

Arrow's General Possibility Theorem

Peter Gammie
peteg42 at gmail.com

December 14, 2021

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: <i>hasw</i> , <i>has</i>	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 naive. 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 ⊆ A
    and empty: P {}
    and insert: ∧a F. [finite F; a ∈ A; F ⊆ A; a ∉ F; P F] ⇒ P (insert a F)
  shows P F
proof -
  from ⟨finite F⟩
  have F ⊆ A ⇒ ?thesis
  proof induct
    show P {} by fact
  next
    fix x F
    assume finite F and x ∉ F and
      P: F ⊆ A ⇒ P F and i: insert x F ⊆ A
    show P (insert x F)
    proof (rule insert)
      from i show x ∈ A by blast
      from i have F ⊆ A by blast
      with P show P F .
      show finite F by fact
      show x ∉ F by fact
      show F ⊆ A by fact
    qed
  qed
  with ⟨F ⊆ A⟩ show ?thesis by blast
qed
```

A slight improvement on *List.finite-list* - add *distinct*.

```
lemma finite-list: finite A ⇒ ∃l. set l = A ∧ distinct l
proof (induct rule: finite-induct)
  case (insert x F)
  then obtain l where set l = F ∧ distinct l by auto
  with insert have set (x#l) = insert x F ∧ distinct (x#l) by auto
```

thus ?case by blast
qed auto

2.2 Extra bijection lemmas

lemma *bij-betw-onto*: $\text{bij-betw } f \ A \ B \implies f \ ' \ A = B$ **unfolding** *bij-betw-def* **by** *simp*

lemma *inj-on-UnI*: $\llbracket \text{inj-on } f \ A; \text{inj-on } f \ B; f \ ' \ (A - B) \cap f \ ' \ (B - A) = \{\} \rrbracket \implies \text{inj-on } f \ (A \cup B)$
by (*auto iff: inj-on-Un*)

lemma *card-compose-bij*:

assumes *bijf*: *bij-betw* *f* *A* *A*

shows $\text{card } \{ a \in A. P \ (f \ a) \} = \text{card } \{ a \in A. P \ a \}$

proof –

from *bijf* **have** $T: f \ ' \ \{ a \in A. P \ (f \ a) \} = \{ a \in A. P \ a \}$

unfolding *bij-betw-def* **by** *auto*

from *bijf* **have** $\text{card } \{ a \in A. P \ (f \ a) \} = \text{card } (f \ ' \ \{ a \in A. P \ (f \ a) \})$

unfolding *bij-betw-def* **by** (*auto intro: subset-inj-on card-image[symmetric]*)

with *T* **show** *thesis* **by** *simp*

qed

lemma *card-eq-bij*:

assumes *cardAB*: $\text{card } A = \text{card } B$

and *finiteA*: *finite* *A* **and** *finiteB*: *finite* *B*

obtains *f* **where** *bij-betw* *f* *A* *B*

proof –

from *finiteA* **obtain** *g* **where** *G*: *bij-betw* *g* *A* $\{0..<\text{card } A\}$

by (*blast dest: ex-bij-betw-finite-nat*)

from *finiteB* **obtain** *h* **where** *H*: *bij-betw* *h* $\{0..<\text{card } B\}$ *B*

by (*blast dest: ex-bij-betw-nat-finite*)

from *G* *H* *cardAB* **have** *I*: *inj-on* $(h \circ g)$ *A*

unfolding *bij-betw-def* **by** – (*rule comp-inj-on, simp-all*)

from *G* *H* *cardAB* **have** $(h \circ g) \ ' \ A = B$

unfolding *bij-betw-def* **by** *auto* (*metis image-cong image-image*)

with *I* **have** *bij-betw* $(h \circ g)$ *A* *B*

unfolding *bij-betw-def* **by** *blast*

thus *thesis* ..

qed

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 $\bigwedge x. x \in A \implies h \ x = f \ x$

and $\bigwedge x. x \in B - A \implies h \ x = g \ x$

proof –

let *?h* = $\lambda x. \text{if } x \in A \text{ then } f \ x \text{ else } g \ x$

have *inj-on* *?h* $(A \cup (B - A))$

proof(*rule inj-on-UnI*)

from *bijf* **show** *inj-on* *?h* *A*

by – (*rule inj-onI, 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
ultimately have *bij-betw* ?*h* *B* *D*
and $\bigwedge x. x \in A \implies ?h\ x = f\ x$
and $\bigwedge x. x \in B - A \implies ?h\ x = g\ x$
unfolding *bij-betw-def* **by** *auto*
thus *thesis* ..
qed

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 $\bigwedge x. x \in A \implies f'\ x = f\ x$
and $\bigwedge x. x \in C - A \implies f'\ x \in C - B$
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*: $\bigwedge x. x \in A \implies f'\ x = f\ x$
and *f'g*: $\bigwedge x. x \in C - A \implies f'\ x = g\ x$
by - (*drule bij-combine*, *auto*)
from *f'g bijg* **have** $\bigwedge x. x \in C - A \implies f'\ x \in C - B$
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 ∈ A . P x }
let ?QA = { x ∈ A . Q x }
from finiteA obtain p where P: bij-betw p { 0..<card ?PA } ?PA
  using ex-bij-betw-nat-finite[where M=?PA]
  by (blast intro: finite-subset)
let ?CN = { card ?QA..<card ?PA }
let ?C = p ' ?CN
have card ( { x ∈ A . P x } - ?C ) = card ?QA
proof -
  have nat-add-sub-shuffle:  $\bigwedge x y z. \llbracket (x::nat) > y; x - y = z \rrbracket \implies x - z = y$  by simp
  from P have T: p ' { card ?QA..<card ?PA }  $\subseteq$  ?PA
    unfolding bij-betw-def by auto
  from P have card ?PA - card ?QA = card ?C
    unfolding bij-betw-def
    by (auto iff: card-image subset-inj-on[where A=?CN])
  with c have card ?PA - card ?C = card ?QA by (rule nat-add-sub-shuffle)
  with finiteA P T have card ( ?PA - ?C ) = card ?QA
    unfolding bij-betw-def by (auto iff: finite-subset card-Diff-subset)
  thus ?thesis .
qed
moreover
from P c have ?C  $\neq$  {}
  unfolding bij-betw-def by auto
moreover
from P have ?C  $\subseteq$  { x ∈ A . P x }
  unfolding bij-betw-def by auto
ultimately show 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 *card* function unfortunately satisfies:

$$\text{Finite-Set.card.infinite: } \text{infinite } A \implies \text{card } A = 0$$

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 \wedge distinct xs

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

declare *hasw-def*[simp]

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

lemma *card-has*:
assumes *cardS*: card S = n
shows *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: ¬ has n S
  with distinct-card[symmetric]
  have nxs: ¬ (∃ xs. set xs ⊆ S ∧ distinct xs ∧ card (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 ⇒ card S ≥ n (is ?lhs ⇒ ?rhs)
proof –
  assume ?lhs
  then obtain xs
    where set xs ⊆ S ∧ n = length xs
    and dxs: 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 ⇒ {} ≠ S
  by (clarsimp simp add: has-def)

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

lemma has-notempty-1:
  assumes Sne: S ≠ {}
  shows has 1 S
proof –
  from Sne obtain x where x ∈ S by blast
  hence set [x] ⊆ S ∧ distinct [x] ∧ 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': n' ≤ 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*: $\neg \text{has } n \ S$
 and *nn'*: $n \leq n'$
 shows $\neg \text{has } n' \ S$
 using *h nn'* by (*blast dest: has-le-has*)

lemma *has-eq*:
 assumes *h*: $\text{has } n \ S$
 and *hn'*: $\neg \text{has } (\text{Suc } n) \ S$
 shows $\text{card } S = n$
proof –
 from *h* **obtain** *xs*
 where *xs*: $\text{hasw } xs \ S$ and *lenxs*: $\text{length } xs = n$ by (*unfold has-def, blast*)
 have $\text{set } xs = S$
proof
 from *xs* **show** $\text{set } xs \subseteq S$ by *simp*
next
 show $S \subseteq \text{set } xs$
 proof(*rule ccontr*)
 assume $\neg S \subseteq \text{set } xs$
 then obtain *x* **where** $x \in S \ x \notin \text{set } xs$ by *blast*
 with *lenxs xs* **have** $\text{hasw } (x \ \# \ xs) \ S$ $\text{length } (x \ \# \ xs) = \text{Suc } n$ by *simp-all*
 with *hn'* **show** *False* by (*unfold has-def, blast*)
 qed
qed
 with *xs lenxs* *distinct-card* **show** $\text{card } S = n$ by *auto*
qed

lemma *has-extend-witness*:
 assumes *h*: $\text{has } n \ S$
 shows $\llbracket \text{set } xs \subseteq S; \text{length } xs < n \rrbracket \implies \text{set } xs \subset S$
proof(*induct xs*)
 case *Nil*
 with *h* *has-suc-notempty* **show** *?case* by (*cases n, auto*)
next
 case (*Cons x xs*)
 have $\text{set } (x \ \# \ xs) \neq S$
 proof
 assume *Sxxs*: $\text{set } (x \ \# \ xs) = S$
 hence *finiteS*: *finite* *S* by *auto*
 from *h* **obtain** *xs'*
 where *Sxs'*: $\text{set } xs' \subseteq S$
 and *dlxs'*: $\text{distinct } xs' \wedge \text{length } xs' = n$
 by (*unfold has-def hasw-def, blast*)
 with *distinct-card* **have** $\text{card } (\text{set } xs') = n$ by *auto*
 with *finiteS Sxs'* *card-mono* **have** $\text{card } S \geq n$ by *auto*
 moreover
 from *Sxxs Cons* *card-length*[**where** $xs = x \ \# \ xs$]
 have $\text{card } S < n$ by *auto*

ultimately show *False* by *simp*
qed
with *Cons* show *?case* by *auto*
qed

lemma *has-extend-witness'*:
 $\llbracket \text{has } n \ S; \text{hasw } xs \ S; \text{length } xs < n \rrbracket \implies \exists x. \text{hasw } (x \# \ xs) \ S$
by (*simp*, *blast dest: has-extend-witness*)

lemma *has-witness-two*:
assumes *hasnS*: *has n S*
and *nn'*: $2 \leq n$
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=[]*]
obtain *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'*: $3 \leq n$
shows $\exists x \ y \ z. \text{hasw } [x,y,z] \ S$
proof –
from *nn'* **obtain** *x y* **where** $\text{hasw } [x,y] \ S$
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-contr*:
assumes *finiteS*: *finite S*
and *Sne*: $S \neq \{\}$
and *cardS*: $\text{card } S > 1 \implies \text{False}$
shows $\exists j. S = \{j\}$
proof –
from *cardS Sne card-0-eq*[*OF finiteS*] **have** *Scard*: $\text{card } S = 1$ 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

population Is .

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 $rpr\text{-}eq\text{-}syntax :: 'a \Rightarrow 'a$ RPR $\Rightarrow 'a \Rightarrow bool$ ($- \preceq -$ [50, 1000, 51] 50) **where**
 $x \preceq y \equiv (x, y) \in r$

definition $indifferent\text{-}pref :: 'a \Rightarrow 'a$ RPR $\Rightarrow 'a \Rightarrow bool$ ($- \approx -$ [50, 1000, 51] 50) **where**
 $x \approx y \equiv (x \preceq y \wedge y \preceq x)$

lemma $indifferent\text{-}prefI[intro]: \llbracket x \preceq y; y \preceq x \rrbracket \Longrightarrow x \approx y$
unfolding $indifferent\text{-}pref\text{-}def$ **by** $simp$

lemma $indifferent\text{-}prefD[dest]: x \approx y \Longrightarrow x \preceq y \wedge y \preceq x$
unfolding $indifferent\text{-}pref\text{-}def$ **by** $simp$

definition $strict\text{-}pref :: 'a \Rightarrow 'a$ RPR $\Rightarrow 'a \Rightarrow bool$ ($- \prec -$ [50, 1000, 51] 50) **where**
 $x \prec y \equiv (x \preceq y \wedge \neg(y \preceq x))$

lemma $strict\text{-}pref\text{-}def\text{-}irrefl[simp]: \neg(x \prec x)$ **unfolding** $strict\text{-}pref\text{-}def$ **by** $blast$

lemma $strict\text{-}prefI[intro]: \llbracket x \preceq y; \neg(y \preceq x) \rrbracket \Longrightarrow x \prec y$
unfolding $strict\text{-}pref\text{-}def$ **by** $simp$

Traditionally, $x \preceq y$ would be written $x R y$, $x \approx y$ as $x I y$ and $x \prec y$ 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 $\Rightarrow 'a$ RPR $\Rightarrow bool$ **where**
 $complete$ A $r \equiv (\forall x \in A. \forall y \in A - \{x\}. x \preceq y \vee y \preceq x)$

lemma $completeI[intro]: (\bigwedge x y. \llbracket x \in A; y \in A; x \neq y \rrbracket \Longrightarrow x \preceq y \vee y \preceq x) \Longrightarrow complete$ A r
unfolding $complete\text{-}def$ **by** $auto$

lemma $completeD[dest]: \llbracket complete$ A $r; x \in A; y \in A; x \neq y \rrbracket \Longrightarrow x \preceq y \vee y \preceq x$
unfolding $complete\text{-}def$ **by** $auto$

lemma $complete\text{-}less\text{-}not: \llbracket complete$ A $r; hasw$ $[x,y]$ $A; \neg x \prec y \rrbracket \Longrightarrow y \preceq x$
unfolding $complete\text{-}def$ $strict\text{-}pref\text{-}def$ **by** $auto$

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

lemma *complete-exh*:
assumes *complete* $A \ r$
and *hasw* $[x,y] \ A$
obtains $(xPy) \ x \ r \prec y$
 $\quad | \ (yPx) \ y \ r \prec x$
 $\quad | \ (xIy) \ x \ r \approx y$
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*:
 $\llbracket \text{complete } A \ r; \text{refl-on } A \ r; x \in A; y \in A \rrbracket \implies x \ r \preceq y \vee y \ r \preceq x$
unfolding *complete-def* **by** *auto*

definition *irrefl* :: $'a \ \text{set} \Rightarrow 'a \ \text{RPR} \Rightarrow \text{bool}$ **where**
 $\text{irrefl } A \ r \equiv r \subseteq A \times A \wedge (\forall x \in A. \neg x \ r \preceq x)$

lemma *irreflI* $[intro]$: $\llbracket r \subseteq A \times A; \bigwedge x. x \in A \implies \neg x \ r \preceq x \rrbracket \implies \text{irrefl } A \ r$
unfolding *irrefl-def* **by** *simp*

lemma *irreflD* $[dest]$: $\llbracket \text{irrefl } A \ r; (x, y) \in r \rrbracket \implies \text{hasw } [x,y] \ A$
unfolding *irrefl-def* **by** *auto*

lemma *irreflD'* $[dest]$:
 $\llbracket \text{irrefl } A \ r; r \neq \{\} \rrbracket \implies \exists x \ y. \text{hasw } [x,y] \ A \wedge (x, y) \in r$
unfolding *irrefl-def* **by** *auto*

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

definition *rpr* :: $'a \ \text{set} \Rightarrow 'a \ \text{RPR} \Rightarrow \text{bool}$ **where**
 $\text{rpr } A \ r \equiv \text{complete } A \ r \wedge \text{refl-on } A \ r \wedge \text{trans } r$

lemma *rprI* $[intro]$: $\llbracket \text{complete } A \ r; \text{refl-on } A \ r; \text{trans } r \rrbracket \implies \text{rpr } A \ r$
unfolding *rpr-def* **by** *simp*

lemma *rprD*: $\text{rpr } A \ r \implies \text{complete } A \ r \wedge \text{refl-on } A \ r \wedge \text{trans } r$
unfolding *rpr-def* **by** *simp*

lemma *rpr-in-set* $[dest]$: $\llbracket \text{rpr } A \ r; x \ r \preceq y \rrbracket \implies \{x,y\} \subseteq A$
unfolding *rpr-def refl-on-def* **by** *auto*

lemma *rpr-refl* $[dest]$: $\llbracket \text{rpr } A \ r; x \in A \rrbracket \implies x \ r \preceq x$
unfolding *rpr-def* **by** *blast*

lemma *rpr-less-not*: $\llbracket \text{rpr } A \ r; \text{hasw } [x,y] \ A; \neg x \ r \prec y \rrbracket \implies y \ r \preceq x$
unfolding *rpr-def* **by** (*auto simp add: complete-less-not*)

lemma *rpr-less-imp-le* $[simp]$: $\llbracket x \ r \prec y \rrbracket \implies x \ r \preceq y$

unfolding *strict-pref-def* **by** *simp*

lemma *rpr-less-imp-neq*[*simp*]: $\llbracket x \prec y \rrbracket \implies x \neq y$
unfolding *strict-pref-def* **by** *blast*

lemma *rpr-less-trans*[*trans*]: $\llbracket x \prec y; y \prec z; rpr\ A\ r \rrbracket \implies x \prec z$
unfolding *rpr-def strict-pref-def trans-def* **by** *blast*

lemma *rpr-le-trans*[*trans*]: $\llbracket x \preceq y; y \preceq z; rpr\ A\ r \rrbracket \implies x \preceq z$
unfolding *rpr-def trans-def* **by** *blast*

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

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

lemma *rpr-complete*: $\llbracket rpr\ A\ r; x \in A; y \in A \rrbracket \implies x \preceq y \vee y \preceq x$
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* \Rightarrow *'a* *RPR*

definition *profile* :: *'a set* \Rightarrow *'i set* \Rightarrow (*'a, 'i*) *Profile* \Rightarrow *bool* **where**
profile *A Is P* \equiv *Is* \neq $\{\}$ \wedge ($\forall i \in Is. rpr\ A\ (P\ i)$)

lemma *profileI*[*intro*]: $\llbracket \bigwedge i. i \in Is \implies rpr\ A\ (P\ i); Is \neq \{\} \rrbracket \implies profile\ A\ Is\ P$
unfolding *profile-def* **by** *simp*

lemma *profile-rprD*[*dest*]: $\llbracket profile\ A\ Is\ P; i \in Is \rrbracket \implies rpr\ A\ (P\ i)$
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* \Rightarrow *'a RPR* \Rightarrow *'a set* **where**
choiceSet *A r* \equiv $\{ x \in A . \forall y \in A. x \preceq y \}$

definition *choiceFn* :: *'a set* \Rightarrow *'a RPR* \Rightarrow *bool* **where**
choiceFn *A r* \equiv $\forall A' \subseteq A. A' \neq \{\} \longrightarrow choiceSet\ A'\ r \neq \{\}$

lemma *choiceSetI*[*intro*]:
 $\llbracket x \in A; \bigwedge y. y \in A \implies x \prec y \rrbracket \implies x \in \text{choiceSet } A \ r$
unfolding *choiceSet-def* **by** *simp*

lemma *choiceFnI*[*intro*]:
 $(\bigwedge A'. \llbracket A' \subseteq A; A' \neq \{\} \rrbracket \implies \text{choiceSet } A' \ r \neq \{\}) \implies \text{choiceFn } A \ r$
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 $r \equiv \forall x \ y \ z. x \prec y \wedge y \prec z \longrightarrow x \prec z$

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

lemma *quasi-transD*: $\llbracket x \prec y; y \prec z; \text{quasi-trans } r \rrbracket \implies x \prec z$
unfolding *quasi-trans-def* **by** *blast*

lemma *trans-imp-quasi-trans*: $\text{trans } r \implies \text{quasi-trans } r$
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** $B: B \subseteq A \ B \neq \{\}$
with *finite-subset* *finiteA* **have** *finiteB*: *finite* B **by** *auto*
from *finiteB* B **show** $\text{choiceSet } B \ r \neq \{\}$
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*: $B \neq \{\} \implies \text{choiceSet } B \ r \neq \{\}$ **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 $a \prec b \vee b \prec a$ **unfolding** *choiceSet-def* *strict-pref-def* **by** *blast*

thus *?thesis*

proof

assume *ab*: $b \prec a$

