The Incompatibility of Strategy-Proofness and Representation in Party-Approval Multi-Winner Elections

Théo Delemazure
Tom Demeulemeester
Manuel Eberl
Jonas Israel
Patrick Lederer

March 24, 2023

In party-approval multi-winner elections, the goal is to allocate the seats of a fixed-size committee to parties based on approval ballots of the voters over the parties. In particular, each voter can approve multiple parties and each party can be assigned multiple seats.

Three central requirements in this settings are:

Anonymity: The result is invariant under renaming the voters.

Representation: Every sufficiently large group of voters with similar preferences is represented by some committee members.

Strategy-proofness: No voter can benefit by misreporting her true preferences.

We show that these three basic axioms are incompatible for party-approval multi-winner voting rules, thus proving a far-reaching impossibility theorem.

The proof of this result is obtained by formulating the problem in propositional logic and then letting a SAT solver show that the formula is unsatisfiable. The DRUP proof output by the SAT solver is then converted into Lammich's GRAT format and imported into Isabelle/HOL with some custom-written ML code.

This transformation is proof-producing, so the final Isabelle/HOL theorem does not rely on any oracles or other trusted external trusted components.

Contents

1 Auxiliary Facts About Multisets

theory *PAPP-Multiset-Extras*
imports *HOL−Library.Multiset*
begin

This section contains a number of not particularly interesting small facts about multisets.

lemma *mset-set-subset-iff*: *finite* $A \Longrightarrow$ *mset-set* $A \subseteq\# B \longleftrightarrow A \subseteq$ *set-mset* B
⟨*proof*⟩

lemma *mset-subset-size-ge-imp-eq*:
assumes $A \subseteq\# B$ *size* $A \geq$ *size* B
shows $A = B$
⟨*proof*⟩

lemma *mset-psubset-iff*:
$X \subset\# Y \longleftrightarrow X \subseteq\# Y \wedge (\exists x.$ *count* $X\ x <$ *count* $Y\ x)$
⟨*proof*⟩

lemma *count-le-size*: *count* $A\ x \leq$ *size* A
⟨*proof*⟩

lemma *size-filter-eq-conv-count* [*simp*]: *size* (*filter-mset* $(\lambda y.\ y = x)\ A$) = *count* $A\ x$
⟨*proof*⟩

lemma *multiset-filter-mono′*:
assumes $\bigwedge x.\ x \in\# A \Longrightarrow P\ x \Longrightarrow Q\ x$
shows *filter-mset* $P\ A \subseteq\#$ *filter-mset* $Q\ A$
⟨*proof*⟩

lemma *multiset-filter-mono″*:
assumes $A \subseteq\# B \bigwedge x.\ x \in\# A \Longrightarrow P\ x \Longrightarrow Q\ x$
shows *filter-mset* $P\ A \subseteq\#$ *filter-mset* $Q\ B$
⟨*proof*⟩

lemma *filter-mset-disjunction*:
assumes $\bigwedge x.\ x \in\# X \Longrightarrow P\ x \Longrightarrow Q\ x \Longrightarrow$ *False*
shows *filter-mset* $(\lambda x.\ P\ x \vee Q\ x)\ X =$ *filter-mset* $P\ X +$ *filter-mset* $Q\ X$
⟨*proof*⟩

lemma *size-mset-sum-mset*: *size* (*sum-mset* X) = $(\sum x \in\# X.$ *size* $(x ::$ *′a multiset*$))$
⟨*proof*⟩

lemma *count-sum-mset*: *count* (*sum-mset* X) $x = (\sum Y \in\# X.$ *count* $Y\ x)$
⟨*proof*⟩

lemma *replicate-mset-rec*: $n > 0 \Longrightarrow$ *replicate-mset* $n\ x =$ *add-mset* x (*replicate-mset* $(n - 1)$ x)
⟨*proof*⟩

lemma *add-mset-neq*: $x \notin\# B \Longrightarrow$ *add-mset* $x\ A \neq B$
⟨*proof*⟩

lemma *filter-replicate-mset*:
filter-mset P (*replicate-mset* $n\ x$) = (*if* $P\ x$ *then replicate-mset* $n\ x$ *else* $\{\#\}$)
⟨*proof*⟩

lemma *filter-diff-mset′*: *filter-mset* $P\ (X - Y)$ = *filter-mset* $P\ X - Y$
⟨*proof*⟩

lemma *in-diff-multiset-absorb2*: $x \notin\# B \Longrightarrow x \in\# A - B \longleftrightarrow x \in\# A$
⟨*proof*⟩

end

2 Anonymous Party Approval Rules

theory *Anonymous-PAPP*
imports *Complex-Main Randomised-Social-Choice.Order-Predicates PAPP-Multiset-Extras*
begin

In this section we will define (anonymous) P-APP rules and some basic desirable properties of P-APP rules.

