Arrow and Gibbard-Satterthwaite

Tobias Nipkow

March 17, 2025

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*

axiomatization where
alt3: $\exists\, a\ b\ c{::}alt.\ distinct[a,b,c]$ **and**
finite-alt: $finite(UNIV :: alt\ set)$ **and**

finite-indi: $finite(UNIV :: indi\ set)$

lemma *third-alt*: $a \neq b \Longrightarrow \exists\, c{::}alt.\ distinct[a,b,c]$
⟨*proof*⟩

lemma *alt2*: $\exists\, b{::}alt.\ b \neq a$
⟨*proof*⟩

type-synonym $pref = alt \Rightarrow real$
type-synonym $prof = indi \Rightarrow pref$

definition
top :: *pref* ⇒ *alt* ⇒ *bool* (**infixr** ‹<·› *60*) **where**
$p <\cdot\ b \ \equiv\ \forall a.\ a \neq b \longrightarrow p\ a < p\ b$

definition
bot :: *alt* ⇒ *pref* ⇒ *bool* (**infixr** ‹·<› *60*) **where**
$b\ \cdot\!< p \ \equiv\ \forall a.\ a \neq b \longrightarrow p\ b < p\ a$

definition
extreme :: *pref* ⇒ *alt* ⇒ *bool* **where**
$extreme\ p\ b \ \equiv\ b\ \cdot\!< p \lor p <\cdot\ b$

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

lemma [*simp*]: $r <= s \Longrightarrow r < s+(1::real)$
⟨*proof*⟩
lemma [*simp*]: $r < s \Longrightarrow r < s+(1::real)$
⟨*proof*⟩
lemma [*simp*]: $r <= s \Longrightarrow \neg\ s+(1::real) < r$
⟨*proof*⟩
lemma [*simp*]: $(r < s-(1::real)) = (r+1 < s)$
⟨*proof*⟩
lemma [*simp*]: $(s-(1::real) < r) = (s < r+1)$
⟨*proof*⟩

lemma *less-if-bot*[*simp*]: $[\![\ b\ \cdot\!< p;\ x \neq b\ ]\!] \Longrightarrow p\ b < p\ x$
⟨*proof*⟩

lemma [*simp*]: $[\![\ p <\cdot\ b;\ x \neq b\ ]\!] \Longrightarrow p\ x < p\ b$
⟨*proof*⟩

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

lemma *not-less-if-bot*[*simp*]:
assumes *bot*: $b\ \cdot\!< p$ **shows** $\neg\ p\ c < p\ b$
⟨*proof*⟩

lemma *top-impl-not-bot*[*simp*]: $p <\cdot\ b \Longrightarrow \neg\ b\ \cdot\!< p$
⟨*proof*⟩

lemma [*simp*]: $extreme\ p\ b \Longrightarrow (\neg\ p <\cdot\ b) = (b\ \cdot\!< p)$
⟨*proof*⟩

lemma [*simp*]: $extreme\ p\ b \Longrightarrow (\neg\ b\ \cdot\!< p) = (p <\cdot\ b)$
⟨*proof*⟩

Auxiliary construction to hide details of preference model.

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

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

definition
 between :: *pref* ⇒ *alt* ⇒ *alt* ⇒ *alt* ⇒ *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*]: $a \neq b \Longrightarrow mktop\ p\ b\ a = p\ a$
⟨*proof*⟩

lemma [*simp*]: $a \neq b \Longrightarrow mkbot\ p\ b\ a = p\ a$
⟨*proof*⟩

lemma [*simp*]: $a \neq b \Longrightarrow p\ a < mktop\ p\ b\ b$
⟨*proof*⟩

lemma [*simp*]: $a \neq b \Longrightarrow mkbot\ p\ b\ b < p\ a$
⟨*proof*⟩

lemma [*simp*]: $mktop\ p\ b <\cdot\ b$
⟨*proof*⟩

lemma [*simp*]: $\neg\ b\ \cdot< mktop\ p\ b$
⟨*proof*⟩

lemma [*simp*]: $a \neq b \Longrightarrow \neg\ P\ p\ a < mkbot\ (P\ p)\ b\ b$
⟨*proof*⟩

The proof starts here.

