The Incompatibility of Fishburn-Strategyproofness and Pareto-Efficiency

Felix Brandt, Manuel Eberl, Christian Saile, Christian Stricker

February 23, 2021

Abstract

This formalisation contains the proof that there is no anonymous Social Choice Function for at least three agents and alternatives that satisfies both Pareto-Efficiency and Fishburn-Strategyproofness. It was derived from a proof of Brandt *et al.* [1], which relies on an unverified translation of a fixed finite instance of the original problem to SAT. This Isabelle proof contains a machine-checked version of both the statement for exactly three agents and alternatives and the lifting to the general case.

Contents

1 Social Choice Functions

theory *Social-Choice-Functions*
imports
Randomised-Social-Choice.Preference-Profile-Cmd
begin

1.1 Weighted majority graphs

definition *supporters* :: ($'agent$, $'alt$) *pref-profile* $\Rightarrow$ $'alt \Rightarrow 'alt \Rightarrow 'agent$ *set* **where**
supporters-auxdef: *supporters R x y* = $\{i.\ x \succeq[R\ i]\ y\}$

definition *weighted-majority* :: ($'agent$, $'alt$) *pref-profile* $\Rightarrow$ $'alt \Rightarrow 'alt \Rightarrow int$ **where**
weighted-majority R x y = *int* (*card* (*supporters R x y*)) − *int* (*card* (*supporters R y x*))

lemma *weighted-majority-refl* [*simp*]: *weighted-majority R x x* = *0*
⟨*proof*⟩

lemma *weighted-majority-swap*: *weighted-majority R x y* = −*weighted-majority R y x*
⟨*proof*⟩

lemma *eval-set-filter*:
Set.filter P {} = {}
$P\ x \Longrightarrow$ *Set.filter P* (*insert x A*) = *insert x* (*Set.filter P A*)
$\neg P\ x \Longrightarrow$ *Set.filter P* (*insert x A*) = *Set.filter P A*
⟨*proof*⟩

context *election*
begin

lemma *supporters-def*:
assumes *is-pref-profile R*
shows *supporters R x y* = $\{i \in agents.\ x \succeq[R\ i]\ y\}$
⟨*proof*⟩

lemma *supporters-nonagent1*:
assumes *is-pref-profile R* $x \notin alts$
shows *supporters R x y* = {}
⟨*proof*⟩

lemma *supporters-nonagent2*:
assumes *is-pref-profile R* $y \notin alts$
shows *supporters R x y* = {}
⟨*proof*⟩

lemma *weighted-majority-nonagent1*:
assumes *is-pref-profile R* $x \notin alts$
shows *weighted-majority R x y* = *0*

⟨*proof*⟩

lemma *weighted-majority-nonagent2*:
assumes *is-pref-profile R* $y \notin$ *alts*
shows *weighted-majority R x y* = *0*
⟨*proof*⟩

lemma *weighted-majority-eq-iff*:
assumes *is-pref-profile R1 is-pref-profile R2*
shows *weighted-majority R1* = *weighted-majority R2* $\longleftrightarrow$
($\forall$ *x*$\in$*alts*. $\forall$ *y*$\in$*alts*. *weighted-majority R1 x y* = *weighted-majority R2 x y*)
⟨*proof*⟩

end

1.2 Definition of Social Choice Functions

locale *social-choice-function* = *election agents alts*
for *agents* :: *'agent set* **and** *alts* :: *'alt set* +
fixes *scf* :: (*'agent*, *'alt*) *pref-profile* $\Rightarrow$ *'alt set*
assumes *scf-nonempty*: *is-pref-profile R* $\Longrightarrow$ *scf R* $\neq$ {}
assumes *scf-alts*: *is-pref-profile R* $\Longrightarrow$ *scf R* $\subseteq$ *alts*

1.3 Anonymity

An SCF is anonymous if permuting the agents in the input does not change the result.

locale *anonymous-scf* = *social-choice-function agents alts scf*
for *agents* :: *'agent set* **and** *alts* :: *'alt set* **and** *scf* +
assumes *anonymous*: π *permutes agents* $\Longrightarrow$ *is-pref-profile R* $\Longrightarrow$ *scf* ($R \circ \pi$) = *scf R*
begin

lemma *anonymous'*:
assumes *anonymous-profile R1* = *anonymous-profile R2*
assumes *is-pref-profile R1 is-pref-profile R2*
shows *scf R1* = *scf R2*
⟨*proof*⟩

lemma *anonymity-prefs-from-table*:
assumes *prefs-from-table-wf agents alts xs prefs-from-table-wf agents alts ys*
assumes *mset* (*map snd xs*) = *mset* (*map snd ys*)
shows *scf* (*prefs-from-table xs*) = *scf* (*prefs-from-table ys*)
⟨*proof*⟩

context
begin
qualified lemma *anonymity-prefs-from-table-aux*:

assumes *R1* = *prefs-from-table xs prefs-from-table-wf agents alts xs*
assumes *R2* = *prefs-from-table ys prefs-from-table-wf agents alts ys*
assumes *mset* (*map snd xs*) = *mset* (*map snd ys*)
shows *scf R1* = *scf R2* ⟨*proof*⟩
end

end

1.4 Neutrality

An SCF is neutral if permuting the alternatives in the input does not change the result, modulo the equivalent permutation in the output lottery.