2.1 Definition of the General Setting

The following locale encapsulates an anonymous *party approval election*; that is:

- a number of voters
- a set of parties
- the size of the desired committee

The number of parties and voters is assumed to be finite and non-zero. As a modelling choice, we do not distinguish the voters at all; there is no explicit set of voters. We only care about their number.

locale *anon-papp-election* =
fixes *n-voters* :: *nat* **and** *parties* :: *'a set* **and** *committee-size* :: *nat*
assumes *finite-parties* [*simp*, *intro*]: *finite parties*
assumes *n-voters-pos*: *n-voters* > *0*
assumes *nonempty-parties* [*simp*]: *parties* ≠ {}
begin

The result of a P-APP election is a committee, i.e. a multiset of parties with the desired size.

definition *is-committee* :: *'a multiset* ⇒ *bool* **where**
is-committee $W \longleftrightarrow$ *set-mset* $W \subseteq$ *parties* ∧ *size* W = *committee-size*

end

A *preference profile* for a P-APP collection consists of one approval list (i.e. a set of approved parties) for each voter. Since we are in an anonymous setting, this means that we have a *multiset* consisting of n sets of parties (where n is the number of voters).

Moreover, we make the usual assumption that the approval lists must be non-empty.

locale *anon-papp-profile* = *anon-papp-election* +
  fixes A :: *'a set multiset*
  assumes *A-subset*: $\bigwedge X.\ X \in\# A \Longrightarrow X \subseteq$ *parties*
  assumes *A-nonempty*: $\{\} \notin\# A$
  assumes *size-A*: *size* $A =$ *n-voters*
begin

lemma *A-nonempty'*: $A \neq \{\#\}$
  ⟨*proof*⟩

end

context *anon-papp-election*
begin

abbreviation
  is-pref-profile **where** *is-pref-profile* ≡ *anon-papp-profile n-voters parties*

lemma *is-pref-profile-iff*:
  is-pref-profile $A \longleftrightarrow$ *set-mset* $A \subseteq$ *Pow parties* $- \{\{\}\} \wedge$ *size* $A =$ *n-voters*
  ⟨*proof*⟩

lemma *not-is-pref-profile-empty* [*simp*]: ¬*is-pref-profile* $\{\#\}$
  ⟨*proof*⟩

The following relation is a key definition: it takes an approval list A and turns it into a preference relation on committees. A committee is to be at least as good as another if the number of approved parties in it is at least as big.

This relation is a reflexive, transitive, and total.

definition *committee-preference* :: *'a set* ⇒ *'a multiset relation* (*Comm*) **where**
  W1 $\preceq$[*Comm*(*A*)] *W2* $\longleftrightarrow$ *size* $\{\#\ x \in\# W1.\ x \in A\ \#\} \leq$ *size* $\{\#\ x \in\# W2.\ x \in A\ \#\}$

lemma *not-strict-Comm* [*simp*]: ¬(*W1* $\prec$[*Comm*(*A*)] *W2*) $\longleftrightarrow$ *W1* $\succeq$[*Comm*(*A*)] *W2*
  ⟨*proof*⟩

lemma *not-weak-Comm* [*simp*]: ¬(*W1* $\preceq$[*Comm*(*A*)] *W2*) $\longleftrightarrow$ *W1* $\succ$[*Comm*(*A*)] *W2*
  ⟨*proof*⟩

sublocale *Comm*: *preorder Comm*(*A*) $\lambda x\ y.\ x \prec$[*Comm*(*A*)] y

⟨*proof*⟩

lemma *strong-committee-preference-iff*:
W1 ≺[*Comm*(*A*)] *W2* ⟷ *size* {# *x*∈#*W1*. *x* ∈ *A* #} < *size* {# *x*∈#*W2*. *x* ∈ *A* #}
⟨*proof*⟩

We also define the Pareto ordering on parties induced by a given preference profile: One party is at least as good (in the Pareto relation) as another if all voters agree that it is at least as good. That is, $y \succeq x$ in the Pareto ordering if all voters who approve x also approve y.

This relation is also reflexive and transitive.

definition *Pareto* :: *'a set multiset* ⇒ *'a relation* **where**
x ⪯[*Pareto*(*A*)] *y* ⟷ *x* ∈ *parties* ∧ *y* ∈ *parties* ∧ (∀ *X*∈#*A*. *x* ∈ *X* ⟶ *y* ∈ *X*)

sublocale *Pareto*: *preorder-on parties Pareto A*
⟨*proof*⟩

Pareto losers are parties that are (strictly) Pareto-dominated, i.e. there exists some other party that all voters consider to be at least as good and at least one voter considers it to be strictly better.

definition *pareto-losers* :: *'a set multiset* ⇒ *'a set* **where**
pareto-losers A = {*x*. ∃ *y*. *y* ≻[*Pareto*(*A*)] *x*}

end

2.2 P-APP rules and Desirable Properties

The following locale describes a P-APP rule. This is simply a function that maps every preference profile to a committee of the desired size.

Note that in our setting, a P-APP rule has a fixed number of voters, a fixed set of parties, and a fixed desired committee size.

locale *anon-papp* = *anon-papp-election* +
fixes *r* :: *'a set multiset* ⇒ *'a multiset*
assumes *rule-wf*: *is-pref-profile A* ⟹ *is-committee* (*r A*)

2.3 Efficiency

Efficiency is a common notion in Social Choice Theory. The idea is that if a party is "obviously bad", then it should not be chosen. What "obviously bad" means depends on the precise notion of Efficiency that is used. We will talk about two notions: Weak Efficiency and Pareto Efficiency.

