Arrow’s General Possibility Theorem

Peter Gammie
peteg42 at gmail.com

September 19, 2015

Contents

1 Overview 2

2 General Lemmas 2
  2.1 Extra Finite-Set Lemmas ........................................... 2
  2.2 Extra bijection lemmas ............................................. 3
  2.3 Collections of witnesses: hasw, has .............................. 5

3 Preliminaries 8
  3.1 Rational Preference Relations (RPRs) ............................... 9
  3.2 Profiles .............................................................. 11
  3.3 Choice Sets, Choice Functions ...................................... 11
  3.4 Social Choice Functions (SCFs) .................................... 13
  3.5 Social Welfare Functions (SWFs) .................................. 13
  3.6 General Properties of an SCF ...................................... 14
  3.7 Decisiveness and Semi-decisiveness ................................. 15

4 Arrow’s General Possibility Theorem 16
  4.1 Semi-decisiveness Implies Decisiveness .......................... 16
  4.2 The Existence of a Semi-decisive Individual ....................... 23
  4.3 Arrow’s General Possibility Theorem .............................. 27

5 Sen’s Liberal Paradox 27
  5.1 Social Decision Functions (SDFs) .................................. 27
  5.2 Sen’s Liberal Paradox ................................................ 30

6 May’s Theorem 35
  6.1 May’s Conditions .................................................... 35
  6.2 The Method of Majority Decision satisfies May’s conditions .... 37
  6.3 Everything satisfying May’s conditions is the Method of Majority Decision ... 39
  6.4 The Plurality Rule ................................................... 45

7 Bibliography 46
1 Overview

This is a fairly literal encoding of some of Armatya Sen’s proofs [Sen70] in Isabelle/HOL. The author initially wrote it while learning to use the proof assistant, and some locutions remain naïve. This work is somewhat complementary to the mechanisation of more recent proofs of Arrow’s Theorem and the Gibbard-Satterthwaite Theorem by Tobias Nipkow [Nip08].

I strongly recommend Sen’s book to anyone interested in social choice theory; his proofs are quite lucid and accessible, and he situates the theory quite well within the broader economic tradition.

2 General Lemmas

2.1 Extra Finite-Set Lemmas

Small variant of _Finite-Set_.finite-subset-induct: also assume \( F \subseteq A \) in the induction hypothesis.

**lemma** finite-subset-induct’ [consumes 2, case-names empty insert]:

assumes finite \( F \) and \( F \subseteq A \)
and empty: \( P \{\} \)
and insert: \( \forall a. \ (\text{finite } F; a \in A; F \subseteq A; a \notin F; P F) \implies P (\text{insert } a F) \)

shows \( P F \)

**proof** –
from (finite \( F \))
have \( F \subseteq A \implies \text{thesis} \)
**proof** _induct_
  show \( P \{\} \) by _fact_
next
fix \( x F \)
assume finite \( F \) and \( x \notin F \) and
\( P; F \subseteq A \implies P F \) and \( i; \text{insert } x F \subseteq A \)
show \( P (\text{insert } x F) \)
**proof** (rule insert)
  from \( i \) show \( x \in A \) by _blast_
  from \( i \) have \( F \subseteq A \) by _blast_
  with \( P \) show \( P F \).
  show finite \( F \) by _fact_
  show \( x \notin F \) by _fact_
  show \( F \subseteq A \) by _fact_
  qed
  with \( i; F \subseteq A \) show \( \text{thesis} \) by _blast_
  qed

A slight improvement on _List_.finite-list - add _distinct_.

**lemma** finite-list: finite \( A \) \( \implies \exists l. \) set \( l = A \land \text{distinct } l \)
**proof** (induct rule: finite-induct)
  case (insert \( x F \))
  then obtain \( l \) where set \( l = F \land \text{distinct } l \) by _auto_
  with insert have set \((x\#l) = \text{insert } x F \land \text{distinct } (x\#l) \) by _auto_
thus ?case by blast 
qed auto

2.2 Extra bijection lemmas

lemma bij-betw-onto: bij-betw f A B \implies f ' A = B unfolding bij-betw-def by simp

lemma inj-on-UnI: [ inj-on f A; inj-on f B; f ' (A - B) \cap f ' (B - A) = {} ] \implies inj-on f (A \cup B) 
by (auto iff: inj-on-Un)

lemma card-compose-bij: 
assumes bijf: bij-betw f A A 
shows card { a \in A. P (f a) } = card { a \in A. P a } 
proof - 
from bijf have T: f ' { a \in A. P (f a) } = { a \in A. P a } 
unfolding bij-betw-def by simp 
from bijf have inj-on f (A \cup B) 
proof -(auto iff: subset-inj-on card-image[ symmetric ])
qed

lemma card-eq-bij: 
assumes cardAB: card A = card B 
and finiteA: finite A and finiteB: finite B 
obtains f where bij-betw f A B 
proof - 
let ?f = \lambda x. if x \in A then f x else g x 
have inj-on f A 
proof (rule inj-on-UnI)
from bijf show inj-on ?f A 
by -(rule inj-on-UnI, auto dest: inj-onD bij-betw-imp-inj-on)

lemma bij-combine: 
assumes ABCD: A \subseteq B C \subseteq D 
and bijf: bij-betw f A C 
and bijg: bij-betw g (B - A) (D - C) 
obtains h 
where bij-betw h B D 
and \forall x. x \in A \implies h x = f x 
and \forall x. x \in B - A \implies h x = g x 
proof - 
let ?h = \lambda x. if x \in A then f x else g x 
have inj-on ?h (A \cup (B - A))
proof (rule inj-on-UnI)
from bijf show inj-on ?h A 
by -(rule inj-on-UnI, auto dest: inj-onD bij-betw-imp-inj-on)
from bijg show inj-on ?h (B - A)
  by -(rule inj-onI, auto dest: inj-onD bij-betw-imp-inj-on)
from bijf bijg show ?h ' (A - (B - A)) \cap ?h ' (B - A - A) = {}
  by (simp, blast dest: bij-betw-onto)
qed

with ABCD have inj-on ?h B by (auto iff: Un-absorb1)
moreover
have ?h ' B = D
proof -
  from ABCD have ?h ' B = f ' A \cup g ' (B - A) by (auto iff: image-Un Un-absorb1)
  also from ABCD bijf bijg have ... = D by (blast dest: bij-betw-onto)
finally show thesis.
qed

with ABf CD have inj-on ?h B D
by -(rule inj-onI, auto)

lemma bij-complete:
  assumes finiteC: finite C
             and ABC: A \subseteq C B \subseteq C
             and bijf: bij-betw f A B
  obtains f' where bij-betw f' C C
                and \( \forall x. x \in A \Rightarrow f' x = f x \)
                and \( \forall x. x \in B - A \Rightarrow f' x = g x \)
proof -
  from finiteC ABC bijf have card B = card A
    unfolding bij-betw-def
    by (auto iff: inj-on-iff-eq-card [symmetric] intro: finite-subset)
  with finiteC ABC bijf have card (C - A) = card (C - B)
    by (auto iff: finite-subset card-Diff-subset)
  with finiteC obtain g where bijg: bij-betw g (C - A) (C - B)
    by -(drule card-eq-bij, auto)
  from ABC bijf bijg obtain f' where bijf': bij-betw f' C C
      and f'f: \( \forall x. x \in A \Rightarrow f' f x = f x \)
      and f'g: \( \forall x. x \in C - A \Rightarrow f' x = g x \)
    by -(drule bij-combine, auto)
  from f'g bijg have \( \forall x. x \in C - A \Rightarrow f' x \in C - B \)
    by (blast dest: bij-betw-onto)
  with bijf' f'f show thesis.
qed

lemma card-greater:
  assumes finiteA: finite A
             and c: card \{ x \in A. P x \} > card \{ x \in A. Q x \}
  obtains C
    where card \{ x \in A. P x \} - C) = card \{ x \in A. Q x \}
    and C \neq {}
    and C \subseteq \{ x \in A. P x \}
proof -
let \( ?PA = \{ x \in A . P x \} \)
let \( ?QA = \{ x \in A . Q x \} \)

from \( \text{finite} A \) obtain \( p \) where \( P : \text{bij-betw} \ p \ \{ 0 .. < \text{card} \ ?PA \} \ ?PA \)
using \( \text{ex-bij-betw-nat-finite} [\text{where} \ M = ?PA] \)
by (\( \text{blast intro: finite-subset} \))

let \( ?CN = \{ \text{card} \ ?QA .. < \text{card} \ ?PA \} \)
let \( ?C = p ' \ ?CN \)

have \( \text{card} (\{ x \in A . P x \} - ?C) = \text{card} ?QA \)
proof
−
have \( \text{nat-add-sub-shuffle: } \land x y z. [ (x::nat) > y; x - y = z ] \implies x - z = y \) by simp
from \( P \) have \( T: p ' \ \{ \text{card} \ ?QA .. < \text{card} \ ?PA \} \subseteq ?PA \)
unfolding \( \text{bij-betw-def} \) by auto
from \( P \) have \( \text{card} ?PA - \text{card} ?QA = \text{card} ?C \)
unfolding \( \text{bij-betw-def} \)
by (auto iff: \( \text{card-image subset-inj-on} [\text{where} \ A = ?CN] \))
with \( c \) have \( \text{card} ?PA - \text{card} ?C = \text{card} ?QA \) by (rule \( \text{nat-add-sub-shuffle} \))
with \( \text{finite} A \ P \ T \) have \( \text{card} (?PA - ?C) = \text{card} ?QA \)
unfolding \( \text{bij-betw-def} \) by (auto iff: \( \text{finite-subset card-Diff-subset} \))
thus \( ?\text{thesis} \).

qed
moreover
from \( P \ c \) have \( ?C \neq \{ \} \)
unfolding \( \text{bij-betw-def} \) by auto
moreover
from \( P \) have \( ?C \subseteq \{ x \in A . P x \} \)
unfolding \( \text{bij-betw-def} \) by auto
ultimately show \( \text{thesis} \) ..

qed

2.3 Collections of witnesses: hasw, has

Given a set of cardinality at least \( n \), we can find up to \( n \) distinct witnesses. The built-in \( \text{card} \) function unfortunately satisfies:

\[
\text{Finite-Set.card-infinite: } \neg \text{finite} A \implies \text{card} A = 0
\]

These lemmas handle the infinite case uniformly.

Thanks to Gerwin Klein suggesting this approach.

definition hasw :: 'a list \( \Rightarrow \) 'a set \( \Rightarrow \) bool where
hasw xs S \( \equiv \) set xs \( \subseteq \) S \( \land \) distinct xs

definition has :: nat \( \Rightarrow \) 'a set \( \Rightarrow \) bool where
has n S \( \equiv \) \( \exists \) xs. hasw xs S \( \land \) length xs = n

declare hasw-def[simp]

lemma hasI[intro]: hasw xs S \( \implies \) has (length xs) S by (unfold hasw-def, auto)

lemma card-has:
assumes cardS: \( \text{card} S = n \)
shows \( \text{has} n S \)
proof(cases \( n = 0 \))
case True thus ?thesis by (simp add: has-def) 
next
  case False with cardS card-eq-0-iff[where A=S] have finiteS: finite S by simp
  show ?thesis
  proof (rule ccontr)
    assume nhas: \neg has n S
    with distinct-card[symmetric]
    have nxs: \neg (\exists xs. set xs \subseteq S \land \text{distinct} \; xs \land \text{card} \; (\text{set} \; xs) = n)
      by (auto simp add: has-def)
    from finite-list finiteS obtain xs where S = set xs by blast
    with cardS nxs show False by auto
  qed
qed

lemma card-has-rev:
  assumes finiteS: finite S
  shows has n S \implies \text{card} S \geq n
proof -
  assume \?lhs
  then obtain xs where \text{set} \; xs \subseteq S \land \text{n} = \text{length} \; xs
    and dxs: \text{distinct} \; xs by (unfold has-def hasw-def, blast)
  with card-mono[OF finiteS, distinct-card[OF dxs, symmetric]]
  show \?rhs by simp
qed

lemma has-0: has 0 S by (simp add: has-def)

lemma has-suc-notempty: has (Suc n) S \implies \{\} \neq S
  by (clarsimp simp add: has-def)

lemma has-suc-subset: has (Suc n) S \implies \{\} \subseteq S
  by (rule psubsetI, (simp add: has-suc-notempty)+)

lemma has-notempty-1:
  assumes Sne: S \neq \{\}
  shows has 1 S
proof -
  from Sne obtain x where x \in S by blast
  hence set [x] \subseteq S \land \text{distinct} \; [x] \land \text{length} \; [x] = 1 by auto
  thus ?thesis by (unfold has-def hasw-def, blast)
qed

lemma has-le-has:
  assumes h: has n S
    and nn': \text{n'} \leq n
  shows has n' S
proof -
  from h obtain xs where hasw xs S length xs = n by (unfold has-def, blast)
  with nn' set-take-subset[where n=n' and xs=xs]
  have hasw (take n' xs) S length (take n' xs) = n'
by (simp-all add: min-def, blast+)
thus ?thesis by (unfold has-def, blast)
qed

lemma has-ge-has-not:
assumes h: ¬has n S
and nn’: n ≤ n’
shows ¬has n’ S
using h nn’ by (blast dest: has-le-has)

lemma has-eq:
assumes h: has n S
and hn: ¬has (Suc n) S
shows card S = n
proof -
  from h obtain xs
  where xs: hasw xs S and lenxs: length xs = n by (unfold has-def, blast)
  have set xs = S
  proof
    from xs show set xs ⊆ S by simp
  next
    show S ⊆ set xs
  proof (rule ccontr)
    assume ¬S ⊆ set xs
    then obtain x where x ∈ S x /∈ set xs by blast
    with lenxs xs have hasw (x # xs) S length (x # xs) = Suc n by simp-all
    with hn show False by (unfold has-def, blast)
  qed
  qed
  with xs lenxs distinct-card show card S = n by auto
qed

lemma has-extend-witness:
assumes h: has n S
shows [ set xs ⊆ S; length xs < n ] ⇒ set xs ⊂ S
proof (induct xs)
  case Nil
  with h has-suc-notempty show ?case by (cases n, auto)
next
  case (Cons x xs)
  have set (x # xs) ≠ S
  proof
    assume Sxs: set (x # xs) = S
    hence finiteS: finite S by auto
    from h obtain xs’
      where Sxs’: set xs’ ⊆ S
      and dlds: distinct xs’ ∧ length xs’ = n
      by (unfold has-def hasw-def, blast)
    with distinct-card have card (set xs’) = n by auto
    with finiteS Sxs’ card mono have card S ≥ n by auto
    moreover
    from Sxs Cons card-length[where xs=x # xs]
    have card S < n by auto
  qed
ultimately show False by simp
qed
with Cons show ?case by auto
qed

lemma has-extend-witness':

\[
\begin{array}{l}
\text{has n S;} \text{ hasw xs S;} \text{ length xs < n } \\
\end{array}
\implies \exists x. \text{ hasw (x # xs) S}
\]

by (simp, blast dest: has-extend-witness)

lemma has-witness-two:

assumes hasnS: has n S
and nn': 2 ≤ n
shows \( \exists x y. \text{ hasw [x,y] S} \)

proof –

have has2S: has 2 S by (rule has-le-has [OF hasnS nn'])
from has-extend-witness [OF has2S, where xs=[]] obtai
x where \( x \in S \) by auto
with has-extend-witness [OF has2S, where xs=[x]]
show ?thesis by auto
qed

lemma has-witness-three:

assumes hasnS: has n S
and nn': 3 ≤ n
shows \( \exists x y z. \text{ hasw [x,y,z] S} \)

proof –

from nn' obtain x y where hasw [x,y] S
using has-witness-two [OF hasnS] by auto
with nn' show ?thesis
using has-extend-witness [OF hasnS, where xs=[x,y]] by auto
qed

lemma finite-set-singleton-contra:

assumes finiteS: finite S
and Sne: S \neq {} 
and cardS: card S > 1 \implies False
shows \( \exists j. S = \{j\} \)

proof –

from cardS Sne card-0-eq [OF finiteS] have Scard: card S = 1 by auto
from has-extend-witness [where xs=[], OF card-has [OF this]]
obtain j where \( \{j\} \subseteq S \) by auto
from card-seteq [OF finiteS this] Scard show ?thesis by auto
qed

3 Preliminaries

The auxiliary concepts defined here are standard [Rou79, Sen70, Tay05]. Throughout we make use of a fixed set A of alternatives, drawn from some arbitrary type 'a of suitable size. Taylor [Tay05] terms this set an agenda. Similarly we have a type 'i of individuals and a
3.1 Rational Preference Relations (RPRs)

Definitions for rational preference relations (RPRs), which represent indifference or strict preference amongst some set of alternatives. These are also called weak orders or (ambiguously) ballots.