locale *neutral-scf* = *social-choice-function agents alts scf*
for *agents* :: *'agent set* **and** *alts* :: *'alt set* **and** *scf* +
assumes *neutral*: σ *permutes alts* ⟹ *is-pref-profile R* ⟹
scf (*permute-profile* σ *R*) = σ ‘ *scf R*
begin

Alternative formulation of neutrality that shows that our definition is equivalent to that in the paper.

lemma *neutral'*:
assumes σ *permutes alts*
assumes *is-pref-profile R*
assumes $a \in$ *alts*
shows σ $a \in$ *scf* (*permute-profile* σ *R*) ⟷ $a \in$ *scf R*
⟨*proof*⟩

end

locale *an-scf* =
anonymous-scf agents alts scf + *neutral-scf agents alts scf*
for *agents* :: *'agent set* **and** *alts* :: *'alt set* **and** *scf*
begin

lemma *scf-anonymous-neutral*:
assumes *perm*: σ *permutes alts* **and** *wf*: *is-pref-profile R1 is-pref-profile R2*
assumes *eq*: *anonymous-profile R1* =
image-mset (*map* ($\lambda A.$ σ ‘ *A*)) (*anonymous-profile R2*)
shows *scf R1* = σ ‘ *scf R2*
⟨*proof*⟩

lemma *scf-anonymous-neutral'*:
assumes *perm*: σ *permutes alts* **and** *wf*: *is-pref-profile R1 is-pref-profile R2*
assumes *eq*: *anonymous-profile R1* =
image-mset (*map* ($\lambda A.$ σ ‘ *A*)) (*anonymous-profile R2*)
shows σ $x \in$ *scf R1* ⟷ $x \in$ *scf R2*

⟨*proof*⟩

lemma *scf-automorphism*:
  assumes *perm*: σ *permutes alts* **and** *wf*: *is-pref-profile R*
  assumes *eq*: *image-mset* (*map* (λA. σ ‘ *A*)) (*anonymous-profile R*) = *anonymous-profile R*
  shows σ ‘ *scf R* = *scf R*
  ⟨*proof*⟩

end

lemma *an-scf-automorphism-aux*:
  assumes *wf*: *prefs-from-table-wf agents alts yss R* $\equiv$ *prefs-from-table yss*
  assumes *an*: *an-scf agents alts scf*
  assumes *eq*: *mset* (*map* ((*map* (λA. *permutation-of-list xs* ‘ *A*)) $\circ$ *snd*) *yss*) = *mset* (*map snd yss*)
  assumes *perm*: *set* (*map fst xs*) $\subseteq$ *alts set* (*map snd xs*) = *set* (*map fst xs*)
    distinct (*map fst xs*)
  and *x*: *x* $\in$ *alts y* = *permutation-of-list xs x*
  shows *x* $\in$ *scf R* $\longleftrightarrow$ *y* $\in$ *scf R*
⟨*proof*⟩

1.5 Weighted-majoritarian SCFs

locale *pairwise-scf* = *social-choice-function agents alts scf*
  for *agents* :: *'agent set* **and** *alts* :: *'alt set* **and** *scf* +
  assumes *pairwise*:
  is-pref-profile R1 $\Longrightarrow$ *is-pref-profile R2* $\Longrightarrow$ *weighted-majority R1* = *weighted-majority R2* $\Longrightarrow$
    scf R1 = *scf R2*

1.6 Pareto efficiency

locale *pareto-efficient-scf* = *social-choice-function agents alts scf*
  for *agents* :: *'agent set* **and** *alts* :: *'alt set* **and** *scf* +
  assumes *pareto-efficient*:
    is-pref-profile R $\Longrightarrow$ *scf R* $\cap$ *pareto-losers R* = {}
begin

lemma *pareto-efficient'*:
  assumes *is-pref-profile R y* $\succ$[*Pareto*(*R*)] *x*
  shows *x* $\notin$ *scf R*
  ⟨*proof*⟩

lemma *pareto-efficient''*:
  assumes *is-pref-profile R i* $\in$ *agents* $\forall$ *i*$\in$*agents*. *y* $\succeq$[*R i*] *x* $\neg$*y* $\preceq$[*R i*] *x*
  shows *x* $\notin$ *scf R*
⟨*proof*⟩

end

1.7 Set extensions

type-synonym *'alt set-extension* = *'alt relation* ⇒ *'alt set relation*

definition *Kelly* :: *'alt set-extension* **where**
$A \succeq[Kelly(R)]\ B \longleftrightarrow (\forall a{\in}A.\ \forall b{\in}B.\ a \succeq[R]\ b)$