A P-APP rule is *weakly efficient* if a party that is approved by no one is never part of the output committee.

Note that approval lists must be non-empty, so there is always at least one party that is approved by at least one voter.

locale *weakly-efficient-anon-papp* = *anon-papp* +
  assumes *weakly-efficient*: *is-pref-profile A* $\Longrightarrow \forall X{\in}\#A.\ x \notin X \Longrightarrow x \notin\# r\ A$

A P-APP rule is *Pareto-efficient* if a Pareto-dominated party is never part of the output committee.

locale *pareto-optimal-anon-papp* = *anon-papp* +
  assumes *pareto-optimal*: *is-pref-profile A* $\Longrightarrow x \in$ *pareto-losers A* $\Longrightarrow x \notin\# r\ A$
begin

Pareto-efficiency implies weak efficiency:

sublocale *weakly-efficient-anon-papp*
⟨*proof*⟩

end

2.4 Strategyproofness

Strategyproofness is another common notion in Social Choice Theory that generally encapsulates the notion that an voter should not be able to manipulate the outcome of an election in their favour by (unilaterally) submitting fake preferences; i.e. reporting one's preferences truthfully should always be the optimal choice.

A P-APP rule is called *cardinality-strategyproof* if an voter cannot obtain a better committee (i.e. one that contains strictly more of their approved parties) by submitting an approval list that is different from their real approval list.

To make the definition simpler, we first define the notion of *manipulability*: in the context of a particular P-APP rule r, a preference profile A is said to be manipulable by the voter i with the fake preference list Y if $r(A(i := Y))$ contains strictly more parties approved by i than $r(A)$.

Since we have anonymous profiles and do not talk about particular voters, we replace i with their approval list X. Since A is a multiset, the definition of manipulability becomes $r(A - \{X\} + \{Y\}) \succ_X r(A)$.

definition (**in** *anon-papp*) *card-manipulable* **where**
  card-manipulable A X Y $\longleftrightarrow$
    is-pref-profile A $\wedge$ $X \in\#$ A $\wedge$ $Y \neq \{\}$ $\wedge$ $Y \subseteq$ *parties* $\wedge$ $r\ (A - \{\#X\#\} + \{\#Y\#\})$
$\succ[Comm(X)]\ r\ A$

A technical (and fairly obvious) lemma: replacing an voter's approval list with a different approval list again yields a valid preference profile.

lemma (**in** *anon-papp*) *is-pref-profile-replace*:
  assumes *is-pref-profile A* **and** $X \in\# A$ **and** $Y \neq \{\}$ **and** $Y \subseteq$ *parties*
  shows *is-pref-profile* $(A - \{\#X\#\} + \{\#Y\#\})$
⟨*proof*⟩

locale *card-stratproof-anon-papp* = *anon-papp* +
  assumes *not-manipulable*: ¬*card-manipulable A X Y*

begin

The two following alternative versions of non-manipulability are somewhat nicer to use in practice.

lemma *not-manipulable'*:
  assumes *is-pref-profile A is-pref-profile A' A + {#Y#} = A' + {#X#}*
  shows $\neg(r\ A' \succ[Comm(X)]\ r\ A)$
⟨*proof*⟩

lemma *not-manipulable''*:
  assumes *is-pref-profile A is-pref-profile A' A + {#Y#} = A' + {#X#}*
  shows $r\ A' \preceq[Comm(X)]\ r\ A$
  ⟨*proof*⟩

end

2.5 Representation

Representation properties are in a sense the opposite of Efficiency properties: if a sufficiently high voters agree that certain parties are good, then these should, to some extent, be present in the result. For instance, if we have 20 voters and 5 of them approve parties A and B, then if the output committee has size 4, we would expect either A or B to be in the committee to ensure that these voters' preferences are represented fairly.

Weak representation is a particularly weak variant of this that states that if at least one k-th of the voters (where k is the size of the output committee) approve only a single party x, then x should be in the committee at least once:

locale *weak-rep-anon-papp* =
  anon-papp n-voters parties committee-size r
  for *n-voters* **and** *parties* :: *'alt set* **and** *committee-size* :: *nat* **and** *r* +
  assumes *weak-representation*:
    is-pref-profile A ⟹ *committee-size* ∗ *count A {x}* ≥ *n-voters* ⟹ *x* ∈# *r A*

The following alternative definition of Weak Representation is a bit closer to the definition given in the paper.

lemma *weak-rep-anon-papp-altdef*:
  weak-rep-anon-papp n-voters parties committee-size r ⟷
    anon-papp n-voters parties committee-size r ∧ (*committee-size* = *0* ∨
    (∀ *A x*. *anon-papp-profile n-voters parties A* ⟶
          count A {x} ≥ *n-voters* / *committee-size* ⟶ *x* ∈# *r A*))
  ⟨*proof*⟩

Justified Representation is a stronger notion which demands that if there is a subgroup of voters that comprises at least one k-th of all voters and for which the intersection of their approval lists is some nonempty set X, then at least one of the parties approved by at least one voter in that subgroup must be in the result committee.

