

Arrow and Gibbard-Satterthwaite

Tobias Nipkow

December 12, 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*: $a \neq b \implies \exists c::alt. distinct[a,b,c]$
<proof>

lemma *alt2*: $\exists b::alt. b \neq a$
<proof>

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

$Extreme P b == \forall i. extreme (P i) b$

lemma $[simp]: r <= s \Longrightarrow r < s+(1::real)$

$\langle proof \rangle$

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

$\langle proof \rangle$

lemma $[simp]: r <= s \Longrightarrow \neg s+(1::real) < r$

$\langle proof \rangle$

lemma $[simp]: (r < s-(1::real)) = (r+1 < s)$

$\langle proof \rangle$

lemma $[simp]: (s-(1::real) < r) = (s < r+1)$

$\langle proof \rangle$

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

$\langle proof \rangle$

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

$\langle proof \rangle$

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

$\langle proof \rangle$

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

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

$\langle proof \rangle$

lemma $top-impl-not-bot[simp]: p <\cdot b \Longrightarrow \neg b \cdot < p$

$\langle proof \rangle$

lemma $[simp]: extreme p b \Longrightarrow (\neg p <\cdot b) = (b \cdot < p)$

$\langle proof \rangle$

lemma $[simp]: extreme p b \Longrightarrow (\neg b \cdot < p) = (p <\cdot b)$

$\langle proof \rangle$

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*]: $a \neq b \implies mktop\ p\ b\ a = p\ a$
 $\langle proof \rangle$

lemma [*simp*]: $a \neq b \implies mkbot\ p\ b\ a = p\ a$
 $\langle proof \rangle$

lemma [*simp*]: $a \neq b \implies p\ a < mktop\ p\ b\ b$
 $\langle proof \rangle$

lemma [*simp*]: $a \neq b \implies mkbot\ p\ b\ b < p\ a$
 $\langle proof \rangle$

lemma [*simp*]: $mktop\ p\ b < \cdot\ b$
 $\langle proof \rangle$

lemma [*simp*]: $\neg b \cdot < mktop\ p\ b$
 $\langle proof \rangle$

lemma [*simp*]: $a \neq b \implies \neg P\ p\ a < mkbot\ (P\ p)\ b\ b$
 $\langle proof \rangle$

The proof starts here.

locale *arrow* =

fixes $F :: pref \Rightarrow pref$

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

$dictates :: indi \Rightarrow alt \Rightarrow alt \Rightarrow bool$ ($-$ dictates $- < -$) **where**

$(i \text{ dictates } a < b) \equiv \forall P. P i a < P i b \longrightarrow F P a < F P b$

definition

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

definition

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

definition

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

definition

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

lemma *all-top[simp]*: $\forall i. P i < \cdot b \Longrightarrow F P < \cdot b$
<proof>

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>

lemma *extremal*:

assumes *extremes*: $\text{Extreme } P b$ **shows** $\text{extreme } (F P) b$
<proof>

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

shows $\bigwedge P. \llbracket D = \{i. b \cdot < P i\}; \text{Extreme } P b; b \cdot < F P \rrbracket$
 $\Longrightarrow \exists i. \text{pivotal } i b \text{ (is } \bigwedge P. ?D D P \Longrightarrow ?E P \Longrightarrow ?B P \Longrightarrow \cdot)$
<proof>

lemma *pivotal-exists*: $\exists i. \text{pivotal } i b$
<proof>

lemma *pivotal-xdictates*: **assumes** *pivo*: *pivotal i b*

shows $i \text{ dictates-except } b$
<proof>

lemma *pivotal-is-dictator*:

assumes *pivo*: *pivotal i b* **and** *ab*: $a \neq b$ **and** *d*: $j \text{ dictates } a, b$
shows $i = j$
<proof>

theorem *dictator*: $\exists i. \text{dictator } i$
<proof>

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 $I == (UNIV::indi \text{ set})$

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

abbreviation $N == \text{card } I$

lemma *third-alt*: $a \neq b \implies \exists c::alt. \text{distinct}[a,b,c]$
<proof>

lemma *alt2*: $\exists b::alt. b \neq a$
<proof>

types *pref* = $(alt * alt) \text{ set}$

definition $Lin == \{L::pref. \text{strict-linear-order } L\}$

lemmas *slo-defs* = $Lin\text{-def } \text{strict-linear-order-on-def } \text{total-on-def } \text{irrefl-def}$