lemma *Kelly-strict-iff*: $A \succ[Kelly(R)]\ B \longleftrightarrow ((\forall a{\in}A.\ \forall b{\in}B.\ R\ b\ a) \wedge \neg (\forall a{\in}B.\ \forall b{\in}A.\ R\ b\ a))$
⟨*proof*⟩

lemmas *Kelly-eval* = *Kelly-def Kelly-strict-iff*

definition *Fishb* :: *'alt set-extension* **where**
$A \succeq[Fishb(R)]\ B \longleftrightarrow (\forall a{\in}A.\ \forall b{\in}B{-}A.\ a \succeq[R]\ b) \wedge (\forall a{\in}A{-}B.\ \forall b{\in}B.\ a \succeq[R]\ b)$

lemma *Fishb-strict-iff*:
$A \succ[Fishb(R)]\ B \longleftrightarrow$
$((\forall a{\in}A.\ \forall b{\in}B - A.\ R\ b\ a) \wedge (\forall a{\in}A - B.\ \forall b{\in}B.\ R\ b\ a)) \wedge$
$\neg ((\forall a{\in}B.\ \forall b{\in}A - B.\ R\ b\ a) \wedge (\forall a{\in}B - A.\ \forall b{\in}A.\ R\ b\ a))$
⟨*proof*⟩

lemmas *Fishb-eval* = *Fishb-def Fishb-strict-iff*

1.8 Strategyproofness

locale *strategyproof-scf* =
social-choice-function agents alts scf
for *agents* :: *'agent set* **and** *alts* :: *'alt set* **and** *scf* +
fixes *set-ext* :: *'alt set-extension*
assumes *strategyproof*:
is-pref-profile R ⟹ *total-preorder-on alts Ri'* ⟹ $i \in$ *agents* ⟹
¬ *scf* (*R*(*i* := *Ri'*)) ≻[*set-ext*(*R i*)] *scf R*

locale *strategyproof-anonymous-scf* =
anonymous-scf agents alts scf + *strategyproof-scf agents alts scf set-ext*
for *agents* :: *'agent set* **and** *alts* :: *'alt set* **and** *scf* **and** *set-ext*
begin

lemma *strategyproof'*:
assumes *is-pref-profile R1 is-pref-profile R2 i* ∈ *agents j* ∈ *agents*
assumes *anonymous-profile R2* = *anonymous-profile R1* −
{#*weak-ranking* (*R1 i*)#} + {#*weak-ranking* (*R2 j*)#}
shows ¬*scf R2* ≻[*set-ext* (*R1 i*)] *scf R1*
⟨*proof*⟩

end

context *preorder-on*

begin

lemma *strict-not-outside*:
  assumes $x \prec[le]\ y$
  shows $x \in carrier\ y \in carrier$
  ⟨*proof*⟩

end

1.9 Lifting preferences

Preference profiles can be lifted to a setting with more agents and alternatives by padding them with dummy agents and alternatives in such a way that every original agent prefers and original alternative over any dummy alternative and is indifferent between the dummy alternatives. Moreover, every dummy agent is completely indifferent.

definition *lift-prefs* ::
  $'alt\ set \Rightarrow\ 'alt\ set \Rightarrow\ 'alt\ relation \Rightarrow\ 'alt\ relation$ **where**
  $lift\text{-}prefs\ alts\ alts'\ R = (\lambda x\ y.$
    $x \in alts' \wedge y \in alts' \wedge (x = y \vee x \notin alts \vee (y \in alts \wedge R\ x\ y)))$

lemma *lift-prefs-wf*:
  assumes $total\text{-}preorder\text{-}on\ alts\ R\ alts \subseteq alts'$
  shows $total\text{-}preorder\text{-}on\ alts'\ (lift\text{-}prefs\ alts\ alts'\ R)$
⟨*proof*⟩

definition *lift-pref-profile* ::
  $'agent\ set \Rightarrow\ 'alt\ set \Rightarrow\ 'agent\ set \Rightarrow\ 'alt\ set \Rightarrow$
    $('agent,\ 'alt)\ pref\text{-}profile \Rightarrow ('agent,\ 'alt)\ pref\text{-}profile$ **where**
  $lift\text{-}pref\text{-}profile\ agents\ alts\ agents'\ alts'\ R = (\lambda i\ x\ y.$
    $x \in alts' \wedge y \in alts' \wedge i \in agents' \wedge$
    $(x = y \vee x \notin alts \vee i \notin agents \vee (y \in alts \wedge R\ i\ x\ y)))$

lemma *lift-pref-profile-conv-vector*:
  assumes $i \in agents\ i \in agents'$
  shows $lift\text{-}pref\text{-}profile\ agents\ alts\ agents'\ alts'\ R\ i = lift\text{-}prefs\ alts\ alts'\ (R\ i)$
  ⟨*proof*⟩

lemma *lift-pref-profile-wf*:
  assumes $pref\text{-}profile\text{-}wf\ agents\ alts\ R$
  assumes $agents \subseteq agents'\ alts \subseteq alts'\ finite\ alts'$
  defines $R' \equiv lift\text{-}pref\text{-}profile\ agents\ alts\ agents'\ alts'\ R$
  shows $pref\text{-}profile\text{-}wf\ agents'\ alts'\ R'$
⟨*proof*⟩

lemma *lift-pref-profile-permute-agents*:
  assumes $\pi\ permutes\ agents\ agents \subseteq agents'$
  shows $lift\text{-}pref\text{-}profile\ agents\ alts\ agents'\ alts'\ (R \circ \pi) =$