locale *justified-rep-anon-papp* =

anon-papp n-voters parties committee-size r
for *n-voters* **and** *parties* :: *'alt set* **and** *committee-size* :: *nat* **and** *r* +
assumes *justified-representation*:
is-pref-profile A $\Longrightarrow$ *G* $\subseteq\#$ *A* $\Longrightarrow$ *committee-size* $*$ *size G* $\geq$ *n-voters* $\Longrightarrow$
$(\bigcap X \in$*set-mset G*. *X*$) \neq \{\} \Longrightarrow \exists X\ x.\ X \in\#\ G \wedge x \in X \wedge x \in\#\ r\ A$
begin

Any rule that satisfies Justified Representation also satisfies Weak Representation

sublocale *weak-rep-anon-papp*
⟨*proof*⟩

end

locale *card-stratproof-weak-rep-anon-papp* =
card-stratproof-anon-papp + *weak-rep-anon-papp*

2.6 Proportional Representation

The notions of Representation we have seen so far are fairly week in that they only demand that certain parties be in the committee at least once if enough voters approve them. Notions of Proportional Representation strengthen this by demanding that if a sufficiently large subgroup of voters approve some parties, then these voters must be represented in the result committe not just once, but to a degree proportional to the size of that subgroup of voters.

For Weak Representation, the proportional generalization is fairly simple: if a fraction of at least $\frac{ln}{k}$ of the voters uniquely approve a party x, then x must be in the committee at least l times.

locale *weak-prop-rep-anon-papp* =
anon-papp n-voters parties committee-size r
for *n-voters* **and** *parties* :: *'alt set* **and** *committee-size* :: *nat* **and** *r* +
assumes *weak-proportional-representation*:
is-pref-profile A $\Longrightarrow$ *committee-size* $*$ *count A* $\{x\} \geq l *$ *n-voters* $\Longrightarrow$ *count* $(r\ A)\ x \geq l$
begin

sublocale *weak-rep-anon-papp*
⟨*proof*⟩

end

Similarly, Justified *Proportional* Representation demands that if the approval lists of a subgroup of at least $\frac{ln}{k}$ voters have a non-empty intersection, then at least l parties in the result committee are each approved by at least one of the voters in the subgroup.

locale *justified-prop-rep-anon-papp* =
anon-papp n-voters parties committee-size r
for *n-voters* **and** *parties* :: *'alt set* **and** *committee-size* :: *nat* **and** *r* +
assumes *justified-proportional-representation*:

is-pref-profile A $\Longrightarrow$ *G* $\subseteq\#$ *A* $\Longrightarrow$ *committee-size* $*$ *size G* $\geq$ *l* $*$ *n-voters* $\Longrightarrow$
$(\bigcap X \in$ *set-mset G. X*$) \neq \{\} \Longrightarrow$ *size* $\{\#\ x \in\#$ *r A. x* $\in (\bigcup X \in$ *set-mset G. X*$)\ \#\} \geq$ *l*
begin

sublocale *justified-rep-anon-papp*
⟨*proof*⟩

sublocale *weak-prop-rep-anon-papp*
⟨*proof*⟩

end

locale *card-stratproof-weak-prop-rep-anon-papp* =
card-stratproof-anon-papp + *weak-prop-rep-anon-papp*

end

3 The Base Case of the Impossibility

theory *PAPP-Impossibility-Base-Case*
imports *Anonymous-PAPP SAT-Replay*
begin

In this section, we will prove the base case of our P-APP impossibility result, namely that there exists no anonymous P-APP rule f for 6 voters, 4 parties, and committee size 3 that satisfies Weak Representation and Cardinality Strategyproofness.

The proof works by looking at some (comparatively small) set of preference profiles and the set of all 20 possible output committees. Each proposition $f(A) = C$ (where A is a profile from our set and C is one of the 20 possible output committees) is considered as a Boolean variable.

All the conditions arising on these variables based on the fact that f is a function and the additional properties (Representation, Strategyproofness) are encoded as SAT clauses. This SAT problem is then proven unsatisfiable by an external SAT solver and the resulting proof re-imported into Isabelle/HOL.

3.1 Auxiliary Material

We define the set of committees of the given size k for a given set of parties P.

definition *committees* :: *nat* $\Rightarrow$ *'a set* $\Rightarrow$ *'a multiset set* **where**
committees k P = $\{W.$ *set-mset W* $\subseteq$ *P* $\wedge$ *size W* = *k*$\}$

We now prove a recurrence for this set so that we can more easily compute the set of all possible committees:

lemma *committees-0* [*simp*]: *committees 0 P* = $\{\{\#\}\}$
⟨*proof*⟩

lemma *committees-Suc*:
committees (*Suc n*) *P* = (⋃*x*∈*P*. ⋃ *W*∈*committees n P*. {{#*x*#} + *W*})
⟨*proof*⟩