Unfortunately Isabelle’s standard ordering operators and lemmas are typeclass-based, and as introducing new types is painful and we need several orders per type, we need to repeat some things.

type-synonym 'a RPR = ('a * 'a) set

abbreviation rpr-eq-syntax :: 'a ⇒ 'a RPR ⇒ 'a ⇒ bool (- - ≤ - [50, 1000, 51] 50) where
  x r ⪯ y == (x, y) ∈ r

definition indifferent-pref :: 'a ⇒ 'a RPR ⇒ 'a ⇒ bool (- - ≈ - [50, 1000, 51] 50) where
  x r ≈ y ≡ (x r ⪯ y ∧ y r ⪯ x)

lemma indifferent-prefI [intro]: [ [ x r ⪯ y; y r ⪯ x ] ] ⇒ x r ≈ y

unfolding indifferent-pref-def by simp

lemma indifferent-prefD [dest]: x r ≈ y ⇒ x r ⪯ y ∧ y r ⪯ x

unfolding indifferent-pref-def by simp

definition strict-pref :: 'a ⇒ 'a RPR ⇒ 'a ⇒ bool (- - ≺ - [50, 1000, 51] 50) where
  x r ≺ y ≡ (x r ⪯ y ∧ ¬(y r ⪯ x))

lemma strict-pref-I [intro]: [ [ x r ⪯ y; ¬(y r ⪯ x) ] ] ⇒ x r ≺ y

unfolding strict-pref-def by simp

Traditionally, x r ⪯ y would be written x R y, x r ≈ y as x I y and x r ≺ y as x P y, where the relation r is implicit, and profiles are indexed by subscripting.

Complete means that every pair of distinct alternatives is ranked. The ”distinct” part is a matter of taste, as it makes sense to regard an alternative as as good as itself. Here I take reflexivity separately.

definition complete :: 'a set ⇒ 'a RPR ⇒ bool where
  complete A r ≡ (∀x ∈ A. ∀y ∈ A − {x}. x r ⪯ y ∨ y r ⪯ x)

lemma completeI [intro]:
  (∀x y. [ x ∈ A; y ∈ A; x ≠ y ] ⇒ x r ⪯ y ∨ y r ⪯ x) ⇒ complete A r

unfolding complete-def by auto

lemma completeD [dest]:
  [ [ complete A r; x ∈ A; y ∈ A; x ≠ y ] ⇒ x r ⪯ y ∨ y r ⪯ x ] ⇒ complete A r

unfolding complete-def by auto

lemma complete-less-not: [ [ complete A r; hasw [x,y] A; ¬ x r ≺ y ] ] ⇒ y r ⪯ x

unfolding complete-def strict-pref-def by auto
lemma complete-indiff-not: \[ \text{complete } A \; r; \; \text{hasw } [x,y] \; A; \; \neg \; x \; r \approx y \] \implies x \prec y \lor y \prec x

unfolding complete-def indifferent-pref-def strict-pref-def by auto

lemma complete-exh: assumes complete A r and hasw [x,y] A obtains (xPy) x r \prec y | (yPx) y r \prec x | (xIy) x r \approx y

using assms unfolding complete-def strict-pref-def indifferent-pref-def by auto

Use the standard refl. Also define irreflexivity analogously to how refl is defined in the standard library.

declare refl-onI[intro] refl-onD[dest]

lemma complete-refl-on: \[ \text{complete } A \; r; \; \text{refl-on } A \; r; \; x \in A; \; y \in A \] \implies x r \preceq y \lor y r \preceq x

unfolding complete-def by auto

definition irrefl :: 'a set \Rightarrow 'a RPR \Rightarrow bool where
irrefl A r \equiv r \subseteq A \times A \land (\forall x \in A. \neg x \preceq x)

lemma irreflI[intro]: \[ r \subseteq A \times A; \; \forall x \in A \implies \neg x \preceq x \] \implies irrefl A r

unfolding irrefl-def by simp

lemma irreflD[dest]: \[ irrefl A r; \; (x, y) \in r \] \implies hasw [x,y] A

unfolding irrefl-def by auto

lemma irreflD'[dest]: \[ irrefl A r; \; r \neq \{\} \] \implies \exists x \; y. \; hasw [x,y] A \land (x, y) \in r

unfolding irrefl-def by auto

Rational preference relations, also known as weak orders and (I guess) complete pre-orders.

definition rpr :: 'a set \Rightarrow 'a RPR \Rightarrow bool where
rpr A r \equiv \exists A \; r \subseteq A \times A \land (\forall x \in A. \neg x \preceq x)

lemma rprI[intro]: \[ \text{complete } A \; r; \; \text{refl-on } A \; r; \; \text{trans } r \] \implies rpr A r

unfolding rpr-def by simp

lemma rprD: rpr A r \implies \text{complete } A \; r \land \text{refl-on } A \; r \land \text{trans } r

unfolding rpr-def by simp

lemma rpr-in-set[dest]: \[ rpr A r; \; x r \preceq y \] \implies \{x,y\} \subseteq A

unfolding rpr-def refl-on-def by auto

lemma rpr-refl[dest]: \[ rpr A r; \; x \in A \] \implies x r \preceq x

unfolding rpr-def by blast

lemma rpr-less-not: \[ rpr A r; \; \text{hasw } [x,y] \; A; \; \neg x r \prec y \] \implies y r \preceq x

unfolding rpr-def by (auto simp add: complete-less-not)

lemma rpr-less-imp-le[simp]: \[ x r \prec y \] \implies x r \preceq y
unfolding strict-pref-def by simp

lemma rpr-less-imp-neq[simp]: \( x \prec y \implies x \neq y \)
unfolding strict-pref-def by blast

lemma rpr-less-trans[trans]: \( x \prec y; y \prec z; rpr \ A \ r \ \implies x \prec z \)
unfolding rpr-def strict-pref-def trans-def by blast

lemma rpr-le-trans[trans]: \( x \preceq y; y \preceq z; rpr \ A \ r \ \implies x \preceq z \)
unfolding rpr-def trans-def by blast

lemma rpr-le-less-trans[trans]: \( x \preceq y; y \prec z; rpr \ A \ r \ \implies x \prec z \)
unfolding rpr-def strict-pref-def trans-def by blast

lemma rpr-less-le-trans[trans]: \( x \prec y; y \preceq z; rpr \ A \ r \ \implies x \prec z \)
unfolding rpr-def strict-pref-def trans-def by blast

lemma rpr-complete: \( rpr \ A \ r; x \in A; y \in A \ \implies x \preceq y \lor y \preceq x \)
unfolding rpr-def by (blast dest: complete-refl-on)

3.2 Profiles

A profile (also termed a collection of ballots) maps each individual to an RPR for that individual.

type-synonym ('a, 'i) Profile = 'i => 'a RPR

definition profile :: 'a set => 'i set => ('a, 'i) Profile => bool where
profile A Is P \equiv Is \neq {} \land (\forall i \in Is. rpr A (P i))

lemma profileI[intro]: \( \forall i \in Is \implies rpr A (P i); Is \neq {} \] \implies profile A Is P
unfolding profile-def by simp

lemma profile-rprD[dest]: [ profile A Is P; i \in Is ] \implies rpr A (P i)
unfolding profile-def by simp

lemma profile-non-empty: profile A Is P \implies Is \neq {}
unfolding profile-def by simp

3.3 Choice Sets, Choice Functions

A choice set is the subset of A where every element of that subset is (weakly) preferred to every other element of A with respect to a given RPR. A choice function yields a non-empty choice set whenever A is non-empty.

definition choiceSet :: 'a set => 'a RPR => 'a set where
choiceSet A r \equiv \{ x \in A . \forall y \in A. x \preceq y \}

definition choiceFn :: 'a set => 'a RPR => bool where
choiceFn A r \equiv \forall A' \subseteq A. A' \neq {} \implies choiceSet A' r \neq {}
lemma choiceSetI[intro]:
\[ x \in A; \forall y. y \in A \implies x \preceq y \implies x \in \text{choiceSet } A r \]
unfolding choiceSet-def by simp

lemma choiceFnI[intro]:
\[ (\forall A'. A' \subseteq A; A' \neq \emptyset) \implies \text{choiceSet } A' r \neq \emptyset \implies \text{choiceFn } A r \]
unfolding choiceFn-def by simp

If a complete and reflexive relation is also quasi-transitive it will yield a choice function.

definition quasi-trans :: 'a RPR \Rightarrow bool where
quasi-trans r \equiv \forall x y z. x \prec y \land y \prec z \rightarrow x \prec z

lemma quasi-transI[intro]:
\[ (\forall x y z. [ x \prec y; y \prec z ] \implies x \prec z) \implies \text{quasi-trans } r \]
unfolding quasi-trans-def by blast

lemma quasi-transD: [ x \prec y; y \prec z; quasi-trans r ] \implies x \prec z
unfolding quasi-trans-def by blast

lemma trans-imp-quasi-trans: trans r \implies quasi-trans r
by (rule quasi-transI, unfold strict-pref-def trans-def, blast)

lemma r-c-qt-imp-cf:  
assumes finiteA: finite A 
and c: complete A r 
and qt: quasi-trans r 
and r: refl-on A r 
shows choiceFn A r 
proof 
fix B assume B: B \subseteq A B \neq \emptyset 
with finite-subset finiteA have finiteB: finite B by auto 
from finiteB B show choiceSet B r \neq \emptyset 
proof(induct rule: finite-subset-induct')
case empty with B show \?case by auto
next
case (insert a B) 
hence finiteB: finite B 
and aA: a \in A 
and AB: B \subseteq A 
and aB: a \notin B 
and cF: B \neq \emptyset \implies \text{choiceSet } B r \neq \emptyset by - blast 
show \?case 
proof(cases B = \{} 
  case True with aA r show \?thesis 
  unfolding choiceSet-def by blast
  next 
  case False 
  with cF obtain b where bCF: b \in \text{choiceSet } B r by blast 
  from AB aA bCF complete-refl-on[OF c r] 
  have a \prec b \lor b \preceq a unfolding choiceSet-def strict-pref-def by blast 
  thus \?thesis 
  proof 
  assume ab: b \preceq a

12
with bCF show ?thesis unfolding choiceSet-def by auto
next
assume ab: a r≺ b
have a ∈ choiceSet (insert a B) r
proof (rule ccontr)
  assume aCF: a ∉ choiceSet (insert a B) r
  from aB have ∨ b. b ∈ B ⇒ a ≠ b by auto
  with aCF aA AB c r obtain b' where B: b' ∈ B b' r≺ a
  unfolding choiceSet-def complete-def strict-pref-def by blast
  with ab qf have b' r≺ b by (blast dest: quasi-transD)
  with bCF B show False unfolding choiceSet-def strict-pref-def by blast
qed
thus ?thesis by auto
qed
qed
qed
lemma rpr-choiceFn: [ finite A ; rpr A r ] ⇒ choiceFn A r
  unfolding rpr-def by (blast dest: trans-imp-quasi-trans r-c-qt-imp-cf)

3.4 Social Choice Functions (SCFs)
A social choice function (SCF), also called a collective choice rule by Sen [Sen70, p28], is a function that somehow aggregates society’s opinions, expressed as a profile, into a preference relation.
type-synonym ('a, 'i) SCF = ('a, 'i) Profile ⇒ 'a RPR

The least we require of an SCF is that it be complete and some function of the profile. The latter condition is usually implied by other conditions, such as iia.
definition
SCF :: ('a, 'i) SCF ⇒ 'a set ⇒ 'i set ⇒ ('a set ⇒ 'i set ⇒ ('a, 'i) Profile ⇒ bool) ⇒ bool
where
SCF scf A Is Pcond ≡ (∀ P. Pcond A Is P ⇒ (complete A (scf P)))

lemma SCFI[intro]:
  assumes c: ∨ P. Pcond A Is P ⇒ complete A (scf P)
  shows SCF scf A Is Pcond
  unfolding SCF-def using assms by blast

lemma SCF-completeD[dest]: [ SCF scf A Is Pcond; Pcond A Is P ] ⇒ complete A (scf P)
  unfolding SCF-def by blast

3.5 Social Welfare Functions (SWFs)
A Social Welfare Function (SWF) is an SCF that expresses the society’s opinion as a single RPR.

In some situations it might make sense to restrict the allowable profiles.
definition
SWF :: ('a, 'i) SCF ⇒ 'a set ⇒ 'i set ⇒ ('a set ⇒ 'i set ⇒ ('a, 'i) Profile ⇒ bool) ⇒ bool
where
SWF  swf  A  Is Pcond  ≡  (∀P.  Pcond  A  Is  P  →  rpr  A  (swf  P))

lemma  SWF-rpr[dest]:  [ SWF  swf  A  Is  Pcond;  Pcond  A  Is  P ]  →  rpr  A  (swf  P)
  unfolding  SWF-def  by  simp

3.6  General Properties of an SCF

An SCF has a universal domain if it works for all profiles.

definition  universal-domain :: 'a set ⇒ 'i set ⇒ ('a, 'i) Profile ⇒ bool  where
  universal-domain  A  Is  P  ≡  profile  A  Is  P

declare  universal-domain-def[simp]

  An SCF is weakly Pareto-optimal if, whenever everyone strictly prefers x to y, the SCF does too.

definition  weak-pareto :: ('a, 'i) SCF ⇒ 'a set ⇒ 'i set ⇒ ('a set ⇒ 'i set ⇒ ('a, 'i) Profile ⇒ bool) ⇒ bool  where
  weak-pareto  scf  A  Is  Pcond  ≡  
  (∀P  x  y.  Pcond  A  Is  P  ∧  x  ∈  A  ∧  y  ∈  A  ∧  (∀i  ∈  Is.  x  (P  i)  ≺  y)  →  x  (scf  P)  ≺  y)

lemma  weak-paretoI[intro]:
  (∀P  x  y.  [Pcond  A  Is  P;  x  ∈  A;  y  ∈  A;  ∀i  ∈  Is,  (P  i)  ≺  y]  →  x  (scf  P)  ≺  y)
  ⇒  weak-pareto  scf  A  Is  Pcond
  unfolding  weak-pareto-def  by  simp

lemma  weak-paretoD:
  [ weak-pareto  scf  A  Is  Pcond;  Pcond  A  Is  P;  x  ∈  A;  y  ∈  A;  ∀i  ∈  Is,  x  (P  i)  ≺  y ]  →  x  (scf  P)  ≺  y
  unfolding  weak-pareto-def  by  simp

  An SCF satisfies independence of irrelevant alternatives if, for two preference profiles P and P' where for all individuals i, alternatives x and y drawn from set S have the same order in P i and P' i, then alternatives x and y have the same order in scf P and scf P'.

definition  iia :: ('a, 'i) SCF ⇒ 'a set ⇒ 'i set ⇒ bool  where
  iia  scf  S  Is  ≡  
  (∀P  P'  x  y.  profile  S  Is  P  ∧  profile  S  Is  P'
  ∧  x  ∈  S  ∧  y  ∈  S
  ∧  (∀i  ∈  Is,  ((x  (P  i)  ≺  y)  ↔  (x  (P'  i)  ≺  y))  ∧  (((P  i)  ≺  x)  ↔  ((P'  i)  ≺  x)))
  →  ((x  (scf  P)  ≺  y)  ↔  (x  (scf  P')  ≺  y)))

lemma  iiaI[intro]:
  (∀P  P'  x  y.  profile  S  Is  P;  profile  S  Is  P';
  x  ∈  S;  y  ∈  S;
  ∀i  ∈  Is,  (x  (P  i)  ≺  y)  ↔  (x  (P'  i)  ≺  y))  ∧  (((P  i)  ≺  x)  ↔  ((P'  i)  ≺  x))
  →  ((x  (scf  P)  ≺  y)  ↔  (x  (scf  P')  ≺  y))
  ⇒  iia  scf  S  Is
  unfolding  iia-def  by  simp

lemma  iiaE:
3.7 Decisiveness and Semi-decisiveness

This notion is the key to Arrow’s Theorem, and hinges on the use of strict preference [Sen70, p42].