$\quad$ *lift-pref-profile agents alts agents′ alts′ R* $\circ$ π
⟨*proof*⟩

lemma *lift-pref-profile-permute-alts*:
assumes σ *permutes alts alts* $\subseteq$ *alts′*
shows *lift-pref-profile agents alts agents′ alts′* (*permute-profile* σ *R*) =
permute-profile σ (*lift-pref-profile agents alts agents′ alts′ R*)
⟨*proof*⟩

context
fixes *agents alts R agents′ alts′ R′*
assumes *R-wf*: *pref-profile-wf agents alts R*
assumes *election*: *agents* $\subseteq$ *agents′ alts* $\subseteq$ *alts′ alts* $\neq$ {} *agents* $\neq$ {} *finite alts′*
defines *R′* $\equiv$ *lift-pref-profile agents alts agents′ alts′ R*
begin

interpretation *R*: *pref-profile-wf agents alts R* ⟨*proof*⟩
interpretation *R′*: *pref-profile-wf agents′ alts′ R′*
⟨*proof*⟩

lemma *lift-pref-profile-strict-iff*:
x $\prec$[*lift-pref-profile agents alts agents′ alts′ R i*] *y* $\longleftrightarrow$
i $\in$ *agents* $\wedge$ ((*y* $\in$ *alts* $\wedge$ *x* $\in$ *alts′* $-$ *alts*) $\vee$ *x* $\prec$[*R i*] *y*)
⟨*proof*⟩

lemma *preferred-alts-lift-pref-profile*:
assumes *i*: *i* $\in$ *agents′* **and** *x*: *x* $\in$ *alts′*
shows *preferred-alts* (*R′ i*) *x* =
(*if i* $\in$ *agents* $\wedge$ *x* $\in$ *alts then preferred-alts* (*R i*) *x else alts′*)
⟨*proof*⟩

lemma *lift-pref-profile-Pareto-iff*:
x $\preceq$[*Pareto*(*R′*)] *y* $\longleftrightarrow$ *x* $\in$ *alts′* $\wedge$ *y* $\in$ *alts′* $\wedge$ (*x* $\notin$ *alts* $\vee$ *x* $\preceq$[*Pareto*(*R*)] *y*)
⟨*proof*⟩

lemma *lift-pref-profile-Pareto-strict-iff*:
x $\prec$[*Pareto*(*R′*)] *y* $\longleftrightarrow$ *x* $\in$ *alts′* $\wedge$ *y* $\in$ *alts′* $\wedge$ (*x* $\notin$ *alts* $\wedge$ *y* $\in$ *alts* $\vee$ *x* $\prec$[*Pareto*(*R*)]
y)
⟨*proof*⟩

lemma *pareto-losers-lift-pref-profile*:
shows *pareto-losers R′* = *pareto-losers R* $\cup$ (*alts′* $-$ *alts*)
⟨*proof*⟩

end

1.10 Lowering SCFs

Using the preference lifting, we can now *lower* an SCF to a setting with fewer agents and alternatives under mild conditions to the original SCF. This preserves many nice properties, such as anonymity, neutrality, and strategyproofness.

locale *scf-lowering* =
  pareto-efficient-scf agents alts scf
  for *agents* :: *'agent set* **and** *alts* :: *'alt set* **and** *scf* +
  fixes *agents' alts'*
  assumes *agents'-subset*: *agents'* ⊆ *agents* **and** *alts'-subset*: *alts'* ⊆ *alts*
    and *agents'-nonempty* [*simp*]: *agents'* ≠ {} **and** *alts'-nonempty* [*simp*]: *alts'* ≠ {}
begin

lemma *finite-agents'* [*simp*]: *finite agents'*
  ⟨*proof*⟩

lemma *finite-alts'* [*simp*]: *finite alts'*
  ⟨*proof*⟩

abbreviation *lift* :: (*'agent*, *'alt*) *pref-profile* ⇒ (*'agent*, *'alt*) *pref-profile* **where**
  lift ≡ *lift-pref-profile agents' alts' agents alts*

definition *lowered* :: (*'agent*, *'alt*) *pref-profile* ⇒ *'alt set* **where**
  lowered = *scf* ∘ *lift*

lemma *lift-wf* [*simp*, *intro*]:
  pref-profile-wf agents' alts' R ⟹ *is-pref-profile* (*lift R*)
  ⟨*proof*⟩