The following function takes a list $[a_1, \ldots, a_n]$ and computes the list of all pairs of the form (a_i, a_j) with $i < j$:

fun *pairs* :: '*a list* ⇒ ('*a* × '*a*) *list* **where**
pairs [] = []
| *pairs* (*x* # *xs*) = *map* (λ*y*. (*x*, *y*)) *xs* @ *pairs xs*

lemma *distinct-conv-pairs*: *distinct xs* ⟷ *list-all* (λ(*x*,*y*). *x* ≠ *y*) (*pairs xs*)
⟨*proof*⟩

lemma *list-ex-unfold*: *list-ex P* (*x* # *y* # *xs*) ⟷ *P x* ∨ *list-ex P* (*y* # *xs*) *list-ex P* [*x*] ⟷ *P x*
⟨*proof*⟩

lemma *list-all-unfold*: *list-all P* (*x* # *y* # *xs*) ⟷ *P x* ∧ *list-all P* (*y* # *xs*) *list-all P* [*x*] ⟷ *P x*
⟨*proof*⟩

3.2 Setup for the Base Case

We define a locale for an anonymous P-APP rule for 6 voters, 4 parties, and committee size 3 that satisfies weak representation and cardinality strategyproofness. Our goal is to prove the theorem *False* inside this locale.

locale *papp-impossibility-base-case* =
card-stratproof-weak-rep-anon-papp 6 parties 3 r
for *parties* :: '*a set* **and** *r* +
assumes *card-parties*: *card parties* = *4*
begin

A slightly more convenient version of Weak Representation:

lemma *weak-representation'*:
assumes *is-pref-profile A A'* ≡ *A* ∀ *z*∈*Z*. *count A* {*z*} ≥ *2* ¬*Z* ⊆ *set-mset W*
shows *r A'* ≠ *W*
⟨*proof*⟩

The following lemma (Lemma 2 in the appendix of the paper) is a strengthening of Weak Representation and Strategyproofness in our concrete setting:

Let A be a preference profile containing approval lists X and let Z be a set of parties such that each element of Z is uniquely approved by at least two voters in A. Due to Weak Representation, at least $|X \cap Z|$ members of the committee are then approved by X.

What the lemma now says is that if there exists another voter with approval list $Y \subseteq X$ and $Y \nsubseteq Z$, then there is an additional committee member that is approved by X.

This lemma will be used both in our symmetry-breaking argument and as a means to add more clauses to the SAT instance. Since these clauses are logical consequences of Strategyproofness and Weak Representation, they are technically redundant – but their presence allows us to use consider a smaller set of profiles and still get a contradiction. Without using the lemma, we would need to feed more profiles to the SAT solver to obtain the same information.

lemma *lemma2*:
assumes *A*: *is-pref-profile A*
assumes $X \in\# A$ **and** $Y \in\# A - \{\#X\#\}$ **and** $Y \subseteq X$ **and** $\neg Y \subseteq Z$
assumes *Z*: $\forall z{\in}Z.$ *count A* $\{z\} \geq 2$
shows *size* (*filter-mset* $(\lambda x.\ x \in X)$ $(r\ A)) >$ *card* $(X \cap Z)$
⟨*proof*⟩

The following are merely reformulation of the above lemma for technical reasons.

lemma *lemma2′*:
assumes *is-pref-profile A*
assumes $\forall z{\in}Z.$ *count A* $\{z\} \geq 2$
assumes $X \in\# A \wedge (\exists\ Y.\ Y \in\# A - \{\#X\#\} \wedge Y \subseteq X \wedge \neg Y \subseteq Z)$
shows $\neg$*filter-mset* $(\lambda x.\ x \in X)$ $(r\ A) \subseteq\#$ *mset-set* $(X \cap Z)$
⟨*proof*⟩

lemma *lemma2′′*:
assumes *is-pref-profile A*
assumes $A' \equiv A$
assumes $\forall z{\in}Z.$ *count A* $\{z\} \geq 2$
assumes $X \in\# A \wedge (\exists\ Y{\in}$*set-mset* $(A - \{\#X\#\}).\ Y \subseteq X \wedge \neg Y \subseteq Z)$
assumes *filter-mset* $(\lambda x.\ x \in X)$ $W \subseteq\#$ *mset-set* $(X \cap Z)$
shows $r\ A' \neq W$
⟨*proof*⟩

3.3 Symmetry Breaking

In the following, we formalize the symmetry-breaking argument that shows that we can reorder the four alternatives C_1 to C_4 in such a way that the preference profile

$$\{C_1\}\ \ \{C_2\}\ \ \{C_1, C_2\}\ \ \{C_3\}\ \ \{C_3\}\ \ \{C_3, C_4\}$$

is mapped to one of the committees $[C_1, C_1, C_3]$ or $[C_1, C_2, C_3]$.

We start with a simple technical lemma that states that if we have a multiset A of size 3 consisting of the elements x and y and x occurs at least as often as y, then $A = [x, x, y]$.

lemma *papp-multiset-3-aux*:
assumes *size A* $= 3$ $x \in\# A$ $y \in\# A$ *set-mset A* $\subseteq \{x, y\}$ $x \neq y$ *count A x* $\geq$ *count A y*
shows $A = \{\#x, x, y\#\}$
⟨*proof*⟩

The following is the main symmetry-breaking result. It shows that we can find parties C_1 to C_4 with the desired property.