```

  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 qt 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
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 *iii*.

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

$SWF\ swf\ A\ Is\ Pcond \equiv (\forall P. Pcond\ A\ Is\ P \longrightarrow rpr\ A\ (swf\ P))$

lemma $SWF\text{-}rpr[dest]$: $\llbracket SWF\ swf\ A\ Is\ Pcond; Pcond\ A\ Is\ P \rrbracket \Longrightarrow rpr\ A\ (swf\ P)$
unfolding $SWF\text{-}def$ **by** $simp$

3.6 General Properties of an SCF

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

definition $universal\text{-}domain :: 'a\ set \Rightarrow 'i\ set \Rightarrow ('a, 'i)\ Profile \Rightarrow bool$ **where**
 $universal\text{-}domain\ A\ Is\ P \equiv profile\ A\ Is\ P$

declare $universal\text{-}domain\text{-}def[simp]$

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

definition

$weak\text{-}pareto :: ('a, 'i)\ SCF \Rightarrow 'a\ set \Rightarrow 'i\ set \Rightarrow ('a\ set \Rightarrow 'i\ set \Rightarrow ('a, 'i)\ Profile \Rightarrow bool) \Rightarrow bool$
where

$weak\text{-}pareto\ scf\ A\ Is\ Pcond \equiv$
 $(\forall P\ x\ y. Pcond\ A\ Is\ P \wedge x \in A \wedge y \in A \wedge (\forall i \in Is. x\ (P\ i) \prec y) \longrightarrow x\ (scf\ P) \prec y)$

lemma $weak\text{-}paretoI[intro]$:

$(\bigwedge P\ x\ y. \llbracket Pcond\ A\ Is\ P; x \in A; y \in A; \bigwedge i. i \in Is \Longrightarrow x\ (P\ i) \prec y \rrbracket \Longrightarrow x\ (scf\ P) \prec y)$
 $\Longrightarrow weak\text{-}pareto\ scf\ A\ Is\ Pcond$

unfolding $weak\text{-}pareto\text{-}def$ **by** $simp$

lemma $weak\text{-}paretoD$:

$\llbracket weak\text{-}pareto\ scf\ A\ Is\ Pcond; Pcond\ A\ Is\ P; x \in A; y \in A;$
 $(\bigwedge i. i \in Is \Longrightarrow x\ (P\ i) \prec y) \rrbracket \Longrightarrow x\ (scf\ P) \prec y$

unfolding $weak\text{-}pareto\text{-}def$ **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 $iaa :: ('a, 'i)\ SCF \Rightarrow 'a\ set \Rightarrow 'i\ set \Rightarrow bool$ **where**

$iaa\ scf\ S\ Is \equiv$
 $(\forall P\ P'\ x\ y. profile\ S\ Is\ P \wedge profile\ S\ Is\ P'$
 $\wedge x \in S \wedge y \in S$
 $\wedge (\forall i \in Is. ((x\ (P\ i) \preceq y) \longleftrightarrow (x\ (P'\ i) \preceq y)) \wedge ((y\ (P\ i) \preceq x) \longleftrightarrow (y\ (P'\ i) \preceq x)))$
 $\longrightarrow ((x\ (scf\ P) \preceq y) \longleftrightarrow (x\ (scf\ P') \preceq y)))$

lemma $iaaI[intro]$:

$(\bigwedge P\ P'\ x\ y.$
 $\llbracket profile\ S\ Is\ P; profile\ S\ Is\ P';$
 $x \in S; y \in S;$
 $\bigwedge i. i \in Is \Longrightarrow ((x\ (P\ i) \preceq y) \longleftrightarrow (x\ (P'\ i) \preceq y)) \wedge ((y\ (P\ i) \preceq x) \longleftrightarrow (y\ (P'\ i) \preceq x))$
 $\rrbracket \Longrightarrow ((x\ (swf\ P) \preceq y) \longleftrightarrow (x\ (swf\ P') \preceq y)))$
 $\Longrightarrow iaa\ swf\ S\ Is$

unfolding $iaa\text{-}def$ **by** $simp$

lemma $iaaE$:

\llbracket *iaa swf* S Is ;
 $\{x, y\} \subseteq S$;
 $a \in \{x, y\}; b \in \{x, y\}$;
 $\bigwedge i a b. \llbracket a \in \{x, y\}; b \in \{x, y\}; i \in Is \rrbracket \implies (a \text{ (} P' i \text{)} \preceq b) \longleftrightarrow (a \text{ (} P i \text{)} \preceq b)$;
profile S Is P ; *profile* S Is P' \rrbracket
 $\implies (a \text{ (swf } P \text{)} \preceq b) \longleftrightarrow (a \text{ (swf } P' \text{)} \preceq b)$
unfolding *iaa-def* **by** (*simp*, *blast*)

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 *semidecisive* $:: ('a, 'i) SCF \implies 'a \text{ set} \implies 'i \text{ set} \implies 'i \text{ set} \implies 'a \implies 'a \implies \text{bool}$ **where**
semidecisive scf A Is C x $y \equiv$
 $C \subseteq Is \wedge (\forall P. \text{profile } A \text{ } Is \text{ } P \wedge (\forall i \in C. x \text{ (} P i \text{)} \prec y) \wedge (\forall i \in Is - C. y \text{ (} P i \text{)} \prec x)$
 $\longrightarrow x \text{ (scf } P \text{)} \prec y)$

lemma *semidecisiveI*[*intro*]:

$\llbracket C \subseteq Is$;
 $\bigwedge P. \llbracket \text{profile } A \text{ } Is \text{ } P; \bigwedge i. i \in C \implies x \text{ (} P i \text{)} \prec y; \bigwedge i. i \in Is - C \implies y \text{ (} P i \text{)} \prec x \rrbracket$
 $\implies x \text{ (scf } P \text{)} \prec y \rrbracket \implies \text{semidecisive scf } A \text{ } Is \text{ } C \text{ } x \text{ } y$
unfolding *semidecisive-def* **by** *simp*

lemma *semidecisive-coalitionD*[*dest*]: *semidecisive scf* A Is C x $y \implies C \subseteq Is$
unfolding *semidecisive-def* **by** *simp*

lemma *sd-refl*: $\llbracket C \subseteq Is; C \neq \{\} \rrbracket \implies \text{semidecisive scf } A \text{ } Is \text{ } C \text{ } x \text{ } x$
unfolding *semidecisive-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* $:: ('a, 'i) SCF \implies 'a \text{ set} \implies 'i \text{ set} \implies 'i \text{ set} \implies 'a \implies 'a \implies \text{bool}$ **where**
decisive scf A Is C x $y \equiv$
 $C \subseteq Is \wedge (\forall P. \text{profile } A \text{ } Is \text{ } P \wedge (\forall i \in C. x \text{ (} P i \text{)} \prec y) \longrightarrow x \text{ (scf } P \text{)} \prec y)$

lemma *decisiveI*[*intro*]:

$\llbracket C \subseteq Is; \bigwedge P. \llbracket \text{profile } A \text{ } Is \text{ } P; \bigwedge i. i \in C \implies x \text{ (} P i \text{)} \prec y \rrbracket \implies x \text{ (scf } P \text{)} \prec y \rrbracket$
 $\implies \text{decisive scf } A \text{ } Is \text{ } C \text{ } x \text{ } y$
unfolding *decisive-def* **by** *simp*

lemma *d-imp-sd*: *decisive scf* A Is C x $y \implies \text{semidecisive scf } A \text{ } Is \text{ } C \text{ } x \text{ } y$
unfolding *decisive-def* **by** (*rule semidecisiveI*, *blast+*)

lemma *decisive-coalitionD*[*dest*]: *decisive scf* A Is C x $y \implies C \subseteq Is$
unfolding *decisive-def* **by** *simp*

Anyone is trivially decisive for x against x .

lemma *d-refl*: $\llbracket C \subseteq Is; C \neq \{\} \rrbracket \implies \text{decisive scf } A \text{ } Is \text{ } C \text{ } x \text{ } x$

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 \Rightarrow 'a set \Rightarrow 'i set \Rightarrow 'i \Rightarrow bool **where**
dictator scf A Is j $\equiv j \in Is \wedge (\forall x \in A. \forall y \in A. \text{decisive scf A Is } \{j\} x y)$

lemma *dictatorI[intro]*:

$\llbracket j \in Is; \bigwedge x y. \llbracket x \in A; y \in A \rrbracket \implies \text{decisive scf A Is } \{j\} x y \rrbracket \implies \text{dictator scf A Is } j$

unfolding *dictator-def* **by** *simp*

lemma *dictator-individual[dest]*: *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 *ii*a 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 $x (P' j) \prec y \wedge y (P' j) \prec z$

and $\bigwedge i. i \neq j \implies y (P' i) \prec x \wedge y (P' i) \prec z \wedge ((x (P' i) \preceq z) = (x (P i) \preceq z)) \wedge ((z (P' i) \preceq x) = (z (P i) \preceq x))$

proof

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

$$\begin{aligned} & \cup \{ (z, u) \mid u. u \in A - \{x, y\} \} \\ \text{else } & \{ (y, u) \mid u. u \in A \} \\ & \cup \{ (x, u) \mid u. u \in A - \{y, z\} \} \\ & \cup \{ (z, u) \mid u. u \in A - \{x, y\} \} \\ & \cup (\text{if } x (P i) \preceq z \text{ then } \{(x, z)\} \text{ else } \{\}) \\ & \cup (\text{if } z (P i) \preceq x \text{ then } \{(z, x)\} \text{ else } \{\})) \\ & \cup (A - \{x, y, z\}) \times (A - \{x, y, z\}) \end{aligned}$$

show *profile A Is ?P'*
proof
fix *i assume iIs: i ∈ 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 ≠ j .*
show *?thesis*
proof
from *iIs 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 *iIs 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 ≠ {} by (rule profile-non-empty)*
qed
from *has3A*
show $x (P' j) \prec y \wedge y (P' j) \prec z$
and $\bigwedge i. i \neq j \implies y (P' i) \prec x \wedge y (P' i) \prec z \wedge ((x (P' i) \preceq z) = (x (P i) \preceq z)) \wedge ((z (P' i) \preceq x) = (z (P i) \preceq x))$
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 *ia: ia 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} ⊆ Is by blast*

fix *P*

assume *profileP: profile A Is P*

and *jxP: ⋀ i. i ∈ {j} ⟹ x (P i) < z*

from *has3A profileP jIs*

obtain *P'*

where $\text{profile}P'$: $\text{profile } A \text{ Is } P'$
and $\text{jxyz}P'$: $x \text{ (} P' j \text{)} \prec y \text{ (} P' j \text{)} \prec z$
and $\text{ixyz}P'$: $\bigwedge i. i \neq j \longrightarrow y \text{ (} P' i \text{)} \prec x \wedge y \text{ (} P' i \text{)} \prec z \wedge ((x \text{ (} P' i \text{)} \preceq z) = (x \text{ (} P' i \text{)} \preceq z)) \wedge ((z \text{ (} P' i \text{)} \preceq x) = (z \text{ (} P' i \text{)} \preceq x))$
by $-$ (*rule decisive1-witness, blast+*)
from iaa **have** $\bigwedge a b. \llbracket a \in \{x, z\}; b \in \{x, z\} \rrbracket \Longrightarrow (a \text{ (} \text{swf } P \text{)} \preceq b) = (a \text{ (} \text{swf } P' \text{)} \preceq b)$
proof(*rule iiaE*)
from has3A **show** $\{x, z\} \subseteq A$ **by** *simp*
next
fix i **assume** $iIs: i \in Is$
fix $a b$ **assume** $ab: a \in \{x, z\} b \in \{x, z\}$
show $(a \text{ (} P' i \text{)} \preceq b) = (a \text{ (} P' i \text{)} \preceq b)$
proof(*cases i = j*)
case *False*
with $ab \text{ iIs } \text{ixyz}P' \text{ profile}P \text{ profile}P' \text{ has3A}$
show *?thesis unfolding profile-def by auto*
next
case *True*
from $\text{profile}P' \text{ jIs } \text{jxyz}P'$ **have** $x \text{ (} P' j \text{)} \prec z$
by (*auto dest: rpr-less-trans*)
with *True ab iIs jxzP profileP profileP' has3A*
show *?thesis unfolding profile-def strict-pref-def by auto*
qed
qed (*simp-all add: profileP profileP'*)
moreover **have** $x \text{ (} \text{swf } P' \text{)} \prec z$
proof $-$
from $\text{profile}P' \text{ sd } \text{jxyz}P' \text{ ixyz}P'$ **have** $x \text{ (} \text{swf } P' \text{)} \prec y$ **by** (*simp add: semidecisive-def*)
moreover
from $\text{jxyz}P' \text{ ixyz}P'$ **have** $\bigwedge i. i \in Is \Longrightarrow y \text{ (} P' i \text{)} \prec z$ **by** (*case-tac i=j, auto*)
with $\text{wp } \text{profile}P' \text{ has3A}$ **have** $y \text{ (} \text{swf } P' \text{)} \prec z$ **by** (*auto dest: weak-paretoD*)
moreover **note** $\text{SWF-rpr}[OF \text{ swf}] \text{ profile}P'$
ultimately **show** $x \text{ (} \text{swf } P' \text{)} \prec z$
unfolding *universal-domain-def* **by** (*blast dest: rpr-less-trans*)
qed
ultimately **show** $x \text{ (} \text{swf } P \text{)} \prec z$ **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 *decisive2-witness*:

assumes $\text{has3A}: \text{hasw } [x, y, z] A$
and $\text{profile}P'$: $\text{profile } A \text{ Is } P'$
and $\text{jIs}: j \in Is$
obtains P'
where $\text{profile } A \text{ Is } P'$
and $z \text{ (} P' j \text{)} \prec x \wedge x \text{ (} P' j \text{)} \prec y$
and $\bigwedge i. i \neq j \Longrightarrow z \text{ (} P' i \text{)} \prec x \wedge y \text{ (} P' i \text{)} \prec x \wedge ((y \text{ (} P' i \text{)} \preceq z) = (y \text{ (} P' i \text{)} \preceq z)) \wedge ((z \text{ (} P' i \text{)} \preceq y) = (z \text{ (} P' i \text{)} \preceq y))$
proof

```

let ?P' = λi. (if i = j then ({ (z, u) | u. u ∈ A }
    ∪ { (x, u) | u. u ∈ A - {z} }
    ∪ { (y, u) | u. u ∈ A - {x,z} })
  else ({ (z, u) | u. u ∈ A - {y} }
    ∪ { (y, u) | u. u ∈ A - {z} }
    ∪ { (x, u) | u. u ∈ A - {y,z} }
    ∪ (if y (P i) ≤ z then {(y,z)} else {})
    ∪ (if z (P i) ≤ y then {(z,y)} else {})))
  ∪ (A - {x,y,z}) × (A - {x,y,z})
show profile A Is ?P'
proof
  fix i assume iIs: i ∈ 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 ≠ j .
    show ?thesis
    proof
      from iIs profileP have complete A (P i) by (auto simp add: rpr-def)
      with ij show complete A (?P' i) by (simp add: complete-def, blast)
      from iIs 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
  show Is ≠ {} by (rule profile-non-empty[OF profileP])
qed
from has3A
show z (?P' j) < x ∧ x (?P' j) < y
  and ∧i. i ≠ j ⇒ z (?P' i) < x ∧ y (?P' i) < x ∧ ((y (?P' i) ≤ z) = (y (P i) ≤ z)) ∧ ((z (?P' i) ≤
y) = (z (P i) ≤ y))
  unfolding strict-pref-def by auto
qed