sublocale *lowered*: *election agents' alts'*
  ⟨*proof*⟩

lemma *preferred-alts-lift*:
  lowered.is-pref-profile R ⟹ *i* ∈ *agents* ⟹ *x* ∈ *alts* ⟹
    preferred-alts (*lift R i*) *x* =
      (*if i* ∈ *agents'* ∧ *x* ∈ *alts' then preferred-alts* (*R i*) *x else alts*)
  ⟨*proof*⟩

lemma *pareto-losers-lift*:
  lowered.is-pref-profile R ⟹ *pareto-losers* (*lift R*) = *pareto-losers R* ∪ (*alts* − *alts'*)
  ⟨*proof*⟩

sublocale *lowered*: *social-choice-function agents' alts' lowered*
⟨*proof*⟩

sublocale *lowered*: *pareto-efficient-scf agents′ alts′ lowered*
⟨*proof*⟩

end

locale *scf-lowering-anonymous* =
anonymous-scf agents alts scf +
scf-lowering agents alts scf agents′ alts′
for *agents* :: *′agent set* **and** *alts* :: *′alt set* **and** *scf agents′ alts′*
begin

sublocale *lowered*: *anonymous-scf agents′ alts′ lowered*
⟨*proof*⟩

end

locale *scf-lowering-neutral* =
neutral-scf agents alts scf +
scf-lowering agents alts scf agents′ alts′
for *agents* :: *′agent set* **and** *alts* :: *′alt set* **and** *scf agents′ alts′*
begin

sublocale *lowered*: *neutral-scf agents′ alts′ lowered*
⟨*proof*⟩

end

The following is a technical condition that we need from a set extensions in order for strategyproofness to survive the lowering. The condition could probably be weakened a bit, but it is good enough for our purposes the way it is.

locale *liftable-set-extension* =
fixes *alts′ alts* :: *′alt set* **and** *set-ext* :: *′alt relation* ⇒ *′alt set relation*
assumes *set-ext-strong-lift*:
total-preorder-on alts′ R ⟹ *A* ≠ {} ⟹ *B* ≠ {} ⟹ *A* ⊆ *alts′* ⟹ *B* ⊆ *alts′* ⟹
A ≺[*set-ext R*] *B* ⟹ *A* ≺[*set-ext* (*lift-prefs alts′ alts R*)] *B*

lemma *liftable-set-extensionI-weak*:
assumes ⋀*R A B*. *total-preorder-on alts′ R* ⟹ *A* ≠ {} ⟹ *B* ≠ {} ⟹
A ⊆ *alts′* ⟹ *B* ⊆ *alts′* ⟹
A ≼[*set-ext R*] *B* ⟷ *A* ≼[*set-ext* (*lift-prefs alts′ alts R*)] *B*
shows *liftable-set-extension alts′ alts set-ext*
⟨*proof*⟩

lemma *Kelly-liftable*:
assumes *alts′* ⊆ *alts*

shows *liftable-set-extension alts′ alts Kelly*
⟨*proof*⟩

lemma *Fishburn-liftable*:
assumes $alts' \subseteq alts$
shows *liftable-set-extension alts′ alts Fishb*
⟨*proof*⟩

locale *scf-lowering-strategyproof* =
strategyproof-scf agents alts scf set-ext +
liftable-set-extension alts′ alts set-ext +
scf-lowering agents alts scf agents′ alts′
for *agents* :: *′agent set* **and** *alts* :: *′alt set* **and** *scf agents′ alts′ set-ext*
begin

sublocale *lowered*: *strategyproof-scf agents′ alts′ lowered*
⟨*proof*⟩

end

end

2 Main impossibility result

theory *Fishburn-Impossibility*
imports
Social-Choice-Functions
begin

2.1 Setting of the base case

Suppose we have an anonymous, Fishburn-strategyproof, and Pareto-efficient SCF for three agents $A1$ to $A3$ and three alternatives a, b, and c. We will derive a contradiction from this.

locale *fb-impossibility-3-3* =
strategyproof-anonymous-scf agents alts scf Fishb +
pareto-efficient-scf agents alts scf
for *agents* :: *′agent set* **and** *alts* :: *′alt set* **and** *scf A1 A2 A3 a b c* +
assumes *agents-eq*: $agents = \{A1, A2, A3\}$
assumes *alts-eq*: $alts = \{a, b, c\}$
assumes *distinct-agents*: *distinct* $[A1, A2, A3]$
assumes *distinct-alts*: *distinct* $[a, b, c]$
begin

We first give some simple rules that will allow us to break down the strategyproofness and support conditions more easily later.

lemma *agents-neq* [*simp*]: $A1 \neq A2$ $A2 \neq A1$ $A1 \neq A3$ $A3 \neq A1$ $A2 \neq A3$ $A3 \neq A2$