This is a somewhat ad-hoc argument; in the appendix of the paper this is done more systematically in Lemma 3.

lemma *symmetry-break-aux*:
obtains *C1 C2 C3 C4* **where**
parties = {*C1*, *C2*, *C3*, *C4*} *distinct* [*C1*, *C2*, *C3*, *C4*]
r ({#{*C1*}, {*C2*}, {*C1*, *C2*}, {*C3*}, {*C4*}, {*C3*, *C4*}#}) ∈ {{#*C1*, *C1*, *C3*#}, {#*C1*, *C2*, *C3*#}}
⟨*proof*⟩

We now use the choice operator to get our hands on such values C_1 to C_4.

definition *C1234* **where**
C1234 = (*SOME xs*. *set xs* = *parties* ∧ *distinct xs* ∧
(*case xs of* [*C1*, *C2*, *C3*, *C4*] ⇒
r ({#{*C1*}, {*C2*}, {*C1*, *C2*}, {*C3*}, {*C4*}, {*C3*, *C4*}#}) ∈ {{#*C1*, *C1*, *C3*#}, {#*C1*, *C2*, *C3*#}}))

definition *C1* **where** *C1* = *C1234* ! *0*
definition *C2* **where** *C2* = *C1234* ! *1*
definition *C3* **where** *C3* = *C1234* ! *2*
definition *C4* **where** *C4* = *C1234* ! *3*

lemma *distinct*: *distinct* [*C1*, *C2*, *C3*, *C4*]
and *parties-eq*: *parties* = {*C1*, *C2*, *C3*, *C4*}
and *symmetry-break*:
r ({#{*C1*}, {*C2*}, {*C1*, *C2*}, {*C3*}, {*C4*}, {*C3*, *C4*}#}) ∈ {{#*C1*, *C1*, *C3*#}, {#*C1*, *C2*, *C3*#}}
⟨*proof*⟩

lemma *distinct'* [*simp*]:
C1 ≠ *C2* *C1* ≠ *C3* *C1* ≠ *C4* *C2* ≠ *C1* *C2* ≠ *C3* *C2* ≠ *C4*
C3 ≠ *C1* *C3* ≠ *C2* *C3* ≠ *C4* *C4* ≠ *C1* *C4* ≠ *C2* *C4* ≠ *C3*
⟨*proof*⟩

lemma *in-parties* [*simp*]: *C1* ∈ *parties* *C2* ∈ *parties* *C3* ∈ *parties* *C4* ∈ *parties*
⟨*proof*⟩

3.4 The Set of Possible Committees

Next, we compute the set of the 20 possible committees.

abbreviation *COM* **where** *COM* ≡ *committees 3 parties*

definition *COM'* **where** *COM'* =
[{#*C1*, *C1*, *C1*#}, {#*C1*, *C1*, *C2*#}, {#*C1*, *C1*, *C3*#}, {#*C1*, *C1*, *C4*#},
{#*C1*, *C2*, *C2*#}, {#*C1*, *C2*, *C3*#}, {#*C1*, *C2*, *C4*#}, {#*C1*, *C3*, *C3*#},
{#*C1*, *C3*, *C4*#}, {#*C1*, *C4*, *C4*#}, {#*C2*, *C2*, *C2*#}, {#*C2*, *C2*, *C3*#},
{#*C2*, *C2*, *C4*#}, {#*C2*, *C3*, *C3*#}, {#*C2*, *C3*, *C4*#}, {#*C2*, *C4*, *C4*#},
{#*C3*, *C3*, *C3*#}, {#*C3*, *C3*, *C4*#}, {#*C3*, *C4*, *C4*#},
{#*C4*, *C4*, *C4*#}]

lemma *distinct-COM'*: *distinct COM'*
⟨*proof*⟩

lemma *COM-eq*: *COM* = *set COM'*
⟨*proof*⟩

lemma *r-in-COM*:
assumes *is-pref-profile A*
shows *r A* ∈ *COM*
⟨*proof*⟩

lemma *r-in-COM'*:
assumes *is-pref-profile A A'* ≡ *A*
shows *list-ex* (λ*W*. *r A'* = *W*) *COM'*
⟨*proof*⟩

lemma *r-right-unique*:
list-all (λ(*W1*,*W2*). *r A* ≠ *W1* ∨ *r A* ≠ *W2*) (*pairs COM'*)
⟨*proof*⟩

end

3.5 Generating Clauses and Replaying the SAT Proof

We now employ some custom-written ML code to generate all the SAT clauses arising from the given profiles (read from an external file) as Isabelle/HOL theorems. From these, we then derive *False* by replaying an externally found SAT proof (also written from an external file).

The proof was found with the glucose SAT solver, which outputs proofs in the DRUP format (a subset of the more powerful DRAT format). We then used the *DRAT-trim* tool by Wetzler et al. [2] to make the proof smaller. This was done repeatedly until the proof size did not decrease any longer. Then, the proof was converted into the *GRAT* format introduced by Lammich [1], which is easier to check (or in our case replay) than the less explicit DRAT (or DRUP) format.

external-file *sat-data/profiles*
external-file *sat-data/papp-impossibility.grat.xz*

context *papp-impossibility-base-case*
begin

⟨*ML*⟩

This invocation proves a theorem called *contradiction* whose statement is *False*. Note that the DIMACS version of the SAT file that is being generated can be viewed by clicking on "See theory exports" in the messages output by the invocation below.