```

lemma *decisive2*:

```

assumes has3A: hasw [x,y,z] A
  and uia: uia 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} z y
proof
  from sd show jIs: {j} ⊆ Is by blast
  fix P
  assume profileP: profile A Is P
  and jyzP: ∧i. i ∈ {j} ⇒ z (P i) < y
  from has3A profileP jIs
  obtain P'
  where profileP': profile A Is P'

```

```

    and  $jxyzP'$ :  $z \prec_{(P' j)} x \prec_{(P' j)} y$ 
    and  $ixyzP'$ :  $\bigwedge i. i \neq j \rightarrow z \prec_{(P' i)} x \wedge y \prec_{(P' i)} x \wedge ((y \preceq_{(P' i)} z) = (y \preceq_{(P i)} z)) \wedge ((z \preceq_{(P' i)} y) = (z \preceq_{(P i)} y))$ 
    by - (rule decisive2-witness, blast+)
    from  $ia$  have  $\bigwedge a b. \llbracket a \in \{y, z\}; b \in \{y, z\} \rrbracket \implies (a \preceq_{(swf P)} b) = (a \preceq_{(swf P')} b)$ 
    proof(rule  $iaE$ )
      from  $has3A$  show  $\{y, z\} \subseteq A$  by simp
    next
      fix  $i$  assume  $iIs: i \in Is$ 
      fix  $a b$  assume  $ab: a \in \{y, z\} b \in \{y, z\}$ 
      show  $(a \preceq_{(P' i)} b) = (a \preceq_{(P i)} b)$ 
      proof(cases  $i = j$ )
        case False
          with  $ab iIs ixyzP'$  profileP profileP' has3A
          show ?thesis unfolding profile-def by auto
        case True
          from profileP'  $jIs jxyzP'$  have  $z \prec_{(P' j)} y$ 
            by (auto dest: rpr-less-trans)
          with True  $ab iIs jyzP$  profileP profileP' has3A
          show ?thesis unfolding profile-def strict-pref-def by auto
      qed
    qed (simp-all add: profileP profileP')
  moreover have  $z \prec_{(swf P')} y$ 
  proof -
    from profileP'  $sd jxyzP'$   $ixyzP'$  have  $x \prec_{(swf P')} y$  by (simp add: semidecisive-def)
  moreover
    from  $jxyzP'$   $ixyzP'$  have  $\bigwedge i. i \in Is \implies z \prec_{(P' i)} x$  by (case-tac  $i=j$ , auto)
  with  $wp$  profileP' has3A have  $z \prec_{(swf P')} x$  by (auto dest: weak-paretoD)
  moreover note  $SWF$ -rpr[ $OF swf$ ] profileP'
  ultimately show  $z \prec_{(swf P')} y$ 
    unfolding universal-domain-def by (blast dest: rpr-less-trans)
  qed
  ultimately show  $z \prec_{(swf P)} y$  unfolding 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  $ia: ia swf A Is$ 
    and  $swf: SWF swf A Is universal-domain$ 
    and  $wp: weak-pareto swf A Is universal-domain$ 
    and  $sd: semidecisive swf A Is \{j\} x z$ 
shows  $decisive swf A Is \{j\} y z$ 
using  $has3A$   $decisive2[OF - ia swf wp sd]$  by (simp, blast)

```

lemma *decisive4*:

```

assumes  $has3A: hasw [x,y,z] A$ 
    and  $ia: ia swf A Is$ 
    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 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])+

qed

lemma *j-decisive-xy*:

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
and uv: hasw [u,v] {x,y,z}
shows decisive swf A Is {j} u v
using uv decisive1[OF has3A iia swf wp sd]
      decisive2[OF has3A iia swf wp sd]
      decisive5[OF has3A iia swf wp sd]
      decisive7[OF has3A iia swf wp sd]
by (simp, blast)

```

lemma *j-decisive*:

```

assumes has3A: has 3 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 xyA: hasw [x,y] A
and sd: semidecisive swf A Is {j} x y
and uv: hasw [u,v] A
shows decisive swf A Is {j} u v
proof –
from has-extend-witness'[OF has3A xyA]
obtain z where xyzA: hasw [x,y,z] A by auto
{
  assume ux: u = x and vy: v = y
  with xyzA iia swf wp sd have ?thesis by (auto intro: j-decisive-xy)
}
moreover
{
  assume ux: u = x and vNEy: v ≠ y
  with uv xyA iia swf wp sd have ?thesis by(auto intro: j-decisive-xy[of x y])
}
moreover
{
  assume uy: u = y and vx: v = x
  with xyzA iia swf wp sd have ?thesis by (auto intro: j-decisive-xy)
}
moreover
{
  assume uy: u = y and vNEx: v ≠ x
  with uv xyA iia swf wp sd have ?thesis by (auto intro: j-decisive-xy)
}
moreover
{
  assume uNExy: u ∉ {x,y} and vx: v = x
  with uv xyA iia swf wp sd have ?thesis by (auto intro: j-decisive-xy[of x y])
}
moreover
{
  assume uNExy: u ∉ {x,y} and vy: v = y
  with uv xyA iia swf wp sd have ?thesis by (auto intro: j-decisive-xy[of x y])
}
moreover

```

```

{
  assume  $uNExy: u \notin \{x,y\}$  and  $vNExy: v \notin \{x,y\}$ 
  with  $w xyA$  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 sdxu: semidecisive swf A Is {j} x u by (rule d-imp-sd)
  with  $uNExy vNExy w xyA$  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: has 3 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 uv: 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 uv 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  $Vs: Is = V1 \cup V2 \cup V3$ 

```

```

     $\wedge V1 \cap V2 = \{\} \wedge V1 \cap V3 = \{\} \wedge V2 \cap V3 = \{\}$ 

```

```

  and  $Is: Is \neq \{\}$ 

```

```

obtains  $P$ 

```

```

  where profile A Is P

```

```

  and  $\forall i \in V1. x (P i) \prec y \wedge y (P i) \prec z$ 

```

```

  and  $\forall i \in V2. z (P i) \prec x \wedge x (P i) \prec y$ 

```

```

  and  $\forall i \in V3. y (P i) \prec z \wedge z (P i) \prec x$ 

```

proof

```

let  $?P =$ 

```

```

   $\lambda i. (if i \in V1 then (\{ (x, u) \mid u. u \in A \})$ 

```

$$\cup \{ (y, u) \mid u. u \in A \wedge u \neq x \}$$

$$\cup \{ (z, u) \mid u. u \in A \wedge u \neq x \wedge u \neq y \})$$

else

if $i \in V2$ *then* $(\{ (z, u) \mid u. u \in A \}$

$$\cup \{ (x, u) \mid u. u \in A \wedge u \neq z \}$$

$$\cup \{ (y, u) \mid u. u \in A \wedge u \neq x \wedge u \neq z \})$$

else $(\{ (y, u) \mid u. u \in A \}$

$$\cup \{ (z, u) \mid u. u \in A \wedge u \neq y \}$$

$$\cup \{ (x, u) \mid u. u \in A \wedge u \neq y \wedge u \neq z \})$$

$$\cup \{ (u, v) \mid u. v. u \in A - \{x, y, z\} \wedge v \in A - \{x, y, z\}\})$$

show *profile* A *Is* $?P$

proof

fix i **assume** $iIs: i \in Is$

show *rpr* A $(?P i)$

proof

show *complete* A $(?P i)$ **by** (*simp add: complete-def, blast*)

from *has3A* iIs **show** *refl-on* A $(?P i)$ **by** $-$ (*simp, blast*)

from *has3A* iIs **show** *trans* $(?P i)$ **by** (*clarsimp simp add: trans-def*)

qed

next

from Is **show** $Is \neq \{\}$.

qed

from *has3A* Vs

show $\forall i \in V1. x$ $(?P i) \prec y \wedge y$ $(?P i) \prec z$

and $\forall i \in V2. z$ $(?P i) \prec x \wedge x$ $(?P i) \prec y$

and $\forall i \in V3. y$ $(?P i) \prec z \wedge z$ $(?P i) \prec x$

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

qed

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

lemma *sd-exists*:

assumes *has3A*: *has 3 A*

and *finiteIs*: *finite Is*

and *twoIs*: *has 2 Is*

and *ia*: *ia swf A Is*

and *swf*: *SWF swf A Is universal-domain*

and *wp*: *weak-pareto swf A Is universal-domain*

shows $\exists j u v. hasw [u, v] A \wedge semidecisive swf A Is \{j\} u v$

proof $-$

let $?P = \lambda S. S \subseteq Is \wedge S \neq \{\} \wedge (\exists u v. hasw [u, v] A \wedge semidecisive swf A Is S u v)$

obtain $u v$ **where** $uwA: hasw [u, v] A$

using *has-witness-two[OF has3A]* **by** *auto*

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

hence *decisive swf A Is Is u v*

by $-$ (*rule, auto intro: weak-paretoD[OF wp]*)

hence *semidecisive swf A Is Is u v* **by** (*rule d-imp-sd*)

with uwA *twoIs* *has-suc-notempty[where n=1]* *nat-2[symmetric]*

have $?P Is$ **by** *auto*

$-$ Obtain a minimally-sized semi-decisive set.