A coalition $C$ of agents is semi-decisive for $x$ over $y$ if, whenever the coalition prefers $x$ to $y$ and all other agents prefer the converse, the coalition prevails.

**Definition** semi-decisive :: ('a, 'i) SCF $\Rightarrow$ 'a set $\Rightarrow$ 'i set $\Rightarrow$ 'i set $\Rightarrow$ 'a $\Rightarrow$ bool where

semi-decisive $\operatorname{scf} A$ Is $C$ $x$ $y$ $\equiv$

$C \subseteq$ Is $\land$ $\forall$ $P$. profile $A$ Is $P$ $\land$ $\forall$ $i \in C$. $x$ $(P_i) \prec y$ $\land$ $\forall$ $i \in$ Is $- C$. $y$ $(P_i) \prec x$

$\rightarrow$ $x$ $(\operatorname{scf} P) \prec y$

**Lemma** semi-decisiveI[intro]:

$\forall$ $C \subseteq$ Is;

$\forall$ $P$. $\forall$ $i \in C$. $x$ $(P_i) \prec y$ $\forall$ $i \in$ Is $- C$. $y$ $(P_i) \prec x$

$\rightarrow$ $x$ $(\operatorname{scf} P) \prec y$

**Lemma** semi-decisive-coalitionD[dest]: semi-decisive $\operatorname{scf} A$ Is $C$ $x$ $y$ $\Rightarrow$ $C \subseteq$ Is

**Unfolding** semi-decisive-def by simp

**Lemma** sd-refl: $C \subseteq$ Is; $C \neq \{\}$ $\Rightarrow$ semi-decisive $\operatorname{scf} A$ Is $C$ $x$ $x$

**Unfolding** semi-decisive-def strict-pref-def by blast

A coalition $C$ is decisive for $x$ over $y$ if, whenever the coalition prefers $x$ to $y$, the coalition prevails.

**Definition** decisive :: ('a, 'i) SCF $\Rightarrow$ 'a set $\Rightarrow$ 'i set $\Rightarrow$ 'i set $\Rightarrow$ 'a $\Rightarrow$ bool where
decisive $\operatorname{scf} A$ Is $C$ $x$ $y$ $\equiv$

$C \subseteq$ Is $\land$ $\forall$ $P$. profile $A$ Is $P$ $\land$ $\forall$ $i \in C$. $x$ $(P_i) \prec y$ $\rightarrow$ $x$ $(\operatorname{scf} P) \prec y$

**Lemma** decisiveI[intro]:

$\forall$ $C \subseteq$ Is; $\forall$ $P$. $\forall$ $i \in C$. $x$ $(P_i) \prec y$

$\rightarrow$ decisive $\operatorname{scf} A$ Is $C$ $x$ $y$

**Unfolding** decisive-def by simp

**Lemma** d-imp-sd: decisive $\operatorname{scf} A$ Is $C$ $x$ $y$ $\Rightarrow$ semi-decisive $\operatorname{scf} A$ Is $C$ $x$ $y$

**Unfolding** decisive-def by (rule semi-decisiveI, blast+)

**Lemma** decisive-coalitionD[dest]: decisive $\operatorname{scf} A$ Is $C$ $x$ $y$ $\Rightarrow$ $C \subseteq$ Is

**Unfolding** decisive-def by simp

Anyone is trivially decisive for $x$ against $x$.

**Lemma** d-refl: $C \subseteq$ Is; $C \neq \{\}$ $\Rightarrow$ decisive $\operatorname{scf} A$ Is $C$ $x$ $x$
unfolding decisive-def strict-pref-def by simp

Agent $j$ is a dictator if her preferences always prevail. This is the same as saying that she is decisive for all $x$ and $y$.

**Definition**

 dictator :: ('a, 'i) SCF ⇒ 'a set ⇒ 'i set ⇒ 'i ⇒ bool

 dictator scf A Is $j$ ≡ $j \in Is \land (\forall x \in A. \forall y \in A. \text{decisive scf } A \text{ Is } \{j\} \ x y)$

**Lemma**

**dictatorI**: $j \in Is; \land x y. \ x \in A; \ y \in A \implies \text{decisive scf } A \text{ Is } \{j\} \ x y \implies \text{dictator scf } A \text{ Is } j$

unfolding dictator-def by simp

**Lemma**

**dictator-individual**: dictator scf A Is $j$ \implies j \in Is

unfolding dictator-def by simp

### 4 Arrow’s General Possibility Theorem

The proof falls into two parts: showing that a semi-decisive individual is in fact a dictator, and that a semi-decisive individual exists. I take them in that order.

It might be good to do some of this in a locale. The complication is untangling where various witnesses need to be quantified over.

#### 4.1 Semi-decisiveness Implies Decisiveness

I follow [Sen70, Chapter 3*] quite closely here. Formalising his appeal to the **iia** assumption is the main complication here.

The witness for the first lemma: in the profile $P'$, special agent $j$ strictly prefers $x$ to $y$ to $z$, and doesn’t care about the other alternatives. Everyone else strictly prefers $y$ to each of $x$ to $z$, and inherits the relative preferences between $x$ and $z$ from profile $P$.

The model has to be specific about ordering all the other alternatives, but these are immaterial in the proof that uses this witness. Note also that the following lemma is used with different instantiations of $x$, $y$ and $z$, so we need to quantify over them here. This happens implicitly, but in a locale we would have to be more explicit.

This is just tedious.

**Lemma**

**decisive1-witness**: assumes has3A: hasw [x,y,z] A and profileP: profile A Is P and jIs: j \in Is obtains P'

where profile A Is P'

and $x \ (P’ \ j) < y \land y \ (P’ \ j) < z$

and $\land i. \ i \neq j \implies y \ (P’ \ i) < x \land y \ (P’ \ i) < z \land ((x \ (P’ \ i) \leq z) = (x \ (P \ i) \leq z)) \land ((z \ (P’ \ i) \leq x) = (z \ (P \ i) \leq x))$

**Proof**

let $P' = \lambda i. \ (if \ i = j \ then \ \{(x, u) \ | \ u. u \in A\} \cup \{(y, u) \ | \ u. u \in A - \{x\}\}$

16
∪ \{ (z, u) | u. u \in A - \{x,y\}\} \\
else (\{ (y, u) | u. u \in A \} \\
∪ \{ (x, u) | u. u \in A - \{y,z\}\} \\
∪ \{ (z, u) | u. u \in A - \{x,y\}\} \\
∪ (if x (P i) \preceq z then \{(x,z)\} else \}) \\
∪ (if z (P i) \preceq x then \{(z,x)\} else \})\\n∪ (A - \{x,y,z\}) × (A - \{x,y,z\})

show profile A Is \?P′
proof
fix i assume ils: i \in Is
show rpr A (?P′ i)
proof (cases i = j)
case True with has3A show \?thesis
  by - (rule rprI, simp-all add: trans-def, blast+)
next
case False hence ij: i \neq j .
show \?thesis
proof
  from ils profileP have complete A (P i) by (blast dest: rpr-complete)
  with ij show complete A (?P′ i) by (simp add: complete-def, blast)
  from ils profileP have refl-on A (P i) by (auto simp add: rpr-def)
  with has3A ij show refl-on A (?P′ i) by (simp, blast)
  from ij has3A show trans (?P′ i) by (clarsimp simp add: trans-def)
qed
qed
next
  from profileP show Is \neq {} by (rule profile-non-empty)
qed
from has3A
  show x (?P′ j) \preceq y \land y (?P′ j) \preceq z
  and \( \land i. i \neq j \implies y (?P′ i) \preceq x \land y (?P′ i) \preceq z \land ((x (?P′ i) \preceq z) = (x (P i) \preceq z)) \land ((z (?P′ i) \preceq x) = (z (P i) \preceq x))\)
  unfolding strict-pref-def by auto
qed

The key lemma: in the presence of Arrow’s assumptions, an individual who is semi-decisive for x and y is actually decisive for x over any other alternative z. (This is where the quantification becomes important.)

lemma decisive1:
assumes has3A: hasw [x,y,z] A
  and iia: iia swf A Is
  and swf: SWF swf A Is universal-domain
  and wp: weak-pareto swf A Is universal-domain
  and sd: semidecisive swf A Is \{j\} x y
shows decisive swf A Is \{j\} x z
proof
  from sd show jIs: \{j\} \subseteq Is by blast
  fix P
  assume profileP: profile A Is P
  and jzP: \\( \land i. i \in \{j\} \implies x (P i) \preceq z \)
  from has3A profileP jIs
  obtain P′
where $\text{profileP'}: \text{profile} A Is P'$
and $\text{ixyzP'}: x (P', j) < y (P', j) < z$
and $\text{ixyzP':} \forall i. i \neq j \rightarrow (P', i) < x \land (P', i) < z \land ((x (P',i) \leq z) = (x (P,i) \leq z)) \land ((z (P',i) \leq z))$

$(P',i) \leq z) = (z (P,i) \leq z))$

by $- (\text{rule decisive1-witness, blast+})$
from $\text{iaa have} \land \{ a \in \{x, z\}; b \in \{x, z\} \} \implies (a (\text{suf P}) \leq b) = (a (\text{suf P'}) \leq b)$

$\text{proof(\text{rule iiaE})}$
from $\text{has3A show} \{ x, z \} \subseteq A$ by $\text{simp}$

next
fix $i$ assume $iIs: i \in Is$
fix $a \ b$ assume $ab: a \in \{x, z\} \ b \in \{x, z\}$

show $(a (P', i) \leq b) = (a (P, i) \leq b)$

$\text{proof(cases i = j)}$
\hspace{1em} case $\text{False}$
\hspace{2em} with $ab \ iIs \ \text{ixyzP'} \ \text{profileP'} \ \text{profileP'} \ \text{has3A}$

show $\text{thesis unfolding profile-def by auto}$

next
\hspace{1em} case $\text{True}$
from $\text{profileP'} \ jIs \ \text{ixyzP'} \ \text{have} \ x (P', j) < z$
by $\text{(auto dest: rpr-less-trans)}$
with $\text{True ab iIs \ jxyzP \ profileP'} \ \text{profileP'} \ \text{has3A}$

show $\text{thesis unfolding profile-def strict-pref-def by auto}$

qed

qed ($\text{simp-all add: profileP profileP'}$)

moreover have $x (\text{suf P'}) < z$

$\text{proof -}$
from $\text{profileP'} \ \text{sd} \ \text{ixyzP'} \ \text{ixyzP'} \ \text{have} \ x (\text{suf P'}) < y$ by $\text{(simp add: semidecisive-def)}$

moreover
from $\text{ixyzP'} \ \text{ixyzP'} \ \text{have} \ \land i. i \in Is \implies y (P',i) < z$ by $\text{(case-tac i=j, auto)}$
with $\text{wp profileP'} \ \text{has3A} \ \text{have} \ y (\text{suf P'}) < z$ by $\text{(auto dest: weak-paretoD)}$

moreover note $\text{SWF-rpr[of suf] profileP'}$
ultimately show $x (\text{suf P'}) < z$

unfolding $\text{universal-domain-def by (blast dest: rpr-less-trans)}$

qed
ultimately show $x (\text{suf P'}) < z$ unfolding strict-pref-def by blast

qed

The witness for the second lemma: special agent $j$ strictly prefers $z$ to $x$ to $y$, and everyone else strictly prefers $z$ to $x$ and $y$ to $x$. (In some sense the last part is upside-down with respect to the first witness.)

lemma $\text{decisive2-witness:}$
assumes $\text{has3A:} \ \text{hasw} [x, y, z] A$
and $\text{profileP:} \ \text{profile} A Is P$
and $\text{jIs; j} \in Is$

obtains $P'$
where $\text{profile A Is P'}$
and $\text{z} (P', j) < x \land (P', j) < y$
and $\land i. i \neq j \implies z (P',i) < x \land (P',i) < x \land ((y (P',i) \leq z) = (y (P,i) \leq z)) \land ((z (P',i) \leq y) = (z (P,i) \leq y))$

proof
let $?P' = \lambda i. (\text{if } i = j \text{ then } \{(z, u) | u \in A\} \cup \{(x, u) | u \in A - \{z\}\} \cup \{(y, u) | u \in A - \{x, z\}\} \text{ else } \{(z, u) | u \in A - \{y\}\} \cup \{(y, u) | u \in A - \{z\}\} \cup \{(x, u) | u \in A - \{y, z\}\} \cup \{(y, z) \leq z \text{ then } \{(y, z)\} \text{ else } \{\}\})$

show $\text{profile } A \text{ Is } ?P'$

proof

  fix $i$ assume $\text{ils; } i \in \text{Is}$
  show $\text{rpr } A (\text{?P'} i)$
  proof
    cases $i = j$
    case $\text{True with has3A show } ?\text{thesis}$
      by $(\text{rule rpr1, simp-all add: trans-def, blast+})$
  next
    case $\text{False hence } ij: i \neq j$.
    show $\text{?thesis}$
    proof
      from $\text{ils} \text{ profileP have complete } A (P i)$ by $(\text{auto simp add: rpr-def})$
      with $ij$ show $\text{complete } A (\text{?P'} i)$ by $(\text{simp add: complete-def, blast})$
      from $\text{ils} \text{ profileP have refl-on } A (P i)$ by $(\text{auto simp add: rpr-def})$
      with $\text{has3A ij show refl-on } A (\text{?P'} i)$ by $(\text{simp, blast})$
      from $ij$ $\text{has3A show } \text{trans } (\text{?P'} i)$ by $(\text{clarsimp simp add: trans-def})$
    qed
  qed
next
  show $\text{Is} \neq \{\}$ by $(\text{rule profile-non-empty}(\text{OF profileP}))$
  qed

from $\text{has3A}$
show $z (\text{?P'} i) \preceq x \land x \preceq y$
  and $\forall i. i \neq j \Longrightarrow z (\text{?P'} i) \preceq x \land y (\text{?P'} i) \preceq x \land (y (\text{?P'} i) \preceq z) = (y (P i) \preceq z)) \land ((z (\text{?P'} i) \preceq y) = (z (P i) \preceq y))$
  unfolding strict-pref-def by auto

qed

lemma decisive2: 
assumes $\text{has3A: hasw } [x, y, z] \text{ A}$
  and $\text{iia: iia suf } A \text{ Is}$
  and $\text{suf: SWF suf } A \text{ Is universal-domain}$
  and $\text{wp: weak-pareto suf } A \text{ Is universal-domain}$
  and $\text{sd: semideciseive suf } A \text{ Is } \{j\} x y$
shows $\text{decide suf } A \text{ Is } \{j\} z y$

