} \circ P) = b\}$
using *assms $\langle P:\text{Prof} \rangle$ by(simp add:dictator-def swf-def)*
ultimately show $(a,f P) : P i$ **using** $\langle a \neq f P \rangle$ **by** *simp*
qed

theorem *Gibbard-Satterthwaite*:
 $\exists i. \text{dict } f i$
by *(metis Arrow SWF-swf una-swf IIA-swf dict-swf)*
end

theorem *Gibbard-Satterthwaite*:
 $\neg \text{manipulable } f \implies \forall a. \exists P \in \text{Prof}. a = f P \implies \exists i. \text{dict } f i$
using *GS.Gibbard-Satterthwaite[of f,unfolded GS-def]*
by *blast*
end

References

- [1] J. Geanakoplos. Three brief proofs of Arrow's impossibility theorem. *Economic Theory*, 26:211–215, 2005.
- [2] N. Nisan. Introduction to mechanism design (for computer scientists). In N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*. Cambridge University Press, 2007.