from *ex-has-least-nat[where P=?P and m=card, OF this]*

obtain $V x y$ **where** $VIs: V \subseteq Is$
and $Vnotempty: V \neq \{\}$
and $xyA: hasw [x,y] A$
and $Vsd: semidecisive swf A Is V x y$
and $Vmin: \bigwedge V'. ?P V' \implies card V \leq card V'$
by *blast*
from VIs *finiteIs* **have** $Vfinite: finite V$ **by** (*rule finite-subset*)
— Show that minimal set contains a single individual.
from $Vfinite$ $Vnotempty$ **have** $\exists j. V = \{j\}$
proof(*rule finite-set-singleton-contr*)
assume $Vcard: 1 < card V$
then obtain j **where** $jV: \{j\} \subseteq V$
using *has-extend-witness*[**where** $xs=[]$, *OF card-has*[**where** $n=card V$]] **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* $Vcard$
have $V2card: card ?V2 > 0 \wedge card ?V2 < card V$ **by** *auto*
hence $V2notempty: \{\} \neq ?V2$ **by** *auto*
from jV VIs
have $jV2V3: Is = ?V1 \cup ?V2 \cup ?V3 \wedge ?V1 \cap ?V2 = \{\} \wedge ?V1 \cap ?V3 = \{\} \wedge ?V2 \cap ?V3 =$
 $\{\}$
by *auto*
— Show that that individual is semi-decisive for x over z .
from *has-extend-witness'*[*OF has3A xyA*]
obtain z **where** *threeDist*: $hasw [x,y,z] A$ **by** *auto*
from *sd-exists-witness*[*OF threeDist jV2V3*] VIs $Vnotempty$
obtain P **where** *profileP*: $profile A Is P$
and $V1xyzP: x (P j) \prec y \wedge y (P j) \prec z$
and $V2xyzP: \forall i \in ?V2. z (P i) \prec x \wedge x (P i) \prec y$
and $V3xyzP: \forall i \in ?V3. y (P i) \prec z \wedge z (P i) \prec x$
by (*simp, blast*)
have $xPz: x (swf P) \prec z$
proof(*rule rpr-less-le-trans*[**where** $y=z$])
from *profileP* *swf* **show** *rpr* $A (swf P)$ **by** *auto*
next
— $V2$ is semi-decisive, and everyone else opposes their choice. Ergo they prevail.
show $x (swf P) \prec y$
proof —
from *profileP* $V3xyzP$
have $\forall i \in ?V3. y (P i) \prec x$ **by** (*blast dest: rpr-less-trans*)
with *profileP* $V1xyzP$ $V2xyzP$ Vsd
show *?thesis unfolding semidecisive-def* **by** *auto*
qed
next
— This result is unfortunately quite tortuous.
from *SWF-rpr*[*OF swf*] **show** $y (swf P) \preceq z$
proof(*rule rpr-less-not*[*OF - - notI*])
from *threeDist* **show** $hasw [z, y] A$ **by** *auto*
next
assume $zPy: z (swf P) \prec y$

```

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':  $\bigwedge i. i \in ?V2 \implies z (P' i) \prec y$ 
  and nV2yz':  $\bigwedge i. i \in Is - ?V2 \implies y (P' i) \prec z$ 
from iaa have  $\bigwedge a b. \llbracket a \in \{y, z\}; b \in \{y, z\} \rrbracket \implies (a (swf P) \preceq b) = (a (swf P') \preceq b)$ 
proof(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:  $\forall i \in ?V2. z (P i) \prec y$  by (blast dest: rpr-less-trans)
show  $(a (P' i) \preceq b) = (a (P i) \preceq b)$ 
proof(cases i ∈ ?V2)
  case True
  with VIs profileP profileP' ab V2yz' V2yzP threeDist
  show ?thesis unfolding strict-pref-def profile-def by auto
next
  case False
  from V1xyzP V3xyzP
  have  $\forall i \in Is - ?V2. y (P i) \prec z$  by auto
  with iIs False VIs jV profileP profileP' ab nV2yz' threeDist
  show ?thesis unfolding profile-def strict-pref-def by auto
qed
qed (simp-all add: profileP profileP')
with zPy show z (swf P') < y unfolding strict-pref-def by blast
qed
with VIs Vsd Vmin[where V'=?V2] V2card V2notempty threeDist show False
by auto
qed (simp add: profileP threeDist)
qed
have semidecisive swf A Is ?V1 x z
proof
  from jV VIs show {j} ⊆ Is by blast
next
  — Use iaa to show the SWF must allow the individual to prevail.
fix P'
assume profileP': profile A Is P'
  and V1yz':  $\bigwedge i. i \in ?V1 \implies x (P' i) \prec z$ 
  and nV1yz':  $\bigwedge i. i \in Is - ?V1 \implies z (P' i) \prec x$ 
from iaa have  $\bigwedge a b. \llbracket a \in \{x, z\}; b \in \{x, z\} \rrbracket \implies (a (swf P) \preceq b) = (a (swf P') \preceq b)$ 
proof(rule iiaE)
  from threeDist show xzA: {x,z} ⊆ A by simp
next
fix i assume iIs: i ∈ Is
fix a b assume ab: a ∈ {x, z} b ∈ {x, z}
show  $(a (P' i) \preceq b) = (a (P i) \preceq b)$ 

```

```

proof(cases i ∈ ?V1)
  case True
    with jV VIs profileP V1xyzP
    have ∀ i ∈ ?V1. x (P i) ≺ 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 ∀ i ∈ Is − ?V1. z (P i) ≺ x by auto
    with iIs False VIs jV profileP profileP' ab nV1yz' threeDist
    show ?thesis unfolding strict-pref-def profile-def by auto
  qed
qed (simp-all add: profileP profileP')
with xPz show x (swf P) ≺ z unfolding strict-pref-def by blast
qed
with jV VIs Vsd Vmin[where V'=?V1] V2card threeDist show False
  by auto
qed
with xyA Vsd show ?thesis by blast
qed

```

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.

and $\neg(\exists j. \text{dictator } \text{sdf } A \text{ } Is \ j)$
and $SDF \ \text{sdf } A \text{ } Is \ \text{universal-domain}$
proof –
let $?sdf = \lambda P. \{ (x, y) . x \in A \wedge y \in A$
 $\quad \wedge \neg((\forall i \in Is. y \ (P \ i) \preceq x)$
 $\quad \wedge (\exists i \in Is. y \ (P \ i) \prec x)) \}$
have $\text{weak-pareto } ?sdf \ A \ Is \ \text{universal-domain}$
by (*rule, unfold strict-pref-def, auto dest: profile-non-empty*)
moreover
have $ia \ ?sdf \ A \ Is \ \text{unfolding } \text{strict-pref-def} \ \text{by } \text{auto}$
moreover
have $\neg(\exists j. \text{dictator } ?sdf \ A \ Is \ j)$
proof
assume $\exists j. \text{dictator } ?sdf \ A \ Is \ j$
then obtain $j \ \text{where } jIs: j \in Is$
and $jD: \forall x \in A. \forall y \in A. \text{decisive } ?sdf \ A \ Is \ \{j\} \ x \ y$
unfolding $\text{dictator-def } \text{decisive-def} \ \text{by } \text{auto}$
from $jIs \ \text{has-witness-two}[OF \ \text{has2Is}] \ \text{obtain } i \ \text{where } ijIs: \text{hasw } [i, j] \ Is$
by auto
from $\text{has-witness-two}[OF \ \text{has2A}] \ \text{obtain } x \ y \ \text{where } xyA: \text{hasw } [x, y] \ A \ \text{by } \text{auto}$
from $xyA \ ijIs \ \text{obtain } P$
where $\text{profile}P: \text{profile } A \ Is \ P$
and $yPix: x \ (P \ i) \prec y$
and $yPjx: y \ (P \ j) \prec x$
by (*rule SDF-nodictator-witness*)
from $\text{profile}P \ jD \ jIs \ xyA \ yPjx \ \text{have } y \ (?sdf \ P) \prec x$
unfolding $\text{decisive-def} \ \text{by } \text{simp}$
moreover
from $ijIs \ xyA \ yPjx \ yPix \ \text{have } x \ (?sdf \ P) \preceq y$
unfolding $\text{strict-pref-def} \ \text{by } \text{auto}$
ultimately show False
unfolding $\text{strict-pref-def} \ \text{by } \text{blast}$
qed
moreover
have $SDF \ ?sdf \ A \ Is \ \text{universal-domain}$
proof
fix $P \ \text{assume } ud: \text{universal-domain } A \ Is \ P$
show $\text{choiceFn } A \ (?sdf \ P)$
proof(*rule r-c-qt-imp-cf[OF finiteA]*)
show $\text{complete } A \ (?sdf \ P) \ \text{and } \text{refl-on } A \ (?sdf \ P)$
unfolding $\text{strict-pref-def} \ \text{by } \text{auto}$
show $\text{quasi-trans } (?sdf \ P)$
proof
fix $x \ y \ z \ \text{assume } xy: x \ (?sdf \ P) \prec y \ \text{and } yz: y \ (?sdf \ P) \prec z$
from $xy \ yz \ \text{have } xyzA: x \in A \ y \in A \ z \in A$
unfolding $\text{strict-pref-def} \ \text{by } \text{auto}$
from $xy \ yz \ \text{have } AxRy: \forall i \in Is. x \ (P \ i) \preceq y$
and $ExPy: \exists i \in Is. x \ (P \ i) \prec y$
and $AyRz: \forall i \in Is. y \ (P \ i) \preceq z$
unfolding $\text{strict-pref-def} \ \text{by } \text{auto}$
from $AxRy \ AyRz \ ud \ \text{have } AxRz: \forall i \in Is. x \ (P \ i) \preceq z$

```

    by – (unfold universal-domain-def, blast dest: rpr-le-trans)
  from ExPy AyRz ud have ExPz:  $\exists i \in Is. x (P i) \prec z$ 
    by – (unfold universal-domain-def, blast dest: rpr-less-le-trans)
  from xyzA AxRz ExPz show  $x (?sdf P) \prec z$  unfolding strict-pref-def by auto
qed
qed
qed
ultimately show thesis ..
qed

```

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 \Rightarrow 'a set \Rightarrow 'i set \Rightarrow bool **where**

```

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

```

lemma *liberalE*:

```

[[ liberal scf A Is; i  $\in$  Is ]
 $\implies \exists x \in A. \exists y \in A. x \neq y$ 
    $\wedge$  decisive scf A Is {i} x y  $\wedge$  decisive scf A Is {i} y x
by (simp add: liberal-def)

```

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 \Rightarrow 'a set \Rightarrow 'i set \Rightarrow bool **where**

```

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

```

lemma *liberal-imp-minimally-liberal*:

assumes has2Is: has 2 Is

and L: liberal scf A Is

shows minimally-liberal scf A Is

proof –

```

from has-extend-witness[where xs=[], OF has2Is]
obtain i where i: i  $\in$  Is by auto
with has-extend-witness[where xs=[i], OF has2Is]
obtain j where j: j  $\in$  Is i  $\neq$  j by auto
from L i j show ?thesis

```

unfolding *minimally-liberal-def* **by** (*blast intro: liberalE*)
qed

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*

obtains *P*

where *profile A Is P*

and $x (P i) \prec y$

and $v (P j) \prec x$

and $\forall i \in Is. y (P i) \prec v$

proof –

let *?P =*

*λa. if a = i then { (x, u) | u. u ∈ A }
 ∪ { (y, u) | u. u ∈ A - {x} }
 ∪ (A - {x,y}) × (A - {x,y})
 else { (y, u) | u. u ∈ A }
 ∪ { (v, u) | u. u ∈ A - {y} }
 ∪ (A - {v,y}) × (A - {v,y})*

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** $x (?P i) \prec y v (?P j) \prec x \forall i \in Is. y (?P i) \prec v$

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*

obtains *P*

where *profile A Is P*

and $x (P i) \prec y$

and $u (P j) \prec v$

and $\forall i \in Is. v (P i) \prec x \wedge y (P i) \prec u$
proof –
let $?P =$
 $\lambda a. \text{if } a = i \text{ then } \{ (v, w) \mid w. w \in A \}$
 $\cup \{ (x, w) \mid w. w \in A - \{v\} \}$
 $\cup \{ (y, w) \mid w. w \in A - \{v, x\} \}$
 $\cup (A - \{v, x, y\}) \times (A - \{v, x, y\})$
else $\{ (y, w) \mid w. w \in A \}$
 $\cup \{ (u, w) \mid w. w \in A - \{y\} \}$
 $\cup \{ (v, w) \mid w. w \in A - \{u, y\} \}$
 $\cup (A - \{u, v, y\}) \times (A - \{u, v, y\})$
have profile $A \text{ Is } ?P$
proof
fix i **assume** $iis: i \in Is$
show $rpr A (?P i)$
proof
show $complete A (?P i)$ **by** $(simp, blast)$
from $fourA iis$ **show** $refl-on A (?P i)$ **by** $(simp, blast)$
from $fourA iis$ **show** $trans (?P i)$ **by** $(clarsimp simp add: trans-def)$
qed
next
from $twoIs$ **show** $Is \neq \{\}$ **by** $auto$
qed
moreover
from $fourA twoIs$ **have** $x (?P i) \prec y \wedge u (?P j) \prec v \wedge \forall i \in Is. v (?P i) \prec x \wedge y (?P i) \prec u$
by $(unfold strict-pref-def, auto)$
ultimately show thesis ..
qed