proof

from $\text{sd show } \text{jls: } \{j\} \subseteq \text{Is by blast}$
fix $P$
assume $\text{profileP: profile } A \text{ Is } P$
  and $\text{jyzP: } \forall i. i \in \{j\} \Longrightarrow z (P i) \preceq y$
from $\text{has3A profileP jls}$
obtain $P'$
  where $\text{profileP': profile } A \text{ Is } P'$
  ...
and \( jxyzP' \): \( z (P' j)^x x (P' j)^y \)
and \( jxyzP' \): \( \forall i. i \neq j \rightarrow z (P' j)^x x \wedge (y (P' i)^z z) = (y (P i)^z z) \wedge ((z (P i)^z y)) \)

\[
\text{by } (\text{rule decisive2-witness, blast+})
\]

from \( iia \) have \( \{a b. a \in \{y, z\}; b \in \{y, z\} \} \implies (a (\text{suf } P)^b = (a (\text{suf } P)^b) \)

proof\( (\text{rule iiaE}) \)
from \( has3A \) show \( \{y, z\} \subseteq A \) by simp

next
fix \( i \) assume \( ils: i \in Is \)
fix \( a \ b \) assume \( ab: a \in \{y, z\} \ b \in \{y, z\} \)
show \( (a (P' i)^b = (a (P i)^b) \)

proof\( (\text{cases } i = j) \)

case False
with \( ab \) ils \( jxyzP' \) profileP' \( profileP' \) has3A
show \( \text{thesis unfolding profile-def by auto} \)

next
case True
from \( profileP' \) ils \( jxyzP' \) have \( z (P' j)^y \)
by \( (\text{auto dest: rpr-less-trans}) \)
with \( True \) ab ils \( jxyzP' \) profileP' \( profileP' \) has3A
show \( \text{thesis unfolding profile-def strict-pref-def by auto} \)
qed

qed \( (\text{simp-all add: profileP profileP'}) \)

moreover have \( z (\text{suf } P')^y \)

proof --
from \( profileP' \) sd \( jxyzP' \) \( ixyzP' \) have \( x (\text{suf } P')^x y \) by \( (\text{simp add: semidecisive-def}) \)

moreover
from \( ixyzP' \) \( jxyzP' \) have \( \forall i. i \in Is \implies z (P' i)^x x \) by \( (\text{case-tac } i=j, \text{auto}) \)
with \( wp \) profileP' \( has3A \) have \( z (\text{suf } P')^x x \) by \( (\text{auto dest: weak-paretoD}) \)
moreover note \( \text{SWF-rpr[OF swf profileP'] } \)
ultimately show \( z (\text{suf } P')^y \)

unfolding \( \text{universal-domain-def by } (\text{blast dest: rpr-less-trans}) \)

qed
ultimately show \( z (\text{suf } P)^y \) unfolding \( \text{strict-pref-def by blast} \)

qed

The following results permute \( x, y \) and \( z \) to show how decisiveness can be obtained from semi-decisiveness in all cases. Again, quite tedious.

lemma decisive3:
assumes \( has3A: hasw [x,y,z] \ A \)
and \( iia: iia \text{ swf } A \ Is \)
and \( \text{suf: SWF swf } A \text{ is universal-domain} \)
and \( wp: \text{weak-pareto swf } A \text{ is universal-domain} \)
and \( sd: \text{semidecisive swf } A \text{ is } \{j\} x z \)
sshows decisive \( \text{swf } A \text{ is } \{j\} y z \)

using \( \text{has3A decisive2[OF - iia \text{ swf wp sd}] by } (\text{simp, blast}) \)

lemma decisive4:
assumes \( has3A: hasw [x,y,z] \ A \)
and \( iia: iia \text{ swf } A \ Is \)
and \( \text{suf: SWF swf } A \text{ is universal-domain} \)

20
and wp: weak-pareto swf A Is universal-domain
and sd: semidecisive swf A Is \{j\} y z
shows decisive swf A Is \{j\} y x
using has3A decisive1[OF - iia swf wp sd] by (simp, blast)

lemma decisive5:
assumes has3A: hasw [x,y,z] A
and iia: iia swf A Is
and swf: SWF swf A Is universal-domain
and wp: weak-pareto swf A Is universal-domain
and sd: semidecisive swf A Is \{j\} x y
shows decisive swf A Is \{j\} y x
proof –
from sd
have decisive swf A Is \{j\} x z by (rule decisive1[OF has3A iia swf wp])
hence semidecisive swf A Is \{j\} x z by (rule d-imp-sd)
hence decisive swf A Is \{j\} y z by (rule decisive3[OF has3A iia swf wp])
hence semidecisive swf A Is \{j\} y z by (rule d-imp-sd)
thus decisive swf A Is \{j\} y x by (rule decisive4[OF has3A iia swf wp])
qed

lemma decisive6:
assumes has3A: hasw [x,y,z] A
and iia: iia swf A Is
and swf: SWF swf A Is universal-domain
and wp: weak-pareto swf A Is universal-domain
and sd: semidecisive swf A Is \{j\} y x
shows decisive swf A Is \{j\} y z decisive swf A Is \{j\} z x decisive swf A Is \{j\} x y
proof –
from has3A have has3A’: hasw [y,x,z] A by auto
show decisive swf A Is \{j\} y z by (rule decisive1[OF has3A’ iia swf wp sd])
show decisive swf A Is \{j\} z x by (rule decisive2[OF has3A’ iia swf wp sd])
show decisive swf A Is \{j\} x y by (rule decisive5[OF has3A’ iia swf wp sd])
qed

lemma decisive7:
assumes has3A: hasw [x,y,z] A
and iia: iia swf A Is
and swf: SWF swf A Is universal-domain
and wp: weak-pareto swf A Is universal-domain
and sd: semidecisive swf A Is \{j\} x y
shows decisive swf A Is \{j\} y z decisive swf A Is \{j\} z x decisive swf A Is \{j\} x y
proof –
from sd
have decisive swf A Is \{j\} y x by (rule decisive5[OF has3A iia swf wp])
hence semidecisive swf A Is \{j\} y x by (rule d-imp-sd)
thus decisive swf A Is \{j\} y z decisive swf A Is \{j\} z x decisive swf A Is \{j\} x y
by (rule decisive6[OF has3A’ iia swf wp sd])
qed

lemma j-decisive-xy:
assumes has3A: hasw [x,y,z] A
and iia: iia swf A Is
and $\text{suf}: \text{SWF}\ A$ is universal-domain
and $\text{wp}: \text{weak-pareto}\ A$ is universal-domain
and $\text{sd}: \text{semidecisive}\ A$ is $\{j\} x y$
and $\text{uv}: \text{hasw}[u,v] \{x,y,z\}$
shows decisive $A$ is $\{j\} u v$

using $\text{decisive1}[\text{OF}\ \text{has3A}\ \text{iia}\ \text{suf}\ \text{wp}\ \text{sd}]$
        $\text{decisive2}[\text{OF}\ \text{has3A}\ \text{iia}\ \text{suf}\ \text{wp}\ \text{sd}]$
        $\text{decisive5}[\text{OF}\ \text{has3A}\ \text{iia}\ \text{suf}\ \text{wp}\ \text{sd}]$
        $\text{decisive7}[\text{OF}\ \text{has3A}\ \text{iia}\ \text{suf}\ \text{wp}\ \text{sd}]$
by $(\text{simp, blast})$

lemma $j$-decisive:
assumes $\text{has3A}: \text{has}\ 3\ A$
and $\text{iia}: \text{iia}\ A$ is
and $\text{suf}: \text{SWF}\ A$ is universal-domain
and $\text{wp}: \text{weak-pareto}\ A$ is universal-domain
and $\text{xyA}: \text{hasw}[x,y] A$
and $\text{sd}: \text{semidecisive}\ A$ is $\{j\} x y$
and $\text{uv}: \text{hasw}[u,v] A$
shows decisive $A$ is $\{j\} u v$

proof –
from $\text{has-extend-witness}[\text{OF}\ \text{has3A}\ \text{xyA}]$
obtain $z$ where $\text{xyzA}: \text{hasw}[x,y,z] A$ by auto
{
  assume $\text{ux}: u = x$ and $\text{vy}: v = y$
  with $\text{xyzA iia suf wp sd have } ?\text{thesis}$ by (auto intro: $j$-decisive-xy)
}
moreover
{
  assume $\text{ux}: u = x$ and $\text{vNEy}: v \neq y$
  with $\text{xyzA iia suf wp sd have } ?\text{thesis}$ by (auto intro: $j$-decisive-xy[of x y])
}
moreover
{
  assume $\text{uy}: u = y$ and $\text{vx}: v = x$
  with $\text{xyzA iia suf wp sd have } ?\text{thesis}$ by (auto intro: $j$-decisive-xy)
}
moreover
{
  assume $\text{uy}: u = y$ and $\text{vNEx}: v \neq x$
  with $\text{uv xyA iia suf wp sd have } ?\text{thesis}$ by (auto intro: $j$-decisive-xy)
}
moreover
{
  assume $\text{uNExy}: u \notin \{x,y\}$ and $\text{ex}: v = x$
  with $\text{uv xyA iia suf wp sd have } ?\text{thesis}$ by (auto intro: $j$-decisive-xy[of x y])
}
moreover
{
  assume $\text{uNExy}: u \notin \{x,y\}$ and $\text{vy}: v = y$
  with $\text{uv xyA iia suf wp sd have } ?\text{thesis}$ by (auto intro: $j$-decisive-xy[of x y])
}
moreover

22
{ 
assume uNExy: u \notin \{x,y\} and vNExy: v \notin \{x,y\} 
with uw xyA iia swf wp sd 
have decisive swf A Is \{j\} x u by (auto intro: j-decisive-xy[where x=x and z=u]) 
hence sdzu: semidecisive swf A Is \{j\} x u by (rule d-imp-sd) 
with uNExy vNExy uv xyA iia swf wp have \?thesis by (auto intro: j-decisive-xy[of x]) }
ultimately show \?thesis by blast 
qed

The first result: if \(j\) is semidecisive for some alternatives \(u\) and \(v\), then they are actually a dictator.

**lemma** sd-imp-dictator: 
 assumes has3A: hasw \[x,y,z\] A 
 and iia: iia swf A Is 
 and swf: SWF swf A Is universal-domain 
 and wp: weak-pareto swf A Is universal-domain 
 and uw: hasw [u,v] A 
 and sd: semidecisive swf A Is \{j\} u v 
 shows dictator swf A Is j 
proof 
 fix x y assume x: x \in A and y: y \in A 
 show decisive swf A Is \{j\} x y 
 proof(cases x = y) 
 case True with sd show decisive swf A Is \{j\} x y by (blast intro: d-refl) 
 next 
 case False 
 with x y iia swf wp has3A uw sd show decisive swf A Is \{j\} x y 
 by (auto intro: j-decisive) 
 qed
 next 
 from sd show j \in Is by blast 
qed

4.2 The Existence of a Semi-decisive Individual

The second half of the proof establishes the existence of a semi-decisive individual. The required witness is essentially an encoding of the Condorcet paradox (aka "the paradox of voting" that shows we get tied up in knots if a certain agent didn’t have dictatorial powers.

**lemma** sd-exists-witness: 
 assumes has3A: hasw \[x,y,z\] A 
 and V8: Is = V1 \cup V2 \cup V3 
 \land V1 \cap V2 = {} \land V1 \cap V3 = {} \land V2 \cap V3 = {} 
 and Is: Is \neq {} 
 obtains P 
 where profile A Is P 
 and \(\forall i \in V1\). x (P i) \prec y \land y (P i) \prec z 
 and \(\forall i \in V2\). z (P i) \prec x \land x (P i) \prec y 
 and \(\forall i \in V3\). y (P i) \prec z \land z (P i) \prec x 
proof 
 let \(?P = \) 
 \lambda i. (if i \in V1 then \{ (x, u) | u. u \in A \})
\[
\begin{align*}
\cup \{ (y, u) | u. u \in A \land u \neq x \} \\
\cup \{ (z, u) | u. u \in A \land u \neq x \land u \neq y \}
\end{align*}
\]

else

\[
\begin{align*}
\text{if } i \in V2 \text{ then } \{ (z, u) | u. u \in A \} \\
\cup \{ (x, u) | u. u \in A \land u \neq z \} \\
\cup \{ (y, u) | u. u \in A \land u \neq x \land u \neq z \}
\end{align*}
\]

else

\[
\begin{align*}
\cup \{ (y, u) | u. u \in A \} \\
\cup \{ (z, u) | u. u \in A \land u \neq y \} \\
\cup \{ (x, u) | u. u \in A \land u \neq y \land u \neq z \}
\end{align*}
\]

\[
\cup \{ (u, v) | u. v. u \in A - \{x,y,z\} \land v \in A - \{x,y,z\}\}
\]

show \emph{profile} \(A\ Is\ \neq \emptyset\).

proof

fix \(i\) assume \(i\in Is\)

show \emph{rpr} \(A\ (\neq_i \ i)\)

proof

show \emph{complete} \(A\ (\neq_i \ i)\) by \(\text{(simp add: complete-def, blast)}\)

from \emph{has3A} \emph{ilS} show \emph{refl-on} \(A\ (\neq_i \ i)\) by \(\text{-(simp, blast)}\)

from \emph{has3A} \emph{ilS} show \emph{trans} \((\neq_i \ i)\) by \(\text{(clarsimp simp add: trans-def)}\)

qed

next

from \emph{Is} show \emph{Is} \(\neq \{\}\).

qed

from \emph{has3A} \emph{Vs}

show \(\forall i. x \in V1. \ (\neq_i \ i)\prec y \land y \prec (\neq_i \ i)\prec z\)

and \(\forall i. x \in V2. \ (\neq_i \ i)\prec x \land x \prec (\neq_i \ i)\prec y\)

and \(\forall i. x \in V3. \ (\neq_i \ i)\prec z \land z \prec (\neq_i \ i)\prec x\)

unfolding \emph{strict-pref-def} by \emph{auto}

qed

This proof is unfortunately long. Many of the statements rely on a lot of context, making it difficult to split it up.

\textbf{lemma} \emph{sd-exists}: 

\textbf{assumes} \emph{has3A}: \emph{has} 3 \emph{A} 

and \emph{finiteIs}: \emph{finite} \emph{Is}

and \emph{twoIs}: \emph{has} 2 \emph{Is}

and \emph{iia}: \emph{iia} \emph{swf} \emph{A} \emph{Is}

and \emph{swf}: \emph{SWF} \emph{swf} \emph{A} \emph{Is} \emph{universal-domain}

and \emph{wp}: \emph{weak-pareto} \emph{swf} \emph{A} \emph{Is} \emph{universal-domain}

\textbf{shows} \(\exists j u v. \emph{hasw} [u,v] \ A \land \emph{semidecisive} \emph{swf} \emph{A} \emph{Is} \{j\} u v\)

\textbf{proof}

let \(\neq_P = \lambda S. S \subseteq Is \land S \neq \{\} \land (\exists u v. \emph{hasw} [u,v] \ A \land \emph{semidecisive} \emph{swf} \emph{A} \emph{Is} S u v)\)

obtain \(u v\) \textbf{where} \(uvA: \emph{hasw} [u,v] \ A\)

using \emph{has-witness-two}[(\emph{OF has3A}) \textbf{by} \emph{auto}]

— The weak pareto requirement implies that the set of all individuals is decisive between any given alternatives.