⟨*proof*⟩

lemma *alts-neq* [*simp*]: $a \neq b$ $a \neq c$ $b \neq c$ $b \neq a$ $c \neq a$ $c \neq b$
⟨*proof*⟩

lemma *agent-in-agents* [*simp*]: $A1 \in agents$ $A2 \in agents$ $A3 \in agents$
⟨*proof*⟩

lemma *alt-in-alts* [*simp*]: $a \in alts$ $b \in alts$ $c \in alts$
⟨*proof*⟩

lemma *Bex-alts*: $(\exists x{\in}alts.\ P\ x) \longleftrightarrow P\ a \lor P\ b \lor P\ c$
⟨*proof*⟩

lemma *eval-pareto-loser-aux*:
assumes *is-pref-profile R*
shows $x \in pareto\text{-}losers\ R \longleftrightarrow (\exists y{\in}\{a,b,c\}.\ x \prec[Pareto(R)]\ y)$
⟨*proof*⟩

lemma *eval-Pareto*:
assumes *is-pref-profile R*
shows $x \prec[Pareto(R)]\ y \longleftrightarrow (\forall i{\in}\{A1,A2,A3\}.\ x \preceq[R\ i]\ y) \land (\exists i{\in}\{A1,A2,A3\}.\ \neg x \succeq[R\ i]\ y)$
⟨*proof*⟩

lemmas *eval-pareto* = *eval-pareto-loser-aux eval-Pareto*
lemma *pareto-efficiency*: *is-pref-profile R* $\Longrightarrow x \in pareto\text{-}losers\ R \Longrightarrow x \notin scf\ R$
⟨*proof*⟩

lemma *Ball-scf*:
assumes *is-pref-profile R*
shows $(\forall x{\in}scf\ R.\ P\ x) \longleftrightarrow (a \notin scf\ R \lor P\ a) \land (b \notin scf\ R \lor P\ b) \land (c \notin scf\ R \lor P\ c)$
⟨*proof*⟩

lemma *Ball-scf-diff*:
assumes *is-pref-profile R1 is-pref-profile R2*
shows $(\forall x{\in}scf\ R1 - scf\ R2.\ P\ x) \longleftrightarrow$
$(a \in scf\ R2 \lor a \notin scf\ R1 \lor P\ a) \land (b \in scf\ R2 \lor b \notin scf\ R1 \lor P\ b) \land$
$(c \in scf\ R2 \lor c \notin scf\ R1 \lor P\ c)$
⟨*proof*⟩

lemma *scf-nonempty′*:
assumes *is-pref-profile R*
shows $\exists x{\in}alts.\ x \in scf\ R$
⟨*proof*⟩

2.2 Definition of Preference Profiles and Fact Gathering

We now define the 21 preference profile that will lead to the impossibility result.

preference-profile
agents: *agents*
alts: *alts*
where *R1* = *A1*: [*a*, *c*], *b* *A2*: [*a*, *c*], *b* *A3*: *b*, *c*, *a*
and *R2* = *A1*: *c*, [*a*, *b*] *A2*: *b*, *c*, *a* *A3*: *c*, *b*, *a*
and *R3* = *A1*: [*a*, *c*], *b* *A2*: *b*, *c*, *a* *A3*: *c*, *b*, *a*
and *R4* = *A1*: [*a*, *c*], *b* *A2*: *a*, *b*, *c* *A3*: *b*, *c*, *a*
and *R5* = *A1*: *c*, [*a*, *b*] *A2*: *a*, *b*, *c* *A3*: *b*, *c*, *a*
and *R6* = *A1*: *b*, [*a*, *c*] *A2*: *c*, [*a*, *b*] *A3*: *b*, *c*, *a*
and *R7* = *A1*: [*a*, *c*], *b* *A2*: *b*, [*a*, *c*] *A3*: *b*, *c*, *a*
and *R8* = *A1*: [*b*, *c*], *a* *A2*: *a*, [*b*, *c*] *A3*: *a*, *c*, *b*
and *R9* = *A1*: [*b*, *c*], *a* *A2*: *b*, [*a*, *c*] *A3*: *a*, *b*, *c*
and *R10* = *A1*: *c*, [*a*, *b*] *A2*: *a*, *b*, *c* *A3*: *c*, *b*, *a*
and *R11* = *A1*: [*a*, *c*], *b* *A2*: *a*, *b*, *c* *A3*: *c*, *b*, *a*
and *R12* = *A1*: *c*, [*a*, *b*] *A2*: *b*, *a*, *c* *A3*: *c*, *b*, *a*
and *R13* = *A1*: [*a*, *c*], *b* *A2*: *b*, *a*, *c* *A3*: *c*, *b*, *a*
and *R14* = *A1*: *a*, [*b*, *c*] *A2*: *c*, [*a*, *b*] *A3*: *a*, *c*, *b*
and *R15* = *A1*: [*b*, *c*], *a* *A2*: *a*, [*b*, *c*] *A3*: *a*, *b*, *c*
and *R16* = *A1*: [*a*, *b*], *c* *A2*: *c*, [*a*, *b*] *A3*: *a*, *b*, *c*
and *R17* = *A1*: *a*, [*b*, *c*] *A2*: *a*, *b*, *c* *A3*: *b*, *c*, *a*
and *R18* = *A1*: [*a*, *c*], *b* *A2*: *b*, [*a*, *c*] *A3*: *b*, *a*, *c*
and *R19* = *A1*: *a*, [*b*, *c*] *A2*: *c*, [*a*, *b*] *A3*: *a*, *b*, *c*
and *R20* = *A1*: *b*, [*a*, *c*] *A2*: *a*, *b*, *c* *A3*: *b*, *a*, *c*
and *R21* = *A1*: [*b*, *c*], *a* *A2*: *a*, *b*, *c* *A3*: *b*, *c*, *a*
⟨*proof*⟩