locale *arrow* =
fixes *F* :: *prof* ⇒ *pref*
assumes *unanimity*: $(\bigwedge i.\ P\ i\ a < P\ i\ b) \Longrightarrow F\ P\ a < F\ P\ b$
and *IIA*:
$(\bigwedge i.\ (P\ i\ a < P\ i\ b) = (P'\ i\ a < P'\ i\ b)) \Longrightarrow$
 $(F\ P\ a < F\ P\ b) = (F\ P'\ a < F\ P'\ b)$
begin

lemmas $IIA' = IIA[THEN\ iffD1]$

definition
 dictates :: *indi* ⇒ *alt* ⇒ *alt* ⇒ *bool* (‹- *dictates* - < -›) **where**

$(i \ dictates \ a < b) \equiv \forall P. \ P \ i \ a < P \ i \ b \longrightarrow F \ P \ a < F \ P \ b$
definition
dictates2 :: *indi* ⇒ *alt* ⇒ *alt* ⇒ *bool* (‹- *dictates* -,-›) **where**
$(i \ dictates \ a,b) \equiv (i \ dictates \ a < b) \land (i \ dictates \ b < a)$
definition
dictatesx:: *indi* ⇒ *alt* ⇒ *bool* (‹- *dictates'-except* -›) **where**
$(i \ dictates\text{-}except \ c) \equiv \forall a \ b. \ c \notin \{a,b\} \longrightarrow (i \ dictates \ a<b)$
definition
dictator :: *indi* ⇒ *bool* **where**
$dictator \ i \equiv \forall a \ b. \ (i \ dictates \ a<b)$

definition
pivotal :: *indi* ⇒ *alt* ⇒ *bool* **where**
$pivotal \ i \ b \equiv$
$\exists P. \ Extreme \ P \ b \ \land \ b \cdot< P \ i \ \land \ b \cdot< F \ P \ \land$
$F \ (P(i := mktop \ (P \ i) \ b)) <\cdot \ b$

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

lemma *not-extreme*:
assumes *nex*: $\neg \ extreme \ p \ b$
shows $\exists a \ c. \ distinct[a,b,c] \land \neg \ p \ a < p \ b \land \neg \ p \ b < p \ c$
⟨*proof*⟩

lemma *extremal*:
assumes *extremes*: *Extreme P b* **shows** *extreme* (*F P*) *b*
⟨*proof*⟩

lemma *pivotal-ind*: **assumes** *fin*: *finite D*
shows $\bigwedge P. \ [\![\ D = \{i. \ b \cdot< P \ i\}; \ Extreme \ P \ b; \ b \cdot< F \ P \ ]\!]$
$\Longrightarrow \exists i. \ pivotal \ i \ b$ (**is** $\bigwedge P. \ ?D \ D \ P \Longrightarrow ?E \ P \Longrightarrow ?B \ P \Longrightarrow$ -)
⟨*proof*⟩

lemma *pivotal-exists*: $\exists i. \ pivotal \ i \ b$
⟨*proof*⟩

lemma *pivotal-xdictates*: **assumes** *pivo*: *pivotal i b*
shows *i dictates-except b*
⟨*proof*⟩

lemma *pivotal-is-dictator*:
assumes *pivo*: *pivotal i b* **and** *ab*: $a \neq b$ **and** *d*: *j dictates a,b*
shows $i = j$
⟨*proof*⟩

theorem *dictator*: $\exists\, i.\ dictator\ i$
⟨*proof*⟩

end

end

2 Arrow's Theorem for Strict Linear Orders

theory *Arrow-Order* **imports** *Main HOL−Library.FuncSet*
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 $I == (UNIV{::}indi\ set)$

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

abbreviation $N == card\ I$

lemma *third-alt*: $a \neq b \Longrightarrow \exists\, c{::}alt.\ distinct[a,b,c]$
⟨*proof*⟩

lemma *alt2*: $\exists\, b{::}alt.\ b \neq a$
⟨*proof*⟩

type-synonym *pref* = $(alt * alt)set$

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

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

lemma *notin-Lin-iff*: $L : Lin \Longrightarrow x{\neq}y \Longrightarrow (x,y) \notin L \longleftrightarrow (y,x) : L$
⟨*proof*⟩

lemma *converse-in-Lin*[*simp*]: $L^{-1} : Lin \longleftrightarrow L : Lin$
⟨*proof*⟩