\textbf{hence} \emph{decisive} \emph{swf} \emph{A} \emph{Is} \emph{Is} \emph{u} \emph{v}

by \(\text{-(rule, auto intro: \emph{weak-paretoD}[(\emph{OF \emph{wp}})])}\)

\textbf{hence} \emph{semidecisive} \emph{swf} \emph{A} \emph{Is} \emph{Is} \emph{u} \emph{v} \textbf{by} \(\text{(rule \emph{d-imp-sd})}\)

\textbf{with} \(uvA \ \emph{iia} \emph{swf} \emph{has-suc-notempty}[\textbf{where} \emph{n}=1] \emph{nat-2}[\textbf{symmetric}]\)

\textbf{have} \(\neq_P \emph{Is} \textbf{by} \emph{auto}\)

— Obtain a minimally-sized semi-decisive set.

\textbf{from} \emph{ex-has-least-nat}[\textbf{where} \emph{P}=\neq_P \textbf{and} \emph{m}=\emph{card}, \emph{OF this}]
obtain $V \times y$ where $V \subseteq Is$
and $V \not= \emptyset$
and $xyA$: hasw $[x,y] A$
and $Vsd$: semidecisive $swf A Is V \times y$
and $Vmin: \bigwedge V '. \ ?P V ' \implies card V \leq card V '$
by blast
from $VIs$ finite
have $Vfinite V V \not= \emptyset$
by (rule finite-subset)
— Show that minimal set contains a single individual.
from $Vfinite V Vnotempty$ have $\exists j. \ V = \{j\}$
proof (rule finite-set-singleton-contr)
assume $Vcard 1 < card V$
then obtain $j$ where $jV$
using has-extend-witness
by auto
— Split an individual from the "minimal" set.
let $?V1 = \{j\}$
let $?V2 = V - $?V1
let $?V3 = Is - V$
from $jV card-Diff-singleton$ have $V2card: card ?V2 > 0 card ?V2 < card V$ by auto
hence $V2notempty: \{\} \not= ?V2$ by auto
from $jV VIs$
have $jV2V3: Is = ?V1 \cup ?V2 \cup ?V3 \land ?V1 \cap ?V2 = \{\} \land ?V1 \cap ?V3 = \{\} \land ?V2 \cap ?V3 = \{\}$
by auto
— Show that that individual is semi-decisive for $x$ over $z$.
from $has-extend-witness$ have $\exists x. \ y (swf P)$
by auto
obtain $z$ where $threeDist: hasw [x,y,z] A$
from $sd-exists-witness$ have $\forall i \in ?V3. \ y (P i) \prec x \land x (P i) \prec y$
and $V3xyzP$
by (simp, blast)
have $xPy: x (swf P) < y$
proof (rule rpr-less-le-trans)
from $threeDist$ have $\exists y (swf P)$
by auto
next
— $V2$ is semi-decisive, and everyone else opposes their choice. Ergo they prevail.
show $x (swf P) < y$
proof
from $profileP V3xyzP$
have $\forall i \in ?V3. \ y (P i) < x$ by (blast dest: rpr-less-trans)
with $profileP V1xyzP V2xyzP Vsd$
show $\exists \ thesis$ unfolding semidecisive-def by auto
qed
next
— This result is unfortunately quite tortuous.
from $SWF-rpr[\ OF \ swf]$ have $y (swf P) \preceq z$
proof (rule rpr-less-not)
from $threeDist$ have $hasw [z, y] A$
by auto
next
assume $zPy: z (swf P) < y$
have semidecisive swf A Is ?V2 z y
proof
  from VIs show V - {j} ⊆ Is by blast
next
  fix P'
  assume profileP': profile A Is P'
  and V2yz': \( \forall i. \ i \in ?V2 \implies z (P'_i) \prec y \)
  and nV2yz': \( \forall i. \ i \in Is - ?V2 \implies y (P'_i) \prec z \)
from iia have \( \\bigwedge a \ b. \ [ a \in \{y, z\}; \ b \in \{y, z\} ] \implies (a (\text{swf } P) \preceq b) = (a (\text{swf } P') \preceq b) \)
proof\( (\text{rule iiaE}) \)
  from threeDist show yzA: \{y, z\} ⊆ A by simp
next
  fix i assume iIs: i ∈ Is
  fix a b assume ab: a ∈ \{y, z\} b ∈ \{y, z\}
  with VIs profileP V2xyzP
  have V2yzP: \( \forall i \in ?V2. \ z (P'_i) \prec y \)
  show (a (P'_i) ≤ b) = (a (P_i) ≤ b)
proof\( (\text{cases } i \in ?V2) \)
  case True
    with VIs profileP profileP' ab V2yzP threeDist
    show \( \square \)thesis unfolding strict-pref-def profile-def by auto
next
  case False
    from V1xyzP V3xyzP
    have \( \forall i \in Is - ?V2. \ y (P_i) \prec z \)
    with iIs False VIs jV profileP profileP' ab nV2yz' threeDist
    show \( \square \)thesis unfolding profile-def strict-pref-def by auto
qed

\[ \text{qed (simp-all add: profileP profileP')} \]
with zPy show z (swf P') ∼ y unfolding strict-pref-def by blast

\[ \text{with VIs Vsd Vmin[where } V'=?V2]\]
V2card V2notempty threeDist show False
by auto

\[ \text{qed (simp add: profileP threeDist)} \]

\[ \text{qed} \]
have semidecisive swf A Is ?V1 x z
proof
  from jV VIs show \{j\} ⊆ Is by blast
next
  — Use iia to show the SWF must allow the individual to prevail.
  fix P'
  assume profileP': profile A Is P'
  and V1yz': \( \forall i. \ i \in ?V1 \implies x (P'_i) \prec z \)
  and nV1yz': \( \forall i. \ i \in Is - ?V1 \implies z (P'_i) \prec x \)
from iia have \( \\bigwedge a \ b. \ [ a \in \{x, z\}; \ b \in \{x, z\} ] \implies (a (\text{swf } P) \preceq b) = (a (\text{swf } P') \preceq b) \)
proof\( (\text{rule iiaE}) \)
  from threeDist show xxA: \{x, z\} ⊆ A by simp
next
  fix i assume iIs: i ∈ Is
  fix a b assume ab: a ∈ \{x, z\} b ∈ \{x, z\}
  show (a (P'_i) ≤ b) = (a (P_i) ≤ b)
proof (cases \( i \in ?V1 \))
  case True
  with jV VIs profileP V1xyzP
  have \( \forall i \in ?V1. \ x (p_i) \prec z \) by (blast dest: rpr-less-trans)
  with True jV VIs profileP profileP’ ab V1yz’ threeDist
  show \(?thesis unfolding strict-pref-def profile-def by auto\)
next
  case False
  from V2xyzP V3xyzP
  have \( \forall i \in Is \dashv ?V1. \ z (p_i) \prec x \) by auto
  with iIs False VIs jV profileP profileP’ ab nV1yz’ threeDist
  show \(?thesis unfolding strict-pref-def profile-def by auto\)
qed
qd (simp-all add: profileP profileP’)
with xPz show \( x (swf P) \prec z \) unfolding strict-pref-def by blast
qd
with jV VIs Vsd Vmin[where \( V' \dashv ?V1 \)] V2card threeDist show False
by auto
qd
with xyA Vsd show \(?thesis by blast\)
qd

4.3 Arrow’s General Possibility Theorem

Finally we conclude with the celebrated “possibility” result. Note that we assume the set of
individuals is finite; [Rou79] relaxes this with some fancier set theory. Having an infinite set
of alternatives doesn’t matter, though the result is a bit more plausible if we assume finiteness
[Sen70, p54].

theorem ArrowGeneralPossibility:
assumes has3A: \( has 3 A \)
and finiteIs: finite Is
and has2Is: \( has 2 Is \)
and iia: iia swf A Is
and swf: SWF swf A Is universal-domain
and wp: weak-pareto swf A Is universal-domain
obtains j where dictator swf A Is j
using sd-imp-dictator[OF has3A iia swf wp]
sd-exists[OF has3A finiteIs has2Is iia swf wp]
by blast

5 Sen’s Liberal Paradox

5.1 Social Decision Functions (SDFs)

To make progress in the face of Arrow’s Theorem, the demands placed on the social choice
function need to be weakened. One approach is to only require that the set of alternatives
that society ranks highest (and is otherwise indifferent about) be non-empty.
Following [Sen70, Chapter 4*], a Social Decision Function (SDF) yields a choice function for every profile.

**definition**

\[ SDF :: (\text{a, i}) \rightarrow \text{SCF} \Rightarrow \text{i set} \Rightarrow (\text{a set} \Rightarrow \text{i set} \Rightarrow (\text{a, i}) \rightarrow \text{Profile} \Rightarrow \text{bool}) \Rightarrow \text{bool} \]

**where**

\[ SDF \text{ sdf A Is Pcond} \equiv (\forall P. \text{Pcond A Is P} \rightarrow \text{choiceFn A (sdf P)}) \]

**lemma SDFI [intro]:**

\[ (\forall P. \text{Pcond A Is P} \Rightarrow \text{choiceFn A (sdf P)}) \Rightarrow SDF \text{ sdf A Is Pcond} \]

**unfolding SDF-def by simp**

**lemma SWF-SDF:**

assumes finiteA: \text{finite A}

shows SWF scf A Is universal-domain \Rightarrow SDF scf A Is universal-domain

**unfolding SDF-def SWF-def by (blast dest: rpr-choiceFn[OF finiteA])**

In contrast to SWFs, there are SDFs satisfying Arrow’s (relevant) requirements. The lemma uses a witness to show the absence of a dictatorship.

**lemma SDF-nodictator-witness:**

assumes has2A: \text{has 2 A}

and has2Is: \text{has 2 Is}

obtains P

**where**

profile A Is P

and \( x (P i) \prec y \)

and \( y (P j) \prec x \)

**proof**

let \( \lambda k. (\text{if } k = i \text{ then } \{(x, u) \mid u \in A\} \}
\cup \{(y, u) \mid u \in A - \{x\}\})
\text{else } (\{(y, u) \mid u \in A\}
\cup \{(x, u) \mid u \in A - \{y\}\}))
\cup (A - \{x,y\}) \times (A - \{x,y\})
show profile A Is P

**unfolding strict-pref-def by auto**

**qed**

from has2A has2Is

show \( x (\lambda \text{P i}) \prec y \)

and \( y (\lambda \text{P j}) \prec x \)

**unfolding strict-pref-def by auto**

**qed**

**lemma SDF-possibility:**

assumes finiteA: \text{finite A}

and has2A: \text{has 2 A}

and has2Is: \text{has 2 Is}

obtains sdf

**where**

weak-pareto sdf A Is universal-domain

and iia sdf A Is

28
\[ \neg(\exists j. \text{dictator } \text{sdf} A \text{ Is } j) \]

\[ \text{and } SDF \text{ sdf } A \text{ Is universal-domain} \]

**proof** –

let \( ?\text{sdf} = \lambda P. \{ (x, y) \cdot x \in A \land y \in A \land \neg ((\forall i \in Is.\ y (P i) \preceq x) \land (\exists i \in Is.\ y (P i) < x)) \} \)

have weak-pareto \( ?\text{sdf} A \text{ Is universal-domain} \)

by (rule, unfold strict-pref-def, auto dest: profile-non-empty)

moreover

have \( ?\text{sdf} A \text{ Is unfolding } \text{strict-pref-def by auto} \)

moreover

have \( \neg(\exists j. \text{dictator } ?\text{sdf} A \text{ Is } j) \)

**proof**

assume \( \exists j. \text{dictator } ?\text{sdf} A \text{ Is } j \)

then obtain \( j \text{ where } jIs: j \in Is \)

and \( jD: \forall x \in A.\ \forall y \in A.\ \text{decisive } ?\text{sdf} A \text{ Is } \{j\} x y \)

unfolding decisive-def decisive-def by auto

from \( jIs \text{ has-witness-two[OF has2Is] obtain } i jIs: \text{hasw } [i,j] \text{ Is} \) by auto

from \( \text{has-witness-two[OF has2A] obtain } x y \text{ where } xyA: \text{hasw } [x,y] A \) by auto

from \( xyA i jIs \text{ obtain } P \)

where \( \text{profileP}: \text{profile } A \text{ Is } P \)

and \( \text{yPix}: x (P i) \prec y \)

and \( \text{yPjx}: y (P j) \prec x \)

by (rule SDF-nodictator-witness)

from \( \text{profileP jD jIs xyA yPix} \text{ have } y (?\text{sdf } P) \prec x \)

unfolding decisive-def by simp

moreover

from \( i jIs xyA yPjx yPix \text{ have } x (?\text{sdf } P) \preceq y \)

unfolding strict-pref-def by auto

ultimately show False

unfolding strict-pref-def by blast

qed

moreover

have \( SDF ?\text{sdf} A \text{ Is universal-domain} \)

**proof**

fix \( P \)

assume \( \text{ud}: \text{universal-domain } A \text{ Is } P \)

show \( \text{choiceFn } A ( ?\text{sdf } P) \)

**proof** (rule r-c-qt-imp-cf[OF finiteA])

show \( \text{complete } A ( ?\text{sdf } P) \text{ and refl-on } A ( ?\text{sdf } P) \)

unfolding strict-pref-def by auto

show \( \text{quasi-trans } ( ?\text{sdf } P) \)

**proof**

fix \( x y z \)

assume \( xy: x (?\text{sdf } P) \prec y \text{ and } yz: y (?\text{sdf } P) \prec z \)

from \( xy yz \) have \( xyza: x \in A\ y \in A\ z \in A \)

unfolding strict-pref-def by auto

from \( xy yz \) have \( AxRy: \forall i \in Is.\ x (P i) \preceq y \)

and \( ExPy: \exists i \in Is.\ x (P i) \prec y \)

and \( AyRz: \forall i \in Is.\ y (P i) \preceq z \)

unfolding strict-pref-def by auto

from \( AxRy AyRz ud \) have \( AxRz: \forall i \in Is.\ x (P i) \preceq z \)
Sen makes several other stronger statements about SDFs later in the chapter. I leave these for future work.

### 5.2 Sen’s Liberal Paradox

Having side-stepped Arrow’s Theorem, Sen proceeds to other conditions one may ask of an SCF. His analysis of liberalism, mechanised in this section, has attracted much criticism over the years [AK96].

Following [Sen70, Chapter 6*], a liberal social choice rule is one that, for each individual, there is a pair of alternatives that she is decisive over.

**definition** liberal :: ('a, 'i) SCF ⇒ 'a set ⇒ 'i set ⇒ bool where

\[
\text{liberal scf A Is} \equiv \\
(\forall i \in \text{Is}. \exists x \in A. \exists y \in A. x \neq y \\
\wedge \text{decisive scf A Is \{i\} x y} \wedge \text{decisive scf A Is \{i\} y x})
\]

**lemma** liberalE:

\[
\text{\[ \text{liberal scf A Is; i \in Is} \] } \\
\Rightarrow \exists x \in A. \exists y \in A. x \neq y \\
\wedge \text{decisive scf A Is \{i\} x y} \wedge \text{decisive scf A Is \{i\} y x}
\]

This condition can be weakened to require just two such decisive individuals; if we required just one, we would allow dictatorships, which are clearly not liberal.