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

theorem *LiberalParadox:*

assumes $SDF: SDF \text{ sdf } A \text{ Is universal-domain}$
and $ml: minimally-liberal \text{ sdf } A \text{ Is}$
and $wp: weak-pareto \text{ sdf } A \text{ Is universal-domain}$
shows $False$

proof –

from ml **obtain** $i j x y u v$
where $i: i \in Is$ **and** $j: j \in Is$ **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 $dixy: decisive \text{ sdf } A \text{ Is } \{i\} x y$
and $diy: decisive \text{ sdf } A \text{ Is } \{i\} y x$
and $uv: u \neq v$
and $djuv: decisive \text{ sdf } A \text{ Is } \{j\} u v$
and $djvu: decisive \text{ sdf } A \text{ Is } \{j\} v u$
by $(unfold minimally-liberal-def, auto)$
from $i j ij$ **have** $twoIs: hasw [i,j] Is$ **by** $simp$
{
assume $xv: x = u$ **and** $yv: y = v$
from $xy x y$ **have** $twoA: hasw [x,y] A$ **by** $simp$
obtain P


```

    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 djvu xu yv have False
  by (unfold decisive-def strict-pref-def, blast)
}
moreover
{
  assume xv: x = u and yv: y ≠ v
  with xy uv 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 vPjx j djvu xu profileP have vPx: v (sdf P) < x
    by (unfold decisive-def strict-pref-def, auto)
  from xPiy 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 xPy yPv 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 A Is P x (P i) < y y (P j) < x
    using liberal-witness-two[OF twoA twoIs] by blast
  with i j dixy djvu xv yu have False
    by (unfold decisive-def strict-pref-def, blast)
}
moreover
{
  assume xv: x = v and yu: y ≠ u
  with xy uv u 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 uPjx j djvu xv profileP have uPx: u (sdf P) < x
    by (unfold decisive-def strict-pref-def, auto)
  from xPiy 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) ≠ {}
    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 ≠ u and xv: x ≠ v and yu: y = u
  with v x y xy uv xu have threeA: hasw [y,x,v] A by simp
  obtain P
    where profileP: profile A Is P
      and yPix: y (P i) ≺ x
      and vPjy: v (P j) ≺ y
      and AxPv: ∀ i ∈ Is. x (P i) ≺ v
    using liberal-witness-three[OF threeA twoIs] by blast
  from yPix i diyx profileP have yPx: y (sdf P) ≺ x
    by (unfold decisive-def strict-pref-def, auto)
  from vPjy j djvu yu profileP have vPy: v (sdf P) ≺ y
    by (unfold decisive-def strict-pref-def, auto)
  from AxPv weak-paretoD[OF wp - x v] profileP have xPv: x (sdf P) ≺ v
    by auto
  from threeA profileP SDF have choiceSet {x,y,v} (sdf P) ≠ {}
    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 ≠ u and xv: x ≠ v and yv: y = v
  with u x y xy uv xu have threeA: hasw [y,x,u] A by simp
  obtain P
    where profileP: profile A Is P
      and yPix: y (P i) ≺ x
      and uPjy: u (P j) ≺ y
      and AxPu: ∀ i ∈ Is. x (P i) ≺ u
    using liberal-witness-three[OF threeA twoIs] by blast
  from yPix i diyx profileP have yPx: y (sdf P) ≺ x
    by (unfold decisive-def strict-pref-def, auto)
  from uPjy j djvu yv profileP have uPy: u (sdf P) ≺ y
    by (unfold decisive-def strict-pref-def, auto)
  from AxPu weak-paretoD[OF wp - x u] profileP have xPu: x (sdf P) ≺ u
    by auto
  from threeA profileP SDF have choiceSet {x,y,u} (sdf P) ≠ {}
    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 ≠ u and xv: x ≠ v and yu: y ≠ u and yv: y ≠ v

```

```

with  $u\ v\ x\ y\ xy\ uv\ xu$  have  $fourA: hasw\ [x,y,u,v]\ A$  by simp
obtain  $P$ 
  where  $profileP: profile\ A\ Is\ P$ 
    and  $xPiy: x\ (P\ i) \prec y$ 
    and  $uPjv: u\ (P\ j) \prec v$ 
    and  $AvPxAyPu: \forall i \in Is. v\ (P\ i) \prec x \wedge y\ (P\ i) \prec u$ 
  using liberal-witness-four[OF fourA twoIs] by blast
from  $xPiy\ i\ dixy\ profileP$  have  $xPy: x\ (sdf\ P) \prec y$ 
  by (unfold decisive-def strict-pref-def, auto)
from  $uPjv\ j\ djvw\ profileP$  have  $uPv: u\ (sdf\ P) \prec v$ 
  by (unfold decisive-def strict-pref-def, auto)
from  $AvPxAyPu\ weak-paretoD[OF wp]\ profileP\ x\ y\ u\ v$ 
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* \Rightarrow $'a\ set \Rightarrow 'i\ set \Rightarrow bool$ **where**

anonymous scf $A\ Is \equiv$

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

$\longrightarrow (x\ (scf\ P) \preceq y) = (x\ (scf\ (P \circ f)) \preceq y))$

lemma *anonymousI[intro]*:

$(\bigwedge P\ f\ x\ y. \llbracket profile\ A\ Is\ P; bij\ betw\ f\ Is\ Is;$

$x \in A; y \in A \rrbracket \Longrightarrow (x\ (scf\ P) \preceq y) = (x\ (scf\ (P \circ f)) \preceq y))$

$\Longrightarrow anonymous\ scf\ A\ Is$

unfolding *anonymous-def* **by** *simp*

lemma anonymousD:

$\llbracket \text{anonymous scf } A \text{ Is}; \text{ profile } A \text{ Is } P; \text{ bij-betw } f \text{ Is Is}; x \in A; y \in A \rrbracket$
 $\implies (x \text{ (scf } P) \preceq y) = (x \text{ (scf } (P \circ f)) \preceq y)$
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

$\text{neutral scf } A \text{ Is} \equiv$
 $(\forall P P' x y z w. \text{ profile } A \text{ Is } P \wedge \text{ profile } A \text{ Is } P' \wedge x \in A \wedge y \in A \wedge z \in A \wedge w \in A$
 $\wedge (\forall i \in \text{Is}. x \text{ (} P \text{ } i) \preceq y \longleftrightarrow z \text{ (} P' \text{ } i) \preceq w) \wedge (\forall i \in \text{Is}. y \text{ (} P \text{ } i) \preceq x \longleftrightarrow w \text{ (} P' \text{ } i) \preceq z)$
 $\longrightarrow ((x \text{ (scf } P) \preceq y \longleftrightarrow z \text{ (scf } P') \preceq w) \wedge (y \text{ (scf } P) \preceq x \longleftrightarrow w \text{ (scf } P') \preceq z)))$

lemma neutralI[intro]:

$(\bigwedge P P' x y z w.$
 $\llbracket \text{profile } A \text{ Is } P; \text{ profile } A \text{ Is } P'; \{x,y,z,w\} \subseteq A;$
 $\bigwedge i. i \in \text{Is} \implies x \text{ (} P \text{ } i) \preceq y \longleftrightarrow z \text{ (} P' \text{ } i) \preceq w;$
 $\bigwedge i. i \in \text{Is} \implies y \text{ (} P \text{ } i) \preceq x \longleftrightarrow w \text{ (} P' \text{ } i) \preceq z \rrbracket$
 $\implies ((x \text{ (scf } P) \preceq y \longleftrightarrow z \text{ (scf } P') \preceq w) \wedge (y \text{ (scf } P) \preceq x \longleftrightarrow w \text{ (scf } P') \preceq z))$
 $\implies \text{neutral scf } A \text{ Is}$
unfolding neutral-def by simp

lemma neutralD:

$\llbracket \text{neutral scf } A \text{ Is};$
 $\text{ profile } A \text{ Is } P; \text{ profile } A \text{ Is } P'; \{x,y,z,w\} \subseteq A;$
 $\bigwedge i. i \in \text{Is} \implies x \text{ (} P \text{ } i) \preceq y \longleftrightarrow z \text{ (} P' \text{ } i) \preceq w;$
 $\bigwedge i. i \in \text{Is} \implies y \text{ (} P \text{ } i) \preceq x \longleftrightarrow w \text{ (} P' \text{ } i) \preceq z \rrbracket$
 $\implies (x \text{ (scf } P) \preceq y \longleftrightarrow z \text{ (scf } P') \preceq w) \wedge (y \text{ (scf } P) \preceq x \longleftrightarrow w \text{ (scf } P') \preceq z)$
unfolding neutral-def by simp

Neutrality implies independence of irrelevant alternatives.

lemma neutral-iaa: neutral scf A Is \implies 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

$\text{positively-responsive scf } A \text{ Is} \equiv$
 $(\forall P P' x y. \text{ profile } A \text{ Is } P \wedge \text{ profile } A \text{ Is } P' \wedge x \in A \wedge y \in A$
 $\wedge (\forall i \in \text{Is}. (x \text{ (} P \text{ } i) \prec y \longrightarrow x \text{ (} P' \text{ } i) \prec y) \wedge (x \text{ (} P \text{ } i) \approx y \longrightarrow x \text{ (} P' \text{ } i) \preceq y))$
 $\wedge (\exists k \in \text{Is}. (x \text{ (} P \text{ } k) \approx y \wedge x \text{ (} P' \text{ } k) \prec y) \vee (y \text{ (} P \text{ } k) \prec x \wedge x \text{ (} P' \text{ } k) \preceq y))$
 $\longrightarrow x \text{ (scf } P) \preceq y \longrightarrow x \text{ (scf } P') \prec y)$

lemma positively-responsiveI[intro]:

assumes $I: \bigwedge P P' x y.$
 $\llbracket \text{profile } A \text{ Is } P; \text{ profile } A \text{ Is } P'; x \in A; y \in A;$
 $\bigwedge i. \llbracket i \in \text{Is}; x \text{ (} P \text{ } i) \prec y \rrbracket \implies x \text{ (} P' \text{ } i) \prec y;$
 $\bigwedge i. \llbracket i \in \text{Is}; x \text{ (} P \text{ } i) \approx y \rrbracket \implies x \text{ (} P' \text{ } i) \preceq y;$
 $\exists k \in \text{Is}. (x \text{ (} P \text{ } k) \approx y \wedge x \text{ (} P' \text{ } k) \prec y) \vee (y \text{ (} P \text{ } k) \prec x \wedge x \text{ (} P' \text{ } k) \preceq y);$

$$x \text{ (scf } P) \preceq y \parallel$$

$$\implies x \text{ (scf } P') \prec y$$

shows *positively-responsive scf A Is*
unfolding *positively-responsive-def*
by (*blast intro: I*)

lemma *positively-responsiveD:*

$$\parallel \text{positively-responsive scf } A \text{ Is};$$

$$\text{profile } A \text{ Is } P; \text{ profile } A \text{ Is } P'; x \in A; y \in A;$$

$$\bigwedge i. \parallel i \in \text{Is}; x \text{ (} P \text{ } i) \prec y \parallel \implies x \text{ (} P' \text{ } i) \prec y;$$

$$\bigwedge i. \parallel i \in \text{Is}; x \text{ (} P \text{ } i) \approx y \parallel \implies x \text{ (} P' \text{ } i) \preceq y;$$

$$\exists k \in \text{Is}. (x \text{ (} P \text{ } k) \approx y \wedge x \text{ (} P' \text{ } k) \prec y) \vee (y \text{ (} P \text{ } k) \prec x \wedge x \text{ (} P' \text{ } k) \preceq y);$$

$$x \text{ (scf } P) \preceq y \parallel$$

$$\implies x \text{ (scf } P') \prec y$$

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* :: $'a \text{ set} \Rightarrow ('a, 'i) \text{ SCF where}$

$$\text{MMD Is } P \equiv \{ (x, y) . \text{card} \{ i \in \text{Is}. x \text{ (} P \text{ } i) \prec y \} \geq \text{card} \{ i \in \text{Is}. y \text{ (} P \text{ } i) \prec x \} \}$$

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

lemma *MMD-l2r:*

fixes $A :: 'a \text{ set}$
and $\text{Is} :: 'i \text{ set}$
assumes *finiteIs: finite Is*
shows *SCF (MMD Is) A Is universal-domain*
and *anonymous (MMD Is) A Is*
and *neutral (MMD Is) A Is*
and *positively-responsive (MMD Is) A Is*

proof –
show *SCF (MMD Is) A Is universal-domain*
proof
fix P **show** *complete A (MMD Is P)*
by (*rule completeI, unfold MMD-def, simp, arith*)
qed
show *anonymous (MMD Is) A Is*
proof
fix P
fix $x \ y :: 'a$
fix f **assume** *bijf: bij-betw f Is Is*

show $(x \text{ (MMD } Is \ P) \preceq y) = (x \text{ (MMD } Is \ (P \circ f)) \preceq y)$
using *card-compose-bij*[*OF bijf*, **where** $P = \lambda i. x \text{ (} P \ i) \prec y$]
card-compose-bij[*OF bijf*, **where** $P = \lambda i. y \text{ (} P \ i) \prec x$]
unfolding *MMD-def* **by** *simp*
qed
next
show *neutral (MMD Is) A Is*
proof
fix $P \ P'$
fix $x \ y \ z \ w$ **assume** $xyzwA: \{x, y, z, w\} \subseteq A$
assume $xyzw: \bigwedge i. i \in Is \implies (x \text{ (} P \ i) \preceq y) = (z \text{ (} P' \ i) \preceq w)$
and $yxwz: \bigwedge i. i \in Is \implies (y \text{ (} P \ i) \preceq x) = (w \text{ (} P' \ i) \preceq z)$
from $xyzwA \ xyzw \ yxwz$
have $\{ i \in Is. x \text{ (} P \ i) \prec y \} = \{ i \in Is. z \text{ (} P' \ i) \prec w \}$
and $\{ i \in Is. y \text{ (} P \ i) \prec x \} = \{ i \in Is. w \text{ (} P' \ i) \prec z \}$
unfolding *strict-pref-def* **by** *auto*
thus $(x \text{ (MMD } Is \ P) \preceq y) = (z \text{ (MMD } Is \ P') \preceq w) \wedge$
 $(y \text{ (MMD } Is \ P) \preceq x) = (w \text{ (MMD } Is \ P') \preceq z)$
unfolding *MMD-def* **by** *simp*
qed
next
show *positively-responsive (MMD Is) A Is*
proof
fix $P \ P'$ **assume** *profileP: profile A Is P*
fix $x \ y$ **assume** $xyA: x \in A \ y \in A$
assume $xPy: \bigwedge i. \llbracket i \in Is; x \text{ (} P \ i) \prec y \rrbracket \implies x \text{ (} P' \ i) \prec y$
and $xIy: \bigwedge i. \llbracket i \in Is; x \text{ (} P \ i) \approx y \rrbracket \implies x \text{ (} P' \ i) \preceq y$
and $k: \exists k \in Is. x \text{ (} P \ k) \approx y \wedge x \text{ (} P' \ k) \prec y \vee y \text{ (} P \ k) \prec x \wedge x \text{ (} P' \ k) \preceq y$
and $xRSCFy: x \text{ (MMD } Is \ P) \preceq y$
from k **obtain** k
where $kIs: k \in Is$
and $kcond: (x \text{ (} P \ k) \approx y \wedge x \text{ (} P' \ k) \prec y) \vee (y \text{ (} P \ k) \prec x \wedge x \text{ (} P' \ k) \preceq y)$
by *blast*
let $?xPy = \{ i \in Is. x \text{ (} P \ i) \prec y \}$
let $?xP'y = \{ i \in Is. x \text{ (} P' \ i) \prec y \}$
let $?yPx = \{ i \in Is. y \text{ (} P \ i) \prec x \}$
let $?yP'x = \{ i \in Is. y \text{ (} P' \ i) \prec x \}$
from *profileP xyA xPy xIy* **have** $yP'xyPx: ?yP'x \subseteq ?yPx$
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** $xPyxP'yC: \text{card } ?xPy \leq \text{card } ?xP'y$
by (*blast intro: card-mono finite-subset*)
show $x \text{ (MMD } Is \ P') \prec y$
proof
from $xRSCFy \ xPyxP'yC \ yP'xyPxC$ **show** $x \text{ (MMD } Is \ P') \preceq y$
unfolding *MMD-def* **by** *auto*
next