On a 2021 desktop PC with 12 cores, proving all the clauses takes 8.4 s (multithreaded;

CPU time 55 s). Replaying the proof takes 130 s (singlethreaded).

⟨*ML*⟩

end

With this, we can now prove the impossibility result:

lemma *papp-impossibility-base-case*:
  assumes *card parties = 4*
  shows ¬*card-stratproof-weak-rep-anon-papp 6 parties 3 r*
⟨*proof*⟩

end

4 Lowering P-APP Rules to Smaller Settings

theory *Anonymous-PAPP-Lowering*
  imports *Anonymous-PAPP*
begin

In this section, we prove a number of lemmas (corresponding to Lemma 1 in the paper) that allow us to take an anonymous P-APP rule with some additional properties (typically Cardinality-Strategyproofness and Weak Representation or Weak Proportional Representation) and construct from it an anonymous P-APP rule for a different setting, i.e. different number of voters, parties, and/or result committee size.

In the reverse direction, this also allows us to lift impossibility results from one setting to another.

4.1 Preliminary Lemmas

context *card-stratproof-anon-papp*
begin

The following lemma is obtained by applying Strategyproofness repeatedly. It shows that if we have l voters with identical approval lists, then this entire group of voters has no incentive to submit wrong preferences. That is, the outcome they obtain by submitting their genuine approval lists is weakly preferred by them over all outcomes obtained where these l voters submit any other preferences (and the remaining $n - l$ voters submit the same preferences as before).

This is stronger than regular Strategyproofness, where we only demand that no voter has an incentive to submit wrong preferences *unilaterally* (and everyone else keeps the same preferences). Here we know that the entire group of l voters has no incentive to submit wrong preferences in coordination with one another.

lemma *proposition2*:
  assumes *size B = l size A + l = n-voters*
  assumes *X ≠ {} X ⊆ parties {} ∉# A+B ∀ X'∈#A+B. X' ⊆ parties*

shows $r\ (\mathit{replicate\text{-}mset}\ l\ X + A) \succeq[\mathit{Comm}(X)]\ r\ (B + A)$
⟨*proof*⟩

end

context *card-stratproof-weak-rep-anon-papp*
begin

In a setting with Weak Representation and Cardinality-Strategyproofness, Proposition 2 allows us to strengthen Weak Representation in the following way: Suppose we at least $l\lfloor n/k \rfloor$ voters with the same approval list X, and X consists of at least l parties. Then at least l of the members of the result committee are in X.

lemma *proposition3*:
assumes *is-pref-profile A X* ⊆ *parties card X* ≥ *l*
assumes *committee-size* > *0*
assumes *count A X* ≥ *l* ∗ ⌈*n-voters* / *committee-size*⌉
shows *size* {# *x*∈# *r A*. *x* ∈ *X* #} ≥ *l*
⟨*proof*⟩

end

4.2 Dividing the number of voters

If we have a PAPP rule that satisfies weak representation and cardinality strategyproofness, for ln voters, we can turn it into one for n voters. This is done by simply cloning each voter l times.

Consequently, if we have an impossibility result for n voters, it also holds for any integer multiple of n.

locale *divide-voters-card-stratproof-weak-rep-anon-papp* =
card-stratproof-weak-rep-anon-papp l ∗ *n-voters parties committee-size r*
for *l n-voters parties committee-size r*
begin

definition *lift-profile* :: *'a set multiset* ⇒ *'a set multiset* **where**
lift-profile A = (∑ *X*∈#*A*. *replicate-mset l X*)

sublocale *lowered*: *anon-papp-election n-voters parties*
⟨*proof*⟩

lemma *l-pos*: *l* > *0*
⟨*proof*⟩

lemma *empty-in-lift-profile-iff* [*simp*]: {} ∈# *lift-profile A* ⟷ {} ∈# *A*
⟨*proof*⟩

lemma *set-mset-lift-profile* [*simp*]: *set-mset* (*lift-profile A*) = *set-mset A*

⟨*proof*⟩

lemma *size-lift-profile*: *size* (*lift-profile A*) = *l* ∗ *size A*
⟨*proof*⟩

lemma *count-lift-profile* [*simp*]: *count* (*lift-profile A*) *x* = *l* ∗ *count A x*
⟨*proof*⟩

lemma *is-pref-profile-lift-profile* [*intro*]:
assumes *lowered.is-pref-profile A*
shows *is-pref-profile* (*lift-profile A*)
⟨*proof*⟩

sublocale *lowered*: *anon-papp n-voters parties committee-size r* ∘ *lift-profile*
⟨*proof*⟩

sublocale *lowered*: *weak-rep-anon-papp n-voters parties committee-size r* ∘ *lift-profile*
⟨*proof*⟩

sublocale *lowered*: *card-stratproof-anon-papp n-voters parties committee-size r* ∘ *lift-profile*
⟨*proof*⟩