**definition** minimally-liberal :: ('a, 'i) SCF ⇒ 'a set ⇒ 'i set ⇒ bool where

\[
\text{minimally-liberal scf A Is} \equiv \\
(\exists i \in \text{Is}. \exists j \in \text{Is}. i \neq j \\
\wedge (\exists x \in A. \exists y \in A. x \neq y \\
\wedge \text{decisive scf A Is \{i\} x y} \wedge \text{decisive scf A Is \{i\} y x})
\]

\[
\wedge (\exists x \in A. \exists y \in A. x \neq y \\
\wedge \text{decisive scf A Is \{j\} x y} \wedge \text{decisive scf A Is \{j\} y x})
\]

**lemma** liberal-imp-minimally-liberal:

\[
\text{assumes has2Is: has 2 Is} \\
\text{shows minimally-liberal scf A Is}
\]

**proof**

\[
\text{from has-extend-witness[where xs=[], OF has2Is]} \\
\text{obtain i where i: i \in Is by auto} \\
\text{with has-extend-witness[where xs=[i], OF has2Is]} \\
\text{obtain j where j: j \in Is i \neq j by auto} \\
\text{from L i j show \text{thesis}}
\]
The key observation is that once we have at least two decisive individuals we can complete the Condorcet (paradox of voting) cycle using the weak Pareto assumption. The details of the proof don’t give more insight.

Firstly we need three types of profile witnesses (one of which we saw previously). The main proof proceeds by case distinctions on which alternatives the two liberal agents are decisive for.

**lemmas** liberal-witness-two = SDF-nodictator-witness

**lemma** liberal-witness-three:

**assumes** threeA: hasw [x,y,v] A
  and twoIs: hasw [i,j] Is

**obtain** P
  where profile A Is P
  and x (P i) ≺ y
  and v (P j) ≺ x
  and ∀ i ∈ Is. y (P i) ≺ v

**proof**

- let ?P =

\[ \lambda a. \text{if } a = i \text{ then } \{ (x', u') \mid u' \in A \} \]
  \[ \cup \{ (y', u') \mid u' \in A - \{x\} \} \]
  \[ \cup (A - \{x,y\}) \times (A - \{x\}) \]
  \[ \text{else } \{ (y', u') \mid u' \in A \} \]
  \[ \cup \{ (v', u') \mid u' \in A - \{y\} \} \]
  \[ \cup (A - \{v,y\}) \times (A - \{v\}) \]

have profile A Is ?P

**proof**

- fix i assume iis: i ∈ Is
- show rpr A (?P i)

**proof**

- show complete A (?P i) by (simp, blast)
- from threeA iis show refl-on A (?P i) by (simp, blast)
- from threeA iis show trans (?P i) by (clarsimp simp add: trans-def)

**qed**

next

- from twoIs show Is ≠ {} by auto

**qed**

moreover

- from threeA twoIs have x (?P i) ≺ y v (?P j) ≺ x ∀ i ∈ Is. y (?P i) ≺ v

**unfolding** strict-pref-def by auto

ultimately show ?thesis ..

**qed**

**lemma** liberal-witness-four:

**assumes** fourA: hasw [x,y,u,v] A
  and twoIs: hasw [i,j] Is

**obtain** P
  where profile A Is P
  and x (P i) ≺ y
  and u (P j) ≺ v

**proof**

- let ?P =

\[ \lambda a. \text{if } a = i \text{ then } \{ (x', u') \mid u' \in A \} \]
  \[ \cup \{ (y', u') \mid u' \in A - \{x\} \} \]
  \[ \cup (A - \{x,y\}) \times (A - \{x\}) \]
  \[ \text{else } \{ (y', u') \mid u' \in A \} \]
  \[ \cup \{ (v', u') \mid u' \in A - \{y\} \} \]
  \[ \cup (A - \{v,y\}) \times (A - \{v\}) \]

have profile A Is ?P

**proof**

- fix i assume iis: i ∈ Is
- show rpr A (?P i)

**proof**

- show complete A (?P i) by (simp, blast)
- from threeA iis show refl-on A (?P i) by (simp, blast)
- from threeA iis show trans (?P i) by (clarsimp simp add: trans-def)

**qed**

next

- from twoIs show Is ≠ {} by auto

**qed**

moreover

- from threeA twoIs have x (?P i) ≺ y v (?P j) ≺ x ∀ i ∈ Is. y (?P i) ≺ v

**unfolding** strict-pref-def by auto

ultimately show ?thesis ..

**qed**
and $\forall i \in I_s. v (P_i) \prec x \land y (P_i) \prec u$

**proof**

**let** $\overline{?P} =$

$\lambda a. \text{if } a = i \text{ then } \{ (v, w) \mid w, w \in A \}$

$\cup \{ (x, w) \mid w, w \in A - \{v\} \}$

$\cup \{ (y, w) \mid w, w \in A - \{v,x\} \}$

$\cup (A - \{v,x,y\}) \times (A - \{v,x,y\})$

$\text{else } \{ (y, w) \mid w, w \in A - \{y\} \}$

$\cup \{ (v, w) \mid w, w \in A - \{u,y\} \}$

$\cup (A - \{u,v,y\}) \times (A - \{u,v,y\})$

**have** *profile* $A$ $I_s$ $?P$

**proof**

**fix** $i$ **assume** $iis$: $i \in I_s$

**show** $\text{rpr } A (\overline{?P} i)$

**proof**

**show** $\text{complete } A (\overline{?P} i)$ by (simp, blast)

**from** $\text{four } iis$ **show** $\text{refl-on } A (\overline{?P} i)$ by (simp, blast)

**from** $\text{four } iis$ **show** $\text{trans } (\overline{?P} i)$ by (clarsimp simp add: trans-def)

**qed**

**next**

**from** $\text{twoIs}$ **show** $I_s \neq \{\}$ by auto

**qed**

**moreover**

**from** $\text{four } \text{twoIs}$ **have**

$x (\overline{?P} j) \prec u (\overline{?P} j) \prec v \forall i \in I_s. v (\overline{?P} i) \prec x \land y (\overline{?P} i) \prec u$

by (unfold strict-pref-def, auto)

**ultimately** **show** $\text{thesis ..}$

**qed**

The Liberal Paradox: having two decisive individuals, an SDF and the weak pareto assumption is inconsistent.

**theorem** $\text{LiberalParadox}$:

**assumes** $\text{SDF}: \text{SDF sdf } A I_s \text{ universal-domain}$

and $\text{ml}: \text{minimally-liberal sdf } A I_s$

and $\text{wp}: \text{weak-pareto sdf } A I_s \text{ universal-domain}$

**shows** $\text{False}$

**proof**

**from** $\text{ml}$ **obtain** $i \ j \ x \ y \ u \ v$

**where** $i: i \in I_s$ and $j: j \in I_s$ and $ij: i \neq j$

and $x: x \in A$ and $y: y \in A$ and $u: u \in A$ and $v: v \in A$

and $xy: x \neq y$

and $\text{dixy: decisive sdf } A I_s \{i\} x y$

and $\text{dixy: decisive sdf } A I_s \{i\} y x$

and $\text{djuv: decisive sdf } A I_s \{j\} u v$

and $\text{djuv: decisive sdf } A I_s \{j\} v u$

by (unfold minimally-liberal-def, auto)

**from** $i \ j \ ij$ **have** $\text{twoIs: hasw } [i,j] I_s$ by simp

{ 
**assume** $xz: x = u$ and $yz: y = v$

**from** $xy \ x$ **have** $\text{twoA: hasw } [x,y] A$ by simp

**obtain** $P$

32
where profile A Is P x (P i) y y (P j) x
using liberal-witness-two [OF twoA twoIs] by blast
with i j dixy djuv xu yv have False
by (unfold decisive-def strict-pref-def, blast)
}

moreover
{
assume xu: x = u and yv: y ≠ v
with xy wv xu x y v have threeA: hasw [x, y, v] A by simp
obtain P
  where profileP: profile A Is P
  and xPiy: x (P i) y
  and vPjx: v (P j) x
  and AyPv: ∀i ∈ Is. y (P i) v
using liberal-witness-three [OF threeA twoIs] by blast
from vPj x djuv xu profileP have vPx: v (sdf P) x
by (unfold decisive-def strict-pref-def, auto)
from xPig i dixy profileP have xPy: x (sdf P) y
by (unfold decisive-def strict-pref-def, auto)
from AyPv weak-paretoD [OF wp - y v] profileP have yPv: y (sdf P) v
by auto
from threeA profileP SDF have choiceSet {x, y, v} (sdf P) ≠ {}
by (simp add: SDF-def choiceFn-def)
with vPx xP yP have False
by (unfold choiceSet-def strict-pref-def, blast)
}

moreover
{
assume xv: x = v and yu: y = u
from xy x y have twoA: hasw [x, y] A by auto
obtain P
  where profile P x (P i) y y (P j) x
using liberal-witness-two [OF twoA twoIs] by blast
with i j dixy djuv xv yu have False
by (unfold decisive-def strict-pref-def, blast)
}

moreover
{
assume xv: x = v and yu: y ≠ u
with xy wu xu x y have threeA: hasw [x, y, u] A by simp
obtain P
  where profileP: profile A Is P
  and xPiy: x (P i) y
  and uPjx: u (P j) x
  and AyPu: ∀i ∈ Is. y (P i) u
using liberal-witness-three [OF threeA twoIs] by blast
from uPj x djuv xv profileP have uPx: u (sdf P) x
by (unfold decisive-def strict-pref-def, auto)
from xPig i dixy profileP have xPy: x (sdf P) y
by (unfold decisive-def strict-pref-def, auto)
from AyPu weak-paretoD [OF wp - y u] profileP have yPu: y (sdf P) u

by auto
from threeA profileP SDF have choiceSet \{x,y,u\} (sdf P) \neq {} 
by (simp add: SDF-def choiceFn-def)
with uPx xPy yPu have False 
by (unfold choiceSet-def strict-pref-def, blast)
}

moreover
{
assume xu: x \neq u and xv: x \neq v and yu: y = u 
with v x y xy xv xu have threeA: hasw [y,x,v] A by simp
obtain P 
where profileP: profile A Is P 
and yPix: y (P i) \prec x 
and uPjy: v (P j) \prec y 
and AxPv: \forall i \in Is. x (P i) \prec v 
using liberal-witness-three[OF threeA twoIs] by blast 
from yPix i dyx profileP have yPx: y (sdf P) \prec x 
by (unfold decisive-def strict-pref-def, auto)
from uPjy j djv y profileP have vPy: v (sdf P) \prec y 
by (unfold decisive-def strict-pref-def, auto)
from AxPv weak-paretoD[OF wp - x v] profileP have xPv: x (sdf P) \prec v 
by auto
from threeA profileP SDF have choiceSet \{x,y,v\} (sdf P) \neq {} 
by (simp add: SDF-def choiceFn-def)
with yPx vPy xPv have False 
by (unfold choiceSet-def strict-pref-def, blast)
}

moreover
{
assume xu: x \neq u and xv: x \neq v and yu: y = v 
with u x y xy xv xu have threeA: hasw [y,x,u] A by simp
obtain P 
where profileP: profile A Is P 
and yPix: y (P i) \prec x 
and uPjy: u (P j) \prec y 
and AxPu: \forall i \in Is. x (P i) \prec u 
using liberal-witness-three[OF threeA twoIs] by blast 
from yPix i dyx profileP have yPx: y (sdf P) \prec x 
by (unfold decisive-def strict-pref-def, auto)
from uPjy j djv y profileP have uPy: u (sdf P) \prec y 
by (unfold decisive-def strict-pref-def, auto)
from AxPu weak-paretoD[OF wp - x u] profileP have xPu: x (sdf P) \prec u 
by auto
from threeA profileP SDF have choiceSet \{x,v,u\} (sdf P) \neq {} 
by (simp add: SDF-def choiceFn-def)
with yPx uPy xPu have False 
by (unfold choiceSet-def strict-pref-def, blast)
}

moreover
{
assume xu: x \neq u and xv: x \neq v and yu: y \neq u and yv: y \neq v
with $u, v, x, y, x, y, u, v$ have fourA: hasw $[x, y, u, v]$ $A$ by simp

obtain $P$

where profileP: profile A Is P

and xPy: $x (P \cdot v) \prec y$

and uPjv: $u (P \cdot j) \prec v$

and ArvPzAyPv: $\forall i \in Is. v (P \cdot i) \prec x \wedge y (P \cdot i) \prec u$

using liberal-witness-four[of fourA twoIs] by blast

from xPy i dixy profileP have xPy: $x (sdf P) \prec y$

by (unfold decisive-def strict-pref-def, auto)

from uPjv j djuv profileP have uPv: $u (sdf P) \prec v$

by (unfold decisive-def strict-pref-def, auto)

from ArvPzAyPv weak-paretoD[of wp] profileP x y u v

have vPx: $v (sdf P) \prec x$ and yPu: $y (sdf P) \prec u$ by auto

from fourA profileP SDF have choiceSet $\{x, y, u, v\} (sdf P) \neq \{\}$

by (simp add: SDF-def choiceFn-def)

with xPy uPv vPx yPu have False

by (unfold choiceSet-def strict-pref-def, blast)

ultimately show False by blast

qed

6 May’s Theorem

May’s Theorem [May52] provides a characterisation of majority voting in terms of four conditions that appear quite natural for a priori unbiased social choice scenarios. It can be seen as a refinement of some earlier work by Arrow [Arr63, Chapter V.1].

The following is a mechanisation of Sen’s generalisation [Sen70, Chapter 5*]: originally Arrow and May consider only two alternatives, whereas Sen’s model maps profiles of full RPRs to a possibly intransitive relation that does at least generate a choice set that satisfies May’s conditions.

6.1 May’s Conditions

The condition of anonymity asserts that the individuals’ identities are not considered by the choice rule. Rather than talk about permutations we just assert the result of the SCF is the same when the profile is composed with an arbitrary bijection on the set of individuals.

definition anonymous :: (‘a, ‘i) SCF ⇒ ‘a set ⇒ ‘i set ⇒ bool where

anonymous scf A Is ≡

($\forall P f x y. \text{profile A Is P } \wedge \text{bij-betw } f \text{ Is Is } \wedge x \in A \wedge y \in A$

$\rightarrow (x (scf P) \leq y) = (x (scf (P \circ f)) \leq y)$)

lemma anonymousI[intro]:

($\exists P f x y. \text{profile A Is P } \wedge \text{bij-betw } f \text{ Is Is;}$

$x \in A; y \in A \rightarrow (x (scf P) \leq y) = (x (scf (P \circ f)) \leq y)$)

$\Rightarrow \text{anonymous scf A Is}$

unfolding anonymous-def by simp
lemma anonymousD:
\[
\begin{align*}
\text{anonymous scf} A \, Is \equiv & \quad \text{profile } A \, Is \, P; \text{ bij-betw } f \, Is; \ x \in A; \ y \in A \\
\implies & \quad (x \ (\text{scf} \ P) \preceq y) = (x \ (\text{scf} \ (P \circ f)) \preceq y)
\end{align*}
\]

unfolding anonymous-def by simp

Similarly, an SCF is neutral if it is insensitive to the identity of the alternatives. This is Sen's characterisation [Sen70, p72].