```

{
  assume  $xIky: x (P k) \approx y$  and  $xP'ky: x (P' k) \prec y$ 
  have  $\text{card } ?xPy < \text{card } ?xP'y$ 
  proof -
    from  $xIky$  have  $knP: k \notin ?xPy$ 
      unfolding indifferent-pref-def strict-pref-def by blast
    from  $kIs xP'ky$  have  $kP': k \in ?xP'y$  by simp
    from  $\text{finiteIs } xPy knP kP'$  show ?thesis
      by (blast intro: psubset-card-mono finite-subset)
  qed
  with  $xRSCFy yP'xyPx$  have  $\text{card } ?yP'x < \text{card } ?xP'y$ 
    unfolding MMD-def by auto
}
moreover
{
  assume  $yP'kx: y (P k) \prec x$  and  $xR'ky: x (P' k) \preceq y$ 
  have  $\text{card } ?yP'x < \text{card } ?yPx$ 
  proof -
    from  $kIs yP'kx$  have  $kP: k \in ?yPx$  by simp
    from  $kIs xR'ky$  have  $knP': k \notin ?yP'x$ 
      unfolding strict-pref-def by blast
    from  $yP'xyPx kP knP'$  have  $?yP'x \subset ?yPx$  by blast
    with  $\text{finiteIs}$  show ?thesis
      by (blast intro: psubset-card-mono finite-subset)
  qed
  with  $xRSCFy xPyxP'yC$  have  $\text{card } ?yP'x < \text{card } ?xP'y$ 
    unfolding MMD-def by auto
}
moreover note kcond
ultimately show  $\neg(y (MMD Is P') \preceq x)$ 
  unfolding MMD-def by auto
qed
qed
qed

```

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* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$ **where**
swapAlts a b u \equiv *if* $u = a$ *then* b *else if* $u = b$ *then* a *else* u

lemma *swapAlts-in-set-iff*: $\{a, b\} \subseteq A \implies \text{swapAlts } a b u \in A \longleftrightarrow u \in A$
 unfolding *swapAlts-def* by (*simp split: if-split*)

definition *swapAltsP* :: $('a, 'i)$ Profile $\Rightarrow 'a \Rightarrow 'a \Rightarrow ('a, 'i)$ Profile **where**
swapAltsP P a b \equiv $(\lambda i. \{ (u, v) . (\text{swapAlts } a b u, \text{swapAlts } a b v) \in P i \})$

lemma *swapAltsP-ab*: $a (P i) \preceq b \longleftrightarrow b (\text{swapAltsP } P a b i) \preceq a$ $b (P i) \preceq a \longleftrightarrow a (\text{swapAltsP } P a b i) \preceq b$

```

unfolding 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 iIs: 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 iIs abA show swapAltsP P a b i  $\subseteq$  A  $\times$  A
      unfolding swapAltsP-def by (blast dest: swapAlts-in-set-iff)
      from profileP iIs abA show  $\bigwedge x. x \in A \implies x$  (swapAltsP P a b i)  $\preceq$  x
      unfolding swapAltsP-def swapAlts-def by auto
    qed
  next
  from profileP iIs 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 iIs 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

```


begin

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 *iaa*, implied by neutrality, to restrict attention to alternatives x and y .

lemma *anonymous-card*:

assumes *profileP*: *profile* A Is P

and *profileP'*: *profile* A Is P'

and *xyA*: *hasw* $[x,y]$ A

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

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

shows $x (scf\ P) \preceq y \iff x (scf\ P') \preceq y$

proof –

let $?xPy = \{ i \in Is. x (P\ i) \prec y \}$

let $?xP'y = \{ i \in Is. x (P'\ i) \prec y \}$

let $?yPx = \{ i \in Is. y (P\ i) \prec x \}$

let $?yP'x = \{ i \in Is. y (P'\ i) \prec x \}$

have *disjPxy*: $(?xPy \cup ?yPx) - ?xPy = ?yPx$

unfolding *strict-pref-def* **by** *blast*

have *disjP'xy*: $(?xP'y \cup ?yP'x) - ?xP'y = ?yP'x$

unfolding *strict-pref-def* **by** *blast*

from *finiteIs* *xytally*

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 $?yPx$ $?yP'x$

by – (*drule* *card-eq-bij*, *auto*)

from *bijf* *bijg* *disjPxy* *disjP'xy*

obtain h

where *bijh*: *bij-betw* h $(?xPy \cup ?yPx)$ $(?xP'y \cup ?yP'x)$

and *hf*: $\bigwedge j. j \in ?xPy \implies h\ j = f\ j$

and *hg*: $\bigwedge j. j \in (?xPy \cup ?yPx) - ?xPy \implies h\ j = g\ j$

using *bij-combine*[**where** $f=f$ **and** $g=g$ **and** $A=?xPy$ **and** $B=?xPy \cup ?yPx$ **and** $C=?xP'y$ **and** $D=?xP'y \cup ?yP'x$]

by *auto*

from *bijh* *finiteIs*

obtain h' **where** *bijh'*: *bij-betw* h' Is Is

and *hh'*: $\bigwedge j. j \in (?xPy \cup ?yPx) \implies h'\ j = h\ j$

and *hrest*: $\bigwedge j. j \in Is - (?xPy \cup ?yPx) \implies h'\ j \in Is - (?xP'y \cup ?yP'x)$

by – (*drule* *bij-complete*, *auto*)

from *neutral-iaa*[*OF* *neutral*]

have $x (scf\ (P' \circ h')) \preceq y \iff x (scf\ P) \preceq y$

proof(*rule* *iaaE*)

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 *profileP* *iIs* **have** *completePi*: *complete* A $(P\ i)$ **by** (*auto* *dest*: *rprD*)

from *completePi* *xyA*

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

proof(*cases* *rule*: *complete-exh*)