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

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

abbreviation
rem :: *pref* ⇒ *alt* ⇒ *pref* **where**
$rem\ L\ a \equiv \{(x,y).\ (x,y) \in L \wedge x{\neq}a \wedge y{\neq}a\}$
definition
mktop :: *pref* ⇒ *alt* ⇒ *pref* **where**
$mktop\ L\ b \equiv rem\ L\ b \cup \{(x,b)|x.\ x{\neq}b\}$
definition
mkbot :: *pref* ⇒ *alt* ⇒ *pref* **where**
$mkbot\ L\ b \equiv rem\ L\ b \cup \{(b,y)|y.\ y{\neq}b\}$
definition
below :: *pref* ⇒ *alt* ⇒ *alt* ⇒ *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* ⇒ *alt* ⇒ *alt* ⇒ *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*: $a{\neq}b \Longrightarrow L{:}Lin \Longrightarrow$
$(x,y) : above\ L\ a\ b \longleftrightarrow x{\neq}y \wedge$
$(if\ x{=}b\ then\ (a,y) : L\ else$
$if\ y{=}b\ then\ x{=}a \mid (x,a) : L\ else\ (x,y) : L)$
⟨*proof*⟩

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

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

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

lemma *below-Lin*: $x \neq y \Longrightarrow L : Lin \Longrightarrow below\ L\ x\ y : Lin$
⟨*proof*⟩

lemma *above-Lin*: $x \neq y \Longrightarrow L : Lin \Longrightarrow above\ L\ x\ y : Lin$
⟨*proof*⟩

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

abbreviation *lessLin* :: *alt* ⇒ *pref* ⇒ *alt* ⇒ *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*: $F : SWF \Longrightarrow$
$\forall P \in Prof.\ \forall a\ b.\ a \neq b \longrightarrow (a,b) : P\ i \longrightarrow (a,b) : F\ P \Longrightarrow dictator\ F\ i$
⟨*proof*⟩

lemma *const-Lin-Prof*: $L{:}Lin \Longrightarrow (\%p.\ L) : Prof$
⟨*proof*⟩

lemma *complete-Lin*: **assumes** $a \neq b$ **shows** $\exists L \in Lin.\ (a,b) : L$
⟨*proof*⟩

declare *Let-def*[*simp*]

theorem *Arrow*: **assumes** $F : SWF$ **and** *u*: *unanimity F* **and** *IIA F*
shows $\exists i.\ dictator\ F\ i$
⟨*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 $manipulable\ f == \exists P \in Prof.\ \exists i.\ \exists L \in Lin.\ (f\ P,\ f(P(i{:=}L))) : P\ i$

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-in-Lin*: $L{:}Lin \Longrightarrow Top\ S\ L : Lin$
⟨*proof*⟩

lemma *Top-in-Prof*: $P{:}Prof \Longrightarrow Top\ S\ o\ P : Prof$
⟨*proof*⟩

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

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-manip*: $\neg$ *manipulable f*
and *onto*: *f* ‘ *Prof* = *UNIV*
begin

lemma *nonmanip*:
$P{:}Prof \Longrightarrow L{:}Lin \Longrightarrow f(P(i := L)) \neq f\ P \Longrightarrow$
$(f(P(i := L)), f\ P) : P\ i \wedge (f\ P, f(P(i := L))) : L$
⟨*proof*⟩

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*⟩

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

lemma *SWF-swf*: *swf f* : *SWF*
⟨*proof*⟩

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

lemma *una-swf*: *unanimity*(*swf f*)
⟨*proof*⟩

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

$\langle proof \rangle$

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

theorem *Gibbard-Satterthwaite*:
$\exists\, i.\ dict\ f\ i$
$\langle proof \rangle$

end

theorem *Gibbard-Satterthwaite*:
$\neg\ manipulable\ f \Longrightarrow \forall\, a. \exists\, P \in Prof.\ a = f\ P \Longrightarrow \exists\, i.\ dict\ f\ i$
$\langle proof \rangle$

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.