definition neutral :: ('a, 'i) SCF \Rightarrow 'a set \Rightarrow 'i set \Rightarrow bool where
neutral scf A Is \equiv 
\[
\begin{align*}
\forall P \ P' \ x \ y \ z \ w. \ & \text{profile } A \, Is \, P \land \text{profile } A \, Is \, P' \land x \in A \land y \in A \land z \in A \land w \in A \\
\land & \quad (\forall i \in Is. \ x \ (P \ i) \preceq y \iff z \ (P' \ i) \preceq w) \land (\forall i \in Is. \ y \ (P \ i) \preceq x \iff w \ (P' \ i) \preceq z) \\
\implies & \quad (x \ (\text{scf} \ P) \preceq y \iff z \ (\text{scf} \ P') \preceq w) \land (y \ (\text{scf} \ P) \preceq x \iff w \ (\text{scf} \ P') \preceq z)
\end{align*}
\]

lemma neutralD[intro]:
\[
\begin{align*}
\bigwedge P \ P' \ x \ y \ z \ w. & \quad \text{profile } A \, Is \, P; \text{ profile } A \, Is \, P'; \ \{x,y,z,w\} \subseteq A \\
\bigwedge i. \ i \in Is. & \quad x \ (P \ i) \preceq y \iff z \ (P' \ i) \preceq w; \\
\bigwedge i. \ i \in Is. & \quad y \ (P \ i) \preceq x \iff w \ (P' \ i) \preceq z \\
\implies & \quad (x \ (\text{scf} \ P) \preceq y \iff z \ (\text{scf} \ P') \preceq w) \land (y \ (\text{scf} \ P) \preceq x \iff w \ (\text{scf} \ P') \preceq z)
\end{align*}
\]

lemma neutralD:
\[
\begin{align*}
\bigwedge P \ P' \ x \ y \ z \ w. & \quad \text{profile } A \, Is \, P; \text{ profile } A \, Is \, P'; \ \{x,y,z,w\} \subseteq A \\
\bigwedge i. \ i \in Is. & \quad x \ (P \ i) \preceq y \iff z \ (P' \ i) \preceq w; \\
\bigwedge i. \ i \in Is. & \quad y \ (P \ i) \preceq x \iff w \ (P' \ i) \preceq z \\
\implies & \quad (x \ (\text{scf} \ P) \preceq y \iff z \ (\text{scf} \ P') \preceq w) \land (y \ (\text{scf} \ P) \preceq x \iff w \ (\text{scf} \ P') \preceq z)
\end{align*}
\]

unfolding neutral-def by simp

Neutrality implies independence of irrelevant alternatives.

lemma neutral-iiA: neutral scf A Is \Rightarrow \text{iia} \ scf A Is

unfolding neutral-def by (rule, auto)

Positive responsiveness is a bit like non-manipulability: if one individual improves their opinion of \(x\), then the result should shift in favour of \(x\).

definition positively-responsive :: ('a, 'i) SCF \Rightarrow 'a set \Rightarrow 'i set \Rightarrow bool where
positively-responsive scf A Is \equiv 
\[
\begin{align*}
\forall P \ P' \ x \ y. & \quad \text{profile } A \, Is \, P \land \text{profile } A \, Is \, P' \land x \in A \land y \in A \\
\land & \quad (\exists k \in Is. \ x \ (P k) \approx y \iff x \ (P' k) \approx y) \land (y \ (P k) \prec x \iff x \ (P' k) \prec y) \\
\land & \quad (x \ (\text{scf} \ P) \preceq y \iff x \ (\text{scf} \ P') \preceq y)
\end{align*}
\]

lemma positively-responsiveD[intro]:
\[
\begin{align*}
\text{assumes } I: & \quad \bigwedge P \ P' \ x \ y. \\
\bigwedge i. \ i \in Is; \ & \quad x \ (P \ i) \prec y \\
\bigwedge i. \ i \in Is; \ & \quad x \ (P \ i) \approx y; \\
\exists k \in Is. & \quad x \ (P k) \approx y \iff x \ (P' k) \approx y)
\end{align*}
\]
\[ x \text{ (scf } P \text{)} \preceq y \] 
\[ \Rightarrow x \text{ (scf } P' \text{)} \prec y \]
shows positively-responsive scf A Is unfolding positively-responsive-def by (blast intro: I)

**Lemma** positively-responsiveD:
\[ \text{positively-responsive scf } A \text{ Is; } \]
\[ \text{profile } A \text{ Is } P; \text{ profile } A \text{ Is } P'; x \in A; y \in A; \]
\[ \forall i. \; i \in Is; \; x \text{ (} P \text{) } i \prec y \] \[ \Rightarrow x \text{ (} P' \text{) } i \preceq y; \]
\[ \forall i. \; i \in Is; \; x \text{ (} P \text{) } i \approx y \] \[ \Rightarrow x \text{ (} P' \text{) } i \preceq y; \]
\[ \exists k \in Is. \; (x \text{ (} P \text{) } k \approx y \wedge x \text{ (} P' \text{) } k \prec y) \lor (y \text{ (} P \text{) } k \prec x \wedge x \text{ (} P' \text{) } k \preceq y); \]
\[ x \text{ (scf } P \text{)} \preceq y \] \[ \Rightarrow x \text{ (scf } P' \text{)} \prec y \]
unfolding positively-responsive-def
apply clarsimp
apply (erule allE[where x=P])
apply (erule allE[where x=P'])
apply (erule allE[where x=x])
apply (erule allE[where x=y])
by auto

### 6.2 The Method of Majority Decision satisfies May’s conditions

The *method of majority decision* (MMD) says that if the number of individuals who strictly prefer \( x \) to \( y \) is larger than or equal to those who strictly prefer the converse, then \( x R y \). Note that this definition only makes sense for a finite population.

**Definition** MMD :: `'i set ⇒ ('a, 'i) SCF where
\[ \text{MMD Is } P \equiv \{ (x, y) . \; \text{card}\{i \in Is. \; x \text{ (} P \text{) } i \prec y\} \geq \text{card}\{i \in Is. \; y \text{ (} P \text{) } i \prec x\}\} \]

The first part of May’s Theorem establishes that the conditions are consistent, by showing that they are satisfied by MMD.

**Lemma** MMD-l2r:
\[ \text{fixes } A :: 'a set \]
\[ \text{and } Is :: 'i set \]
\[ \text{assumes finiteIs: finite Is} \]
\[ \text{shows SCF (MMD Is) A Is universal-domain} \]
\[ \text{and anonymous (MMD Is) A Is} \]
\[ \text{and neutral (MMD Is) A Is} \]
\[ \text{and positively-responsive (MMD Is) A Is} \]
**Proof** –
\[ \text{show SCF (MMD Is) A Is universal-domain} \]
**Proof**
\[ \text{fix } P \text{ show complete A (MMD Is } P) \]
\[ \text{by (rule completeI, unfold MMD-def, simp, arith) } \]
**Qed**
\[ \text{show anonymous (MMD Is) A Is} \]
**Proof**
\[ \text{fix } P \]
\[ \text{fix } x \; y :: 'a \]
\[ \text{fix } f \text{ assume bijf: bij-betw } f \text{ Is Is} \]

37
show \((x \ (MMD\ Is\ P) \preceq\ y) = (x \ (MMD\ Is\ (P \circ f)) \preceq\ y)\)

using card-compose-bij[of bijf, where \(P=\lambda i.\ x \ (P\ i) \prec y\)]
card-compose-bij[of bijf, where \(P=\lambda i.\ y \ (P\ i) \prec x\)]

unfolding MMD-def by simp

qed

next

show neutral \((MMD\ Is\ A)\ A\ Is\)

proof

fix \(P\ P'\)
fix \(x\ y\ z\ w\)
assume \(xyzw\): \(\{x, y, z, w\} \subseteq A\)
assume \(xyzw\): \(\forall i.\ i \in Is \implies (x \ (P\ i) \prec y) = (z \ (P'\ i) \preceq w)\)
and \(xyzw\): \(\forall i.\ i \in Is \implies (y \ (P\ i) \preceq y) = (w \ (P'\ i) \prec z)\)

from \(xyzw\) \(xyzw\) \(yzwz\)
have \(\{ i \in Is.\ x \ (P\ i) \prec y \} = \{ i \in Is.\ z \ (P'\ i) \prec w \}\)
and \(\{ i \in Is.\ y \ (P\ i) \preceq x \} = \{ i \in Is.\ w \ (P'\ i) \prec z \}\)

unfolding strict-pref-def by auto

thus \((x \ (MMD\ Is\ P) \preceq y) = (z \ (MMD\ Is\ P') \preceq w) \land\)
\((y \ (MMD\ Is\ P) \preceq z) = (w \ (MMD\ Is\ P') \preceq z)\)

unfolding MMD-def by simp

qed

next

show positively-responsive \((MMD\ Is\ A)\ A\ Is\)

proof

fix \(P\ P'\)
fix \(x\ y\)
assume \(xyA\): \(x \in A\ y \in A\)
assume \(xPy\): \(\forall i.\ [i \in Is;\ x \ (P\ i) \prec y] \implies x \ (P'\ i) \prec y\)
and \(xPy\): \(\forall i.\ [i \in Is;\ x \ (P\ i) \preceq y] \implies x \ (P'\ i) \preceq y\)
and \(k\): \(\forall k \in Is.\ x \ (P\ k) \preceq y \land x \ (P'\ k) \prec y \lor y \ (P\ k) \prec x \land x \ (P'\ k) \preceq y\)
and \(xRSCfy\): \(x \ (MMD\ Is\ P) \preceq y\)

from \(k\) obtain \(k\)
where \(kIs\): \(k \in Is\)
and \(kcond\): \(\{ x \ (P\ k) \preceq y \land x \ (P'\ k) \prec y\} \lor \{ y \ (P\ k) \prec x \land x \ (P'\ k) \preceq y\}\)
by blast
let \(?xPy\) = \(\{ i \in Is.\ x \ (P\ i) \prec y \}\)
let \(?xP'y\) = \(\{ i \in Is.\ x \ (P'\ i) \prec y \}\)
let \(?yPx\) = \(\{ i \in Is.\ y \ (P\ i) \prec x \}\)
let \(?yP'x\) = \(\{ i \in Is.\ y \ (P'\ i) \prec x \}\)

from profileP \(xyA\) \(xPy\) \(xIx\) have \(yP'xyPx\): \(?yP'x\) \(\leq\) \(?yPx\)
unfolding strict-pref-def indifferent-pref-def
by (blast dest: rpr-complete)
with finiteIs have \(yP'xyPxC\): \(\text{card } \ ?yP'x \leq \text{card } \ ?yPx\)
by (blast intro: card-mono finite-subset)
from finiteIs \(xPy\) have \(xPyPx'yC\): \(?xPy \leq\) \(?xP'y\)
by (blast intro: card-mono finite-subset)

show \(x \ (MMD\ Is\ P') \prec y\)

proof
from \(xRSCfy\) \(xPyPx'yC\) \(yP'xyPxC\) show \(x \ (MMD\ Is\ P') \preceq y\)
unfolding MMD-def by auto

next
6.3 Everything satisfying May’s conditions is the Method of Majority Decision

Now show that MMD is the only SCF that satisfies these conditions.

Firstly develop some theory about exchanging alternatives $x$ and $y$ in profile $P$.

definition swapAlts :: 'a ⇒ 'a ⇒ 'a where
swapAlts a b u ≡ if u = a then b else if u = b then a else u

lemma swapAlts-in-set-iff: {a, b} ⊆ A ⇒ swapAlts a b u ∈ A ⟷ u ∈ A
unfolding swapAlts-def by (simp split: split-if)

definition swapAltsP :: ('a, 'i) Profile ⇒ 'a ⇒ 'a ⇒ ('a, 'i) Profile where
swapAltsP P a b ≡ (λi. {(u, v). (swapAlts a b u, swapAlts a b v) ∈ P i })

lemma swapAltsP-ab: a (P i) ≤ b ⟷ b (swapAltsP P a b i) ≤ a (P i) ≤ a (swapAltsP P a b i) ≤ b
unfolding $\text{swapAltsP-def swapAlts-def}$ by simp-all

lemma profile-swapAltsP:
  assumes profileP: profile A Is P
     and abA: \{a,b\} \subseteq A
  shows profile A Is (swapAltsP P a b)
proof (rule profileI)
  from profileP show Is \neq \{\} by (rule profile-non-empty)
next
  fix i assume ils: i \in Is
  show rpr A (swapAltsP P a b i)
proof (rule rprI)
    show refl-on A (swapAltsP P a b i)
proof (rule refl-onI)
      from profileP ils abA show swapAltsP P a b i \subseteq A \times A
      unfolding swapAltsP-def by (blast dest: swapAlts-in-set-iff)
      from profileP ils abA show \(\forall x. x \in A \implies (swapAltsP P a b i) \subseteq x\)
      unfolding swapAltsP-def swapAlts-def by auto
      qed
next
  from profileP ils abA show complete A (swapAltsP P a b i)
  unfolding swapAltsP-def
  by (rule completeI, simp, rule rpr-complete[where A=A],
      auto iff: swapAlts-in-set-iff)
next
  from profileP ils show trans (swapAltsP P a b i)
  unfolding swapAltsP-def by (blast dest: rpr-le-trans intro: transI)
  qed
qed

lemma profile-bij-profile:
  assumes profileP: profile A Is P
     and bijf: bij-betw f Is Is
  shows profile A Is (P \circ f)
using bij-betw-onto[OF bijf] profileP
by (rule, auto dest: profile-non-empty)

The locale keeps the conditions in scope for the next few lemmas. Note how weak the constraints on the sets of alternatives and individuals are; clearly there needs to be at least two alternatives and two individuals for conflict to occur, but it is pleasant that the proof uniformly handles the degenerate cases.

locale May =
  fixes A :: 'a set
  fixes Is :: 'i set
  assumes finiteIs: finite Is
  fixes scf :: ('a, 'i) SCF
  assumes SCF: SCF scf A Is universal-domain
     and anonymous: anonymous scf A Is
     and neutral: neutral scf A Is
     and positively-responsive: positively-responsive scf A Is

40
Anonymity implies that, for any pair of alternatives, the social choice rule can only depend on the number of individuals who express any given preference between them. Note we also need iia, implied by neutrality, to restrict attention to alternatives \( x \) and \( y \).

**Lemma anonymous-card:**

**Assumes** profile\( P \): profile \( A \ Is P \)

and profile\( P' \): profile \( A \ Is P' \)

and \( x,y:A \)

\( \text{hasw} [x,y] A \)

and \( \text{yxtally} \): card \{ \( i \in Is. x \ (P \ i) \prec y \} = \text{card} \{ \( i \in Is. x \ (P' \ i) \prec y \} \)

and \( \text{yxtally} \): card \{ \( i \in Is. y \ (P \ i) \prec x \} = \text{card} \{ \( i \in Is. y \ (P' \ i) \prec x \} \)

shows \( x \ (\text{scf} \ P) \preceq y \iff x \ (\text{scf} \ P') \preceq y \)

**Proof**

- let \( ?xPy = \{ \( i \in Is. x \ (P \ i) \prec y \} \)
- let \( ?xP'y = \{ \( i \in Is. x \ (P' \ i) \prec y \} \)
- let \( ?yPz = \{ \( i \in Is. y \ (P \ i) \prec x \} \)
- let \( ?yP'z = \{ \( i \in Is. y \ (P' \ i) \prec x \} \)

have disjP\( _{xy} \): \( (?xPy \cup ?yPz) \not= ?xPy = ?yPz \)

unfolding strict-pref-def by blast

have disjP\( _{xy} \): \( (?xP'y \cup ?yP'z) \not= ?xP'y = ?yP'z \)

unfolding strict-pref-def by blast

from finiteIs yxtally

obtain \( f \) where bijf: bij-betw \( f \ ?xPy \ ?xP'y \)

by - (drule card-eq-bij, auto)

from finiteIs yxtally

obtain \( g \) where bijg: bij-betw \( g \ ?yPz \ ?yP'z \)

by - (drule card-eq-bij, auto)

from bijf bijg disjP\( _{xy} \) disjP\( _{x'y} \)

obtain \( h \)

where bijh: bij-betw \( h \ (\?xPy \cup \?yPz) \ (\?xP'y \cup \?yP'z) \)

and \( \text{hh} \): \( \forall j. j \in (?xPy \cup ?yPz) \Rightarrow h \ j = f \ j \)

and \( \text{hh} \): \( \forall j. j \in (?xPy \cup ?yPz) \Rightarrow h \ j = g \ j \)

using bij-combine[where \( f = f \) and \( g = g \) and \( A = ?xPy \) and \( B = ?xPy \cup ?yPz \) and \( C = ?xP'y \) and \( D = ?xP'y \cup ?yP'z \)]

by auto

from bijh finiteIs

obtain \( h' \) where bijh': bij-betw \( h' \ Is \ Is \)

and \( \text{hh} \): \( \forall j. j \in (?xPy \cup ?yPz) \Rightarrow h' \ j = h \ j \)

and \( \text{hrest} \): \( \forall j. j \in Is - (?xPy \cup ?yPz) \Rightarrow h' \ j \in Is - (?xP'y \cup ?yP'z) \)

by - (drule bij-complete, auto)

from neutral-iia[OF neutral]

have \( x \ (\text{scf} \ (P' \ o h')) \preceq y \iff x \ (\text{scf} \ P) \preceq y \)

proof(rule iiaE)

from \( xyA \) show \( \{ x, y \} \subseteq A \) by simp

next

fix \( i \) assume \( iIs: i \in Is \)

fix \( a \ b \) assume \( ab: a \in \{ x, y \} \ b \in \{ x, y \} \)

from profile\( P \) \( iIs \) have completePi: complete \( A \) \( (P \ i) \) by (auto dest: rprD)

from completePi \( xyA \)

show \( a \ (P \ i) \preceq b \) \iff \( a \ ((P' \ o h') \ i) \preceq b \)

proof(cases rule: complete-ezh)