```

    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  $profileP\ profileP'\ xyA\ iIs\ ab\ hh'\ hg\ bijg$  show  $?thesis$ 
      unfolding  $strict-pref-def\ bij-betw-def$  by ( $simp, blast$ )
next
    case  $xIy$  with  $profileP\ profileP'\ xyA\ iIs\ ab\ hrest[\mathbf{where}\ j=i]$  show  $?thesis$ 
      unfolding  $indifferent-pref-def\ strict-pref-def\ bij-betw-def$ 
      by ( $simp, blast\ dest: rpr-complete$ )
qed
qed ( $simp-all\ add: profileP\ profile-bij-profile[OF\ profileP'\ bijh']$ )
moreover
from  $anonymousD[OF\ anonymous\ profileP'\ bijh']\ xyA$ 
have  $x\ (scf\ P')\ \preceq\ y \iff x\ (scf\ (P' \circ h'))\ \preceq\ y$  by  $simp$ 
ultimately show  $?thesis$  by  $simp$ 
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  $xyA: hasw\ [x,y]\ A$ 
    and  $tallyP: card\ \{i \in Is.\ x\ (P\ i) \prec y\} = card\ \{i \in Is.\ y\ (P\ i) \prec x\}$ 
shows  $x\ (scf\ P) \approx y$ 
proof -
  — Neutrality insists the results for  $P$  are symmetrical to those for  $swapAltsP\ P$ .
from  $xyA$ 
have  $symPP': (x\ (scf\ P) \preceq y \iff y\ (scf\ (swapAltsP\ P\ x\ y)) \preceq x)$ 
     $\wedge\ (y\ (scf\ P) \preceq x \iff x\ (scf\ (swapAltsP\ P\ x\ y)) \preceq y)$ 
by - ( $rule\ neutralD[OF\ neutral\ profileP\ profile-swapAltsP[OF\ profileP]]$ ,
     $simp-all, (rule\ swapAltsP-ab)+$ )
  — Anonymity and neutrality insist the results for  $P$  are identical to those for  $swapAltsP\ P$ .
from  $xyA\ tallyP$  have  $card\ \{i \in Is.\ x\ (P\ i) \prec y\} = card\ \{i \in Is.\ x\ (swapAltsP\ P\ x\ y\ i) \prec y\}$ 
    and  $card\ \{i \in Is.\ y\ (P\ i) \prec x\} = card\ \{i \in Is.\ y\ (swapAltsP\ P\ x\ y\ i) \prec x\}$ 
unfolding  $swapAltsP-def\ swapAlts-def\ strict-pref-def$  by  $simp-all$ 
with  $profileP\ xyA$  have  $idPP': x\ (scf\ P) \preceq y \iff x\ (scf\ (swapAltsP\ P\ x\ y)) \preceq y$ 
    and  $y\ (scf\ P) \preceq x \iff y\ (scf\ (swapAltsP\ P\ x\ y)) \preceq x$ 
by - ( $rule\ anonymous-card[OF\ profileP\ profile-swapAltsP], clarsimp+$ )
from  $xyA\ SCF-completeD[OF\ SCF]\ profileP\ symPP'\ idPP'$  show  $x\ (scf\ P) \approx y$  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  $xyA: hasw\ [x,y]\ A$ 
    and  $tallyP: card\ \{i \in Is.\ x\ (P\ i) \prec y\} > card\ \{i \in Is.\ y\ (P\ i) \prec x\}$ 
obtains  $P'\ k$ 
  where  $profile\ A\ Is\ P'$ 
    and  $\bigwedge i. \llbracket i \in Is; x\ (P'\ i) \prec y \rrbracket \implies x\ (P\ i) \prec y$ 

```

and $\bigwedge i. \llbracket i \in Is; x (P' i) \approx y \rrbracket \implies x (P i) \preceq y$
and $k \in Is \wedge x (P' k) \approx y \wedge x (P k) \prec y$
and $\text{card} \{ i \in Is. x (P' i) \prec y \} = \text{card} \{ i \in Is. y (P' i) \prec x \}$

proof –

from *tallyP* **obtain** C

where *tallyP'*: $\text{card} (\{ i \in Is. x (P i) \prec y \} - C) = \text{card} \{ i \in Is. y (P i) \prec x \}$

and $C: C \neq \{ \}$ $C \subseteq Is$

and $CxPy: C \subseteq \{ i \in Is. x (P i) \prec y \}$

by – (*drule card-greater[OF finiteIs], auto*)

– Add (b, a) and close under transitivity.

let $?P' = \lambda i. \text{if } i \in C$

then $P i \cup \{ (y, x) \}$

$\cup \{ (y, u) \mid u. x (P i) \preceq u \}$

$\cup \{ (u, x) \mid u. u (P i) \preceq y \}$

$\cup \{ (v, u) \mid u v. x (P i) \preceq u \wedge v (P i) \preceq y \}$

else $P i$

have *profile A Is ?P'*

proof

fix i **assume** $iIs: i \in Is$

show *rpr A (?P' i)*

proof

from *profileP iIs* **show** *complete A (?P' i)*

unfolding *complete-def* **by** (*simp, blast dest: rpr-complete*)

from *profileP iIs xyA* **show** *refl-on A (?P' i)*

by – (*rule refl-onI, auto*)

show *trans (?P' i)*

proof(*cases i \in C*)

case *False* **with** *profileP iIs* **show** *?thesis*

by (*simp, blast dest: rpr-le-trans intro: transI*)

next

case *True* **with** *profileP iIs C CxPy xyA* **show** *?thesis*

unfolding *strict-pref-def*

by – (*rule transI, simp, blast dest: rpr-le-trans rpr-complete*)

qed

qed

next

from C **show** $Is \neq \{ \}$ **by** *blast*

qed

moreover

have $\bigwedge i. \llbracket i \in Is; x (?P' i) \prec y \rrbracket \implies x (P i) \prec y$

unfolding *strict-pref-def* **by** (*simp split: if-split-asm*)

moreover

from *profileP C xyA*

have $\bigwedge i. \llbracket i \in Is; x (?P' i) \approx y \rrbracket \implies x (P i) \preceq y$

unfolding *indifferent-pref-def* **by** (*simp split: if-split-asm*)

moreover

from $C CxPy$ **obtain** k **where** $kC: k \in C$ **and** $xPky: x (P k) \prec y$ **by** *blast*

hence $x (?P' k) \approx y$ **by** *auto*

with $C kC xPky$ **have** $k \in Is \wedge x (?P' k) \approx y \wedge x (P k) \prec y$ **by** *blast*

moreover

have $\text{card} \{ i \in Is. x (?P' i) \prec y \} = \text{card} \{ i \in Is. y (?P' i) \prec x \}$

proof –
have $\{ i \in Is. x \text{ } (\text{?}P' i) \prec y \} = \{ i \in Is. x \text{ } (\text{?}P' i) \prec y \} - C$
proof –
from C **have** $\bigwedge i. \llbracket i \in Is; x \text{ } (\text{?}P' i) \prec y \rrbracket \implies i \in Is - C$
unfolding *indifferent-pref-def strict-pref-def* **by** *auto*
thus *?thesis* **by** *blast*
qed
also have $\dots = \{ i \in Is. x \text{ } (P i) \prec y \} - C$ **by** *auto*
finally have $\text{card } \{ i \in Is. x \text{ } (\text{?}P' i) \prec y \} = \text{card } (\{ i \in Is. x \text{ } (P i) \prec y \} - C)$
by *simp*
with *tallyP'* **have** $\text{card } \{ i \in Is. x \text{ } (\text{?}P' i) \prec y \} = \text{card } \{ i \in Is. y \text{ } (P i) \prec x \}$
by *simp*
also have $\dots = \text{card } \{ i \in Is. y \text{ } (\text{?}P' i) \prec x \}$ (**is** $\text{card } ?lhs = \text{card } ?rhs$)
proof –
from *profileP xyA* **have** $\bigwedge i. \llbracket i \in Is; y \text{ } (\text{?}P' i) \prec x \rrbracket \implies y \text{ } (P i) \prec x$
unfolding *strict-pref-def* **by** (*simp split: if-split-asm, blast dest: rpr-complete*)
hence $?rhs \subseteq ?lhs$ **by** *blast*
moreover
from *profileP xyA* **have** $\bigwedge i. \llbracket i \in Is; y \text{ } (P i) \prec x \rrbracket \implies y \text{ } (\text{?}P' i) \prec x$
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*: *hasw [x,y] A*
and *tallyP*: $\text{card } \{ i \in Is. x \text{ } (P i) \prec y \} > \text{card } \{ i \in Is. y \text{ } (P i) \prec x \}$
shows $x \text{ } (\text{scf } P) \prec y$
proof –
from *assms* **obtain** $P' k$
where *profileP'*: *profile A Is P'*
and $F: \bigwedge i. \llbracket i \in Is; x \text{ } (P' i) \prec y \rrbracket \implies x \text{ } (P i) \prec y$
and $G: \bigwedge i. \llbracket i \in Is; x \text{ } (P' i) \approx y \rrbracket \implies x \text{ } (P i) \preceq y$
and *pivot*: $k \in Is \wedge x \text{ } (P' k) \approx y \wedge x \text{ } (P k) \prec y$
and *cardP'*: $\text{card } \{ i \in Is. x \text{ } (P' i) \prec y \} = \text{card } \{ i \in Is. y \text{ } (P' i) \prec x \}$
by – (*drule positively-responsive-prefer-witness, auto*)
from *profileP' xyA cardP'* **have** $x \text{ } (\text{scf } P') \approx y$
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-r2l*:
assumes *profileP*: *profile A Is P*
and *xyA*: *hasw [x,y] A*

```

shows  $x \text{ (scf } P) \preceq y \iff x \text{ (MMD } Is P) \preceq y$ 
proof(cases rule: linorder-cases)
  assume  $\text{card } \{ i \in Is. x \text{ (} P i) \prec y \} = \text{card } \{ i \in Is. y \text{ (} P i) \prec x \}$ 
  with profileP xyA show ?thesis
    using anonymous-neutral-indifference
    unfolding indifferent-pref-def MMD-def by simp
next
  assume  $\text{card } \{ i \in Is. x \text{ (} P i) \prec y \} > \text{card } \{ i \in Is. y \text{ (} P i) \prec x \}$ 
  with profileP xyA show ?thesis
    using positively-responsive-prefer
    unfolding strict-pref-def MMD-def by simp
next
  assume  $\text{card } \{ i \in Is. x \text{ (} P i) \prec y \} < \text{card } \{ i \in Is. y \text{ (} P i) \prec x \}$ 
  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 $('a, 'i) \text{SVProfile} = 'i \Rightarrow 'a$

definition *svprofile* :: $'a \text{ set} \Rightarrow 'i \text{ set} \Rightarrow ('a, 'i) \text{SVProfile} \Rightarrow \text{bool}$ **where**
svprofile $A \text{ Is } F \equiv \text{Is} \neq \{ \} \wedge F ' \text{Is} \subseteq A$

definition *plurality-rule* :: $'a \text{ set} \Rightarrow 'i \text{ set} \Rightarrow ('a, 'i) \text{SVProfile} \Rightarrow 'a \text{ set}$ **where**
plurality-rule $A \text{ Is } F$
 $\equiv \{ x \in A . \forall y \in A. \text{card } \{ i \in \text{Is} . F i = x \} \geq \text{card } \{ i \in \text{Is} . F i = y \} \}$

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* :: $'a \text{ set} \Rightarrow 'i \text{ set} \Rightarrow ('a, 'i) \text{Profile} \Rightarrow 'a \text{ set}$ **where**
MMD-plurality-rule $A \text{ Is } P \equiv \text{choiceSet } A \text{ (MMD } Is P)$

definition *single-vote-to-RPR* :: 'a set \Rightarrow 'a \Rightarrow 'a RPR where
single-vote-to-RPR A a \equiv { (a, x) | x. x \in A } \cup (A - {a}) \times (A - {a})

lemma *single-vote-to-RPR-iff*:

$\llbracket a \in A; x \in A; a \neq x \rrbracket \Longrightarrow (a \text{ (single-vote-to-RPR A b) } \prec x) \longleftrightarrow (b = a)$

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 \circ F)

proof -

{

fix x y

have $\llbracket x \in A; y \in A \rrbracket \Longrightarrow$

(card {i \in Is. F i = y} \leq card {i \in Is. F i = x}) =

(card {i \in Is. y (single-vote-to-RPR A (F i)) \prec x}

\leq 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

- [AK96] *Analyse & Kritik*, volume 18(1). 1996.
- [Arr63] K. J. Arrow. *Social Choice and Individual Values*. John Wiley and Sons, second edition, 1963.
- [GL06] R. E. Goodin and C. List. A conditional defense of plurality rule: Generalizing May's Theorem in a restricted informational environment. *American Journal of Political Science*, 50(4), 2006.
- [May52] K. O. May. A set of independent, necessary and sufficient conditions for simple majority decision. *Econometrica*, 20(4), 1952.
- [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. *Archive of Formal Proofs*, September 2008. <http://isa-afp.org/entries/ArrowImpossibilityGS.shtml>, Formal proof development.

- [Rou79] R. Routley. Repairing proofs of Arrow's General Impossibility Theorem and enlarging the scope of the theorem. *Notre Dame Journal of Formal Logic*, XX(4), 1979.
- [Sen70] Amartya Sen. *Collective Choice and Social Welfare*. Holden Day, 1970.
- [Tay05] A. D. Taylor. *Social Choice and the Mathematics of Manipulation*. Outlooks. Cambridge University Press, 2005.