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

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

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

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

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*: $(x,y) \in mktop L z \longleftrightarrow x \neq z \wedge (if y=z then x \neq y else (x,y) \in L)$
 ⟨proof⟩

lemma *in-mkbot*: $(x,y) \in mkbot L z \longleftrightarrow y \neq z \wedge (if x=z then x \neq y else (x,y) \in L)$
 ⟨proof⟩

lemma *in-above*: $a \neq b \implies L:Lin \implies$
 $(x,y) : above L a b \longleftrightarrow x \neq y \wedge$
 $(if x=b then (a,y) : L else$
 $if y=b then x=a \mid (x,a) : L else (x,y) : L)$
 ⟨proof⟩

lemma *in-below*: $a \neq b \implies L:Lin \implies$
 $(x,y) : below L a b \longleftrightarrow x \neq y \wedge$
 $(if y=a then (x,b) : L else$
 $if x=a then y=b \mid (b,y) : L else (x,y) : L)$
 ⟨proof⟩

declare $[[simp-depth-limit = 2]]$

lemma *mktop-Lin*: $L : Lin \implies mktop L x : Lin$
 ⟨proof⟩

lemma *mkbot-Lin*: $L : Lin \implies mkbot L x : Lin$
 ⟨proof⟩

lemma *below-Lin*: $x \neq y \implies L : Lin \implies \text{below } L \ x \ y : Lin$
<proof>

lemma *above-Lin*: $x \neq y \implies L : Lin \implies \text{above } L \ x \ y : Lin$
<proof>

declare [[*simp-depth-limit* = 50]]

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

definition $Prof = I \rightarrow Lin$

abbreviation $SWF == Prof \rightarrow Lin$

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

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

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

lemma *dictatorI*: $F : SWF \implies$
 $\forall P \in Prof. \forall a \ b. a \neq b \longrightarrow (a, b) : P \ i \longrightarrow (a, b) : F \ P \implies \text{dictator } F \ i$
<proof>

lemma *const-Lin-Prof*: $L : Lin \implies (\%p. L) : Prof$
<proof>

lemma *complete-Lin*: **assumes** $a \neq b$ **shows** $EX \ L : Lin. (a, b) : L$
<proof>

declare *Let-def*[*simp*]

theorem *Arrow*: **assumes** $F : SWF$ **and** $u : \text{unanimity } F$ **and** $IIA \ F$
shows $EX \ i. \text{dictator } F \ i$
<proof>

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* $f i ::= \forall P \in \text{Prof}. \forall a. a \neq f P \longrightarrow (a, f P) : P i$

definition

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

lemma *Top-in-Lin*: $L:\text{Lin} \Longrightarrow \text{Top } S L : \text{Lin}$
 $\langle \text{proof} \rangle$

lemma *Top-in-Prof*: $P:\text{Prof} \Longrightarrow \text{Top } S o P : \text{Prof}$
 $\langle \text{proof} \rangle$

lemma *not-manipulable*: \neg manipulable $f \longleftrightarrow$
 $(\forall P \in \text{Prof}. \forall i. \forall L \in \text{Lin}. f P \neq f(P(i:=L)) \longrightarrow$
 $(f(P(i:=L)), f P) : P i \wedge (f P, f(P(i:=L))) : L) \text{ (is } ?A = ?B)$
 $\langle \text{proof} \rangle$

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

locale *GS* =

fixes f

assumes *not-manip*: \neg manipulable f

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

begin

lemma *nonmanip*:

$P:\text{Prof} \Longrightarrow L:\text{Lin} \Longrightarrow f(P(i:=L)) \neq f P \Longrightarrow$
 $(f(P(i:=L)), f P) : P i \wedge (f P, f(P(i:=L))) : L$
 $\langle \text{proof} \rangle$

lemma *mono*:

assumes $P \in \text{Prof } P' \in \text{Prof } \forall i a. (a, f P) : P i \longrightarrow (a, f P) : P' i$
shows $f P' = f P$
 $\langle \text{proof} \rangle$

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

lemma *SWF-swf*: $\text{swf } f : \text{SWF}$
 $\langle \text{proof} \rangle$

lemma *Top-top*: $L:\text{Lin} \Longrightarrow (!a. a \neq b \Longrightarrow (a,b) : L) \Longrightarrow \text{Top } \{b\} L = L$
 $\langle \text{proof} \rangle$

lemma *una-swf*: *unanimity*($\text{swf } f$)
 $\langle \text{proof} \rangle$

lemma *IIA-swf*: $IIA(swf\ f)$

<proof>

lemma *dict-swf*: **assumes** *dictator* ($swf\ f$) **i shows** $dict\ f\ i$

<proof>

theorem *Gibbard-Satterthwaite*:

$\exists i. dict\ f\ i$

<proof>

end

theorem *Gibbard-Satterthwaite*:

$\neg\ manipulable\ f \implies \forall a. \exists P \in Prof. a = f\ P \implies \exists i. dict\ f\ i$

<proof>

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.