41
case $xPy$ with profileP profileP' $xyA$ $iIs$ ab $hh'$ $hf$ bijf show ?thesis
  unfolding strict-pref-def bij-betw-def by (simp, blast)
next
case $yPx$ with profileP profileP' $xyA$ $iIs$ ab $hh'$ $hg$ bijf show ?thesis
  unfolding strict-pref-def bij-betw-def by (simp, blast)
next
case $xIy$ with profileP profileP' $xyA$ $iIs$ ab $hrest[\text{where } j=i]$ show ?thesis
  unfolding indifferent-pref-def strict-pref-def bij-betw-def
  by (simp, blast dest: rpr-complete)
qed

Using the previous result and neutrality, it must be the case that if the tallies are tied for alternatives $x$ and $y$ then the social choice function is indifferent between those two alternatives.

lemma anonymous-neutral-indifference:
  assumes profileP: profile $A$ Is $P$
  and $xyA$: basw $[x,y]$ $A$
  and tallyP: card $\{ \ i \in Is. \ x \ (P \ i) \prec y \ \} = \ \text{card} \ \{ \ i \in Is. \ y \ (P \ i) \prec x \ \}$
  shows $x \ (\text{scf } P) \approx y$

proof –
  — Neutrality insists the results for $P$ are symmetrical to those for $\text{swapAltsP } P$.
  from $xyA$
  have $\text{symPP'}: \ x \ (\text{scf } P) \udot y \leftrightarrow y \ (\text{scf } \text{swapAltsP } P \ x \ y) \udot x$
    \land \ (y \ (\text{scf } P) \udot x \leftrightarrow x \ (\text{scf } \text{swapAltsP } P \ x \ y) \udot y)$
    by – (rule neutralD[\text{OF neutral profileP profile-swapAltsP[\text{OF profileP}]},
      simp-all, (rule swapAltsP-ab)+)
  — Anonymity and neutrality insist the results for $P$ are identical to those for $\text{swapAltsP } P$.
  from $xyA$ tallyP have card $\{ \ i \in Is. \ x \ (P \ i) \prec y \ \} = \ \text{card} \ \{ \ i \in Is. \ y \ (P \ i) \prec x \ \}$
    and card $\{ \ i \in Is. \ y \ (P \ i) \prec x \ \} = \ \text{card} \ \{ \ i \in Is. \ y \ (\text{swapAltsP } P \ x \ y) \prec x \ \}$
    unfolding swapAltsP-def swapAlts-def strict-pref-def by simp-all
  with profileP $xyA$ have $\text{idPP'}: \ x \ (\text{scf } P) \udot y \leftrightarrow x \ (\text{scf } \text{swapAltsP } P \ x \ y) \udot y$
    and $y \ (\text{scf } P) \udot x \leftrightarrow y \ (\text{scf } \text{swapAltsP } P \ x \ y) \udot x$
    by – (rule anonymous-card[\text{OF profileP profile-swapAltsP}], clarsimp+)+
  from $xyA$ $\text{SCF-completeD[\text{OF SCF}] profileP symPP'}$ $\text{idPP'}$ show $x \ (\text{scf } P) \approx y$ by (simp, blast)
qed

Finally, if the tallies are not equal then the social choice function must lean towards the one with the higher count due to positive responsiveness.

lemma positively-responsive-prefer-witness:
  assumes profileP: profile $A$ Is $P$
  and $xyA$: basw $[x,y]$ $A$
  and tallyP: card $\{ \ i \in Is. \ x \ (P \ i) \prec y \ \} > \ \text{card} \ \{ \ i \in Is. \ y \ (P \ i) \prec x \ \}$
  obtains $P' k$
  where profile $A$ Is $P'$
  and $\bigwedge i. \ [i \in Is; \ x \ (P' i) \prec y] \ \Rightarrow \ x \ (P i) \prec y$

42
\[ \text{and } \bigwedge_i [i \in Is; x \ (P' \ i) \preceq y] \implies x \ (P \ i) \preceq y \]
\[ \text{and } k \in Is \land x \ (P' \ k) \preceq y \land x \ (P \ k) \prec y \]
\[ \text{and } \text{card } \{i \in Is; x \ (P' \ i) \prec y\} = \text{card } \{i \in Is; y \ (P \ i) \prec x\} \]

\text{proof}

- \text{from } \text{tallyP } \text{obtain } C
  
  \text{where } \text{tallyP'}: \text{card } \{i \in Is; x \ (P \ i) \prec y\} = \text{card } \{i \in Is; y \ (P \ i) \prec x\}
  
  \text{and } C: C \neq \emptyset \subseteq Is
  
  \text{and } C x P y: C \subseteq \{i \in Is; x \ (P \ i) \prec y\}

\text{by} - (\text{drule card-greater[OF finiteIs], auto})

\text{— Add } (b, a) \text{ and close under transitivity.}

\text{let } ?P' = \lambda i. \text{if } i \in C \text{ then } P i \cup \{y, x\} \text{ else } P i

\text{have profile } A \text{ Is } ?P'

\text{proof}

\text{fix } i \text{ assume } iIs: i \in Is

\text{show } rpr A (?P' i)

\text{proof}

- \text{from } profileP iIs \text{ show complete } A (?P' i)
  
  \text{unfolding complete-def by } (\text{simp, blast dest: rpr-complete})

- \text{from } profileP iIs xyA \text{ show refl-on } A (?P' i)
  
  \text{by} - (\text{rule refl-onI, auto})

\text{show trans } (?P' i)

\text{proof(cases } i \in C)

  \text{case } False \text{ with profileP iIs show } ?\text{thesis}
  
  \text{by } (\text{simp, blast dest: rpr-le-trans intro: transI})

  \text{next}

  \text{case } True \text{ with profileP iIs } C \text{ CxyA show } ?\text{thesis}

  \text{unfolding strict-pref-def}

  \text{by} - (\text{rule transI, simp, blast dest: rpr-le-trans rpr-complete})

\text{qed}

\text{next}

\text{from } C \text{ show Is } \neq \emptyset \text{ by blast}

\text{qed}

\text{moreover}

\text{have } \bigwedge_i [i \in Is; x \ (\bar{P'} \ i) \preceq y] \implies x \ (P \ i) \preceq y

\text{unfolding strict-pref-def by } (\text{simp split: split-if-asm})

\text{moreover}

\text{from } profileP C xyA

\text{have } \bigwedge_i [i \in Is; x \ (\bar{P'} \ i) \approx y] \implies x \ (P \ i) \preceq y

\text{unfolding indifferent-pref-def by } (\text{simp split: split-if-asm})

\text{moreover}

\text{from } C \text{ CxyA obtain } k \text{ where } kC: k \in C \text{ and } x P k y: x \ (P \ k) \prec y \text{ by blast}

\text{hence } x \ (\bar{P'} \ k) \approx y \text{ by auto}

\text{with } C k C x P k y \text{ have } k \in Is \land x \ (\bar{P'} \ k) \approx y \land x \ (P \ k) \prec y \text{ by blast}

\text{moreover}

\text{have } \text{card } \{i \in Is; x \ (\bar{P'} \ i) \prec y\} = \text{card } \{i \in Is; y \ (\bar{P'} \ i) \prec x\}
proof
have \{ i \in Is. x (\varphi P')_i \prec y \} = \{ i \in Is. x (\varphi P')_i \prec y \} - C

proof
from C have \{ i \in Is. x (\varphi P')_i \prec y \} \Longrightarrow i \in Is - C
  unfolding indifferent-pref-def strict-pref-def by auto
thus \?thesis by blast
qed

also have \ldots = \{ i \in Is. x (P_i)_\non \prec y \} - C by auto
finally have card \{ i \in Is. x (\varphi P')_i \prec y \} = card \{ i \in Is. y (P_i)_\non \prec x \}
  by simp
with tallyP' have card \{ i \in Is. x (\varphi P')_i \prec y \} = card \{ i \in Is. y (P_i)_\non \prec x \}
  by simp
also have \ldots = card \{ i \in Is. y (\varphi P')_i \prec x \} (is card \?lhs = card \?rhs)

proof
from profileP xyA have \\\\{ i \in Is. y (\varphi P')_i \prec x \} \Longrightarrow y (P_i)_\non \prec x
  unfolding strict-pref-def by (simp split: split-if-asm, blast dest: rpr-complete)
  hence \?rhs \subseteq \?lhs by blast
moreover
from profileP xyA have \\\\{ i \in Is. y (P_i)_\non \prec x \} \Longrightarrow y (\varphi P')_i \prec x
  unfolding strict-pref-def by simp
  hence \?lhs \subseteq \?rhs by blast
ultimately show \?thesis by simp
qed

finally show \?thesis .
qed

ultimately show \?thesis ..
qed

lemma positively-responsive-prefer:
  assumes profileP: profile A Is P
    and xyA: basw [x,y] A
    and tallyP: card \{ i \in Is. x (P_i)_\non \prec y \} > card \{ i \in Is. y (P_i)_\non \prec x \}
  shows x (scf P) \non \prec y

proof
from assms obtain P' k
  where profileP': profile A Is P'
    and F: \\\\{ i \in Is; x (P'_i) \n \prec y \} \Longrightarrow x (P_i)_\non \prec y
    and G: \\\\{ i \in Is; x (P'_i) \n \equiv y \} \Longrightarrow x (P_i)_\non \equiv y
    and pivot: k \in Is \land x (P'_k) \n \equiv y \land x (P_k) \n y
    and cardP': card \{ i \in Is. x (P'_i) \n \prec y \} = card \{ i \in Is. y (P'_i) \n \prec x \}
  by - (drule positively-responsive-prefer-witness, auto)
from profileP' xyA cardP' have x (scf P) \n \equiv y
  by - (rule anonymous-neutral-indifference, auto)
with xyA F G pivot show \?thesis
  by - (rule positively-responsiveD[OF positively-responsive profileP' profileP], auto)
qed

lemma MMD-r2:
  assumes profileP: profile A Is P
    and xyA: basw [x,y] A
shows $x \ (\text{scf} \ P) \preceq y \iff x \ (\text{MMD Is} \ P) \preceq y$

proof (cases rule: linorder-cases)

assume card \{ i \in Is. \ (P i) \prec y \} = card \{ i \in Is. \ (P i) \prec x \}

with profileP xyA show \?thesis

using anonymous-neutral-indifference

unfolding indifferent-pref-def MMD-def by simp

next

assume card \{ i \in Is. \ (P i) \prec y \} > card \{ i \in Is. \ (P i) \prec x \}

with profileP xyA show \?thesis

using positively-responsive-prefer

unfolding strict-pref-def MMD-def by simp

next

assume card \{ i \in Is. \ (P i) \prec y \} < card \{ i \in Is. \ (P i) \prec x \}

with profileP xyA show \?thesis

using positively-responsive-prefer

unfolding strict-pref-def MMD-def by clarsimp

qed

end

May’s original paper [May52] goes on to show that the conditions are independent by exhibiting choice rules that differ from MMD and satisfy the conditions remaining after any particular one is removed. I leave this to future work.

May also wrote a later article [May53] where he shows that the conditions are completely independent, i.e. for every partition of the conditions into two sets, there is a voting rule that satisfies one and not the other.

There are many later papers that characterise MMD with different sets of conditions.

6.4 The Plurality Rule

Goodin and List [GL06] show that May’s original result can be generalised to characterise plurality voting. The following shows that this result is a short step from Sen’s much earlier generalisation.

Plurality voting is a choice function that returns the alternative that receives the most votes, or the set of such alternatives in the case of a tie. Profiles are restricted to those where each individual casts a vote in favour of a single alternative.

type-synonym \('a, 'i\) SVProfile = 'i ⇒ 'a

definition svprofile :: 'a set ⇒ 'i set ⇒ ('a, 'i) SVProfile ⇒ bool where

svprofile A Is F \equiv Is \neq \{} \land F \cdot Is \subseteq A

definition plurality-rule :: 'a set ⇒ 'i set ⇒ ('a, 'i) SVProfile ⇒ 'a set where

plurality-rule A Is F

\equiv \{ x \in A . \forall y \in A. \ card \{ i \in Is . F i = x \} \geq \ card \{ i \in Is . F i = y \} \}

By translating single-vote profiles into RPRs in the obvious way, the choice function arising from MMD coincides with traditional plurality voting.

definition MMD-plurality-rule :: 'a set ⇒ 'i set ⇒ ('a, 'i) Profile ⇒ 'a set where

MMD-plurality-rule A Is P \equiv choiceSet A (MMD Is P)
definition single-vote-to-RPR :: 'a set ⇒ 'a RPR where
  single-vote-to-RPR A a ≡ \{ (a, x) \mid x \in A \} \cup (A - \{a\}) \times (A - \{a\})

lemma single-vote-to-RPR-iff:
  \[ a \in A; x \in A; a \neq x \implies (a \ (single-vote-to-RPR A b) \prec x) \iff (b = a) \]

unfolding single-vote-to-RPR-def strict-pref-def by auto

lemma plurality-rule-equiv:
  plurality-rule A Is F = MMD-plurality-rule A Is (single-vote-to-RPR A ◦ F)

proof
  { fix x y
    have \[ x \in A; y \in A \implies
      (\text{card} \{ i \in Is. F i = y \} \leq \text{card} \{ i \in Is. F i = x \} =
      (\text{card} \{ i \in Is. y \ (single-vote-to-RPR A (F i)) \prec x \}
    \leq \text{card} \{ i \in Is. x \ (single-vote-to-RPR A (F i)) \prec y \})
    \]
  by (cases x=y, auto iff: single-vote-to-RPR-iff)
  }

thus thesis
  unfolding plurality-rule-def MMD-plurality-rule-def choiceSet-def MMD-def
by auto
qed

Thus it is clear that Sen’s generalisation of May’s result applies to this case as well.

Their paper goes on to show how strengthening the anonymity condition gives rise to a
characterisation of approval voting that strictly generalises May’s original theorem. As this
requires some rearrangement of the proof I leave it to future work.

7 Bibliography

References


  Theorem in a restricted informational environment. American Journal of Political

[May52] K. O. May. A set of independent, necessary and sufficient conditions for simple

[May53] K. O. May. A note on the complete independence of the conditions for simple
  majority decision. Econometrica, 21(1), 1953.

[Nip08] Tobias Nipkow. Arrow and Gibbard-Satterthwaite. In Gerwin Klein, Tobias Nipkow,
  net/devel-entries/ArrowImpossibilityGS.shtml, September 2008. Formal proof
development.
[Rou79] R. Routley. Repairing proofs of Arrow’s General Impossibility Theorem and en-