lemmas *R-wfs* =
R1.wf R2.wf R3.wf R4.wf R5.wf R6.wf R7.wf R8.wf R9.wf R10.wf R11.wf R12.wf R13.wf R14.wf R15.wf
R16.wf R17.wf R18.wf R19.wf R20.wf R21.wf

lemmas *R-evals* =
R1.eval R2.eval R3.eval R4.eval R5.eval R6.eval R7.eval R8.eval R9.eval R10.eval R11.eval R12.eval R13.eval
R14.eval R15.eval R16.eval R17.eval R18.eval R19.eval R20.eval R21.eval

lemmas *nonemptiness* = *R-wfs* [*THEN scf-nonempty′*, *unfolded Bex-alts*]

We show the support conditions from Pareto efficiency

lemma [*simp*]: $a \notin$ *scf R1* ⟨*proof*⟩
lemma [*simp*]: $a \notin$ *scf R2* ⟨*proof*⟩
lemma [*simp*]: $a \notin$ *scf R3* ⟨*proof*⟩
lemma [*simp*]: $a \notin$ *scf R6* ⟨*proof*⟩
lemma [*simp*]: $a \notin$ *scf R7* ⟨*proof*⟩
lemma [*simp*]: $b \notin$ *scf R8* ⟨*proof*⟩

lemma [*simp*]: $c \notin$ *scf R9* ⟨*proof*⟩
lemma [*simp*]: $a \notin$ *scf R12* ⟨*proof*⟩
lemma [*simp*]: $b \notin$ *scf R14* ⟨*proof*⟩
lemma [*simp*]: $c \notin$ *scf R15* ⟨*proof*⟩
lemma [*simp*]: $b \notin$ *scf R16* ⟨*proof*⟩
lemma [*simp*]: $c \notin$ *scf R17* ⟨*proof*⟩
lemma [*simp*]: $c \notin$ *scf R18* ⟨*proof*⟩
lemma [*simp*]: $b \notin$ *scf R19* ⟨*proof*⟩
lemma [*simp*]: $c \notin$ *scf R20* ⟨*proof*⟩
lemma [*simp*]: $c \notin$ *scf R21* ⟨*proof*⟩

We derive the strategyproofness conditions:

lemma *s41*: ¬ *scf R4* ≻[*Fishb*(*R1 A2*)] *scf R1*
⟨*proof*⟩

lemma *s32*: ¬ *scf R3* ≻[*Fishb*(*R2 A1*)] *scf R2*
⟨*proof*⟩

lemma *s122*: ¬ *scf R12* ≻[*Fishb*(*R2 A2*)] *scf R2*
⟨*proof*⟩

lemma *s133*: ¬ *scf R13* ≻[*Fishb*(*R3 A2*)] *scf R3*
⟨*proof*⟩

lemma *s102*: ¬ *scf R10* ≻[*Fishb*(*R2 A2*)] *scf R2*
⟨*proof*⟩

lemma *s13*: ¬ *scf R1* ≻[*Fishb*(*R3 A3*)] *scf R3*
⟨*proof*⟩

lemma *s54*: ¬ *scf R5* ≻[*Fishb*(*R4 A1*)] *scf R4*
⟨*proof*⟩

lemma *s174*: ¬ *scf R17* ≻[*Fishb*(*R4 A1*)] *scf R4*
⟨*proof*⟩

lemma *s74*: ¬ *scf R7* ≻[*Fishb*(*R4 A2*)] *scf R4*
⟨*proof*⟩

lemma *s114*: ¬ *scf R11* ≻[*Fishb*(*R4 A3*)] *scf R4*
⟨*proof*⟩

lemma *s45*: ¬ *scf R4* ≻[*Fishb*(*R5 A1*)] *scf R5*
⟨*proof*⟩

lemma *s65*: ¬ *scf R6* ≻[*Fishb*(*R5 A2*)] *scf R5*
⟨*proof*⟩

lemma *s105*: ¬ *scf R10* ≻[*Fishb*(*R5 A3*)] *scf R5*

⟨*proof*⟩

lemma *s67*: $\neg$ *scf R6* $\succ$[*Fishb*(*R7 A1*)] *scf R7*
⟨*proof*⟩

lemma *s187*: $\neg$ *scf R18* $\succ$[*Fishb*(*R7 A3*)] *scf R7*
⟨*proof*⟩

lemma *s219*: $\neg$ *scf R21* $\succ$[*Fishb*(*R9 A2*)] *scf R9*
⟨*proof*⟩

lemma *s1011*: $\neg$ *scf R10* $\succ$[*Fishb*(*R11 A1*)] *scf R11*
⟨*proof*⟩

lemma *s1012*: $\neg$ *scf R10* $\succ$[*Fishb*(*R12 A2*)] *scf R12*
⟨*proof*⟩

lemma *s1213*: $\neg$ *scf R12* $\succ$[*Fishb*(*R13 A1*)] *scf R13*
⟨*proof*⟩

lemma *s1113*: $\neg$ *scf R11* $\succ$[*Fishb*(*R13 A2*)] *scf R13*
⟨*proof*⟩

lemma *s1813*: $\neg$ *scf R18* $\succ$[*Fishb*(*R13 A3*)] *scf R13*
⟨*proof*⟩

lemma *s814*: $\neg$ *scf R8* $\succ$[*Fishb*(*R14 A2*)] *scf R14*
⟨*proof*⟩

lemma *s1914*: $\neg$ *scf R19* $\succ$[*Fishb*(*R14 A3*)] *scf R14*
⟨*proof*⟩

lemma *s1715*: $\neg$ *scf R17* $\succ$[*Fishb*(*R15 A1*)] *scf R15*
⟨*proof*⟩

lemma *s815*: $\neg$ *scf R8* $\succ$[*Fishb*(*R15 A3*)] *scf R15*
⟨*proof*⟩

lemma *s516*: $\neg$ *scf R5* $\succ$[*Fishb*(*R16 A1*)] *scf R16*
⟨*proof*⟩

lemma *s517*: $\neg$ *scf R5* $\succ$[*Fishb*(*R17 A1*)] *scf R17*
⟨*proof*⟩

lemma *s1619*: $\neg$ *scf R16* $\succ$[*Fishb*(*R19 A1*)] *scf R19*
⟨*proof*⟩

lemma *s1820*: $\neg$ *scf R18* $\succ$[*Fishb*(*R20 A2*)] *scf R20*
⟨*proof*⟩

lemma *s920*: $\neg$ *scf R9* $\succ$[*Fishb*(*R20 A3*)] *scf R20*
⟨*proof*⟩

lemma *s521*: $\neg$ *scf R5* $\succ$[*Fishb*(*R21 A1*)] *scf R21*
⟨*proof*⟩

lemma *s421*: $\neg$ *scf R4* $\succ$[*Fishb*(*R21 A1*)] *scf R21*
⟨*proof*⟩

lemmas *sp* = *s41 s32 s122 s102 s133 s13 s54 s174 s54 s74 s114 s45 s65 s105 s67 s187 s219 s1011 s1012 s1213 s1113 s1813 s814 s1914 s1715 s815 s516 s517 s1619 s1820 s920 s521 s421*

We now use the simplifier to break down the strategyproofness conditions into SAT formulae. This takes a few seconds, so we use some low-level ML code to at least do the simplification in parallel.

⟨*ML*⟩

We show that the strategyproofness conditions, the non-emptiness conditions (i. e. every SCF must return at least one winner), and the efficiency conditions are not satisfiable together, which means that the SCF whose existence we assumed simply cannot exist.

theorem *absurd*: *False*
⟨*proof*⟩

end

2.3 Lifting to more than 3 agents and alternatives

We now employ the standard lifting argument outlined before to lift this impossibility from 3 agents and alternatives to any setting with at least 3 agents and alternatives.

locale *fb-impossibility* =
strategyproof-anonymous-scf agents alts scf Fishb +
pareto-efficient-scf agents alts scf
for *agents* :: *'agent set* **and** *alts* :: *'alt set* **and** *scf* +
assumes *card-agents-ge*: *card agents* $\geq$ *3*
and *card-alts-ge*: *card alts* $\geq$ *3*
begin

lemma *finite-list'*:
assumes *finite A*
obtains *xs* **where** *A = set xs distinct xs length xs = card A*
⟨*proof*⟩

lemma *finite-list-subset*:
  assumes *finite A card A* $\geq$ *n*
  obtains *xs* **where** *set xs* $\subseteq$ *A distinct xs length xs* = *n*
⟨*proof*⟩

lemma *card-ge-3E*:
  assumes *finite A card A* $\geq$ *3*
  obtains *a b c* **where** *distinct* [*a*,*b*,*c*] {*a*,*b*,*c*} $\subseteq$ *A*
⟨*proof*⟩

theorem *absurd*: *False*
⟨*proof*⟩

end

end

References

[1] F. Brandt, C. Saile, and C. Stricker. Voting with ties: Strong impossibilities via SAT solving. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. IFAAMAS, 2018. Forthcoming.