sublocale *lowered*: *card-stratproof-weak-rep-anon-papp n-voters parties committee-size r* ∘ *lift-profile*
⟨*proof*⟩

end

locale *divide-voters-card-stratproof-weak-prop-rep-anon-papp* =
card-stratproof-weak-prop-rep-anon-papp l ∗ *n-voters parties committee-size r*
for *l n-voters parties committee-size r*
begin

sublocale *divide-voters-card-stratproof-weak-rep-anon-papp* ⟨*proof*⟩

sublocale *lowered*: *card-stratproof-weak-prop-rep-anon-papp*
n-voters parties committee-size r ∘ *lift-profile*
⟨*proof*⟩

end

4.3 Decreasing the number of parties

If we have a PAPP rule that satisfies weak representation and cardinality strategyproofness, for m parties, we can turn it into one for $m-1$ parties. This is done by simply duplicating one particular party (say x) in the preference profile, i.e. whenever x is part of an approval list, we add a clone of x (say y) as well. Should y then end up in the committee, we simply replace it with x.

Consequently, if we have an impossibility result for k parties, it also holds for $\geq m$ parties.

locale *remove-alt-card-stratproof-weak-rep-anon-papp* =
  card-stratproof-weak-rep-anon-papp n-voters parties committee-size r
  for *n-voters* **and** *parties* :: *'a set* **and** *committee-size r* +
  fixes *x y* :: *'a*
  assumes *xy*: *x* ∈ *parties y* ∈ *parties x* ≠ *y*
begin

definition *lift-applist* :: *'a set* ⇒ *'a set* **where**
  lift-applist X = (*if x* ∈ *X then insert y X else X*)

definition *lift-profile* :: *'a set multiset* ⇒ *'a set multiset* **where**
  lift-profile A = *image-mset lift-applist A*

definition *lower-result* **where** *lower-result C* = *image-mset* (λ*z*. *if z* = *y then x else z*) *C*

definition *lowered* **where** *lowered* = *lower-result* ∘ *r* ∘ *lift-profile*

lemma *lift-profile-empty* [*simp*]: *lift-profile* {#} = {#}
  ⟨*proof*⟩

lemma *lift-profile-add-mset* [*simp*]:
  lift-profile (*add-mset X A*) = *add-mset* (*lift-applist X*) (*lift-profile A*)
  ⟨*proof*⟩

lemma *empty-in-lift-profile-iff* [*simp*]: {} ∈# *lift-profile A* ⟷ {} ∈# *A*
  ⟨*proof*⟩

lemma *size-lift-profile* [*simp*]: *size* (*lift-profile A*) = *size A*
  ⟨*proof*⟩

lemma *lift-applist-eq-self-iff* [*simp*]: *lift-applist X* = *X* ⟷ *x* ∉ *X* ∨ *y* ∈ *X*
  ⟨*proof*⟩

lemma *lift-applist-eq-self-iff'* [*simp*]: *lift-applist* (*X* − {*y*}) = *X* ⟷ (*x* ∈ *X* ⟷ *y* ∈ *X*)
  ⟨*proof*⟩

lemma *in-lift-applist-iff*: *z* ∈ *lift-applist X* ⟷ *z* ∈ *X* ∨ (*z* = *y* ∧ *x* ∈ *X*)
  ⟨*proof*⟩

lemma *count-lift-profile*:
  assumes ∀ *Y*∈#*A*. *y* ∉ *Y*
  shows *count* (*lift-profile A*) *X* = (*if x* ∈ *X* ⟷ *y* ∈ *X then count A* (*X* − {*y*}) *else 0*)
  ⟨*proof*⟩

lemma *y-notin-lower-result* [*simp*]: *y* ∉# *lower-result C*
  ⟨*proof*⟩

lemma *lower-result-subset*: *set-mset* (*lower-result C*) ⊆ *insert x* (*set-mset C* − {*y*})
⟨*proof*⟩

lemma *lower-result-subset'*: *set-mset C* ⊆ *parties* ⟹ *set-mset* (*lower-result C*) ⊆ *parties*
⟨*proof*⟩

lemma *size-lower-result* [*simp*]: *size* (*lower-result C*) = *size C*
⟨*proof*⟩

lemma *count-lower-result*:
count (*lower-result C*) *z* =
(*if z* = *y then 0*
else if z = *x then count C x* + *count C y*
else count C z)
⟨*proof*⟩

lemma *in-lower-result-iff*:
z ∈# *lower-result C* ⟷ *z* ≠ *y* ∧ (*z* ∈# *C* ∨ (*z* = *x* ∧ *y* ∈# *C*))
⟨*proof*⟩

sublocale *lowered*: *anon-papp-election n-voters parties* − {*y*}
⟨*proof*⟩

lemma *is-pref-profile-lift-profile* [*intro*]:
assumes *lowered.is-pref-profile A*
shows *is-pref-profile* (*lift-profile A*)
⟨*proof*⟩

sublocale *lowered*: *anon-papp n-voters parties* − {*y*} *committee-size lowered*
⟨*proof*⟩

sublocale *lowered*: *weak-rep-anon-papp n-voters parties* − {*y*} *committee-size lowered*
⟨*proof*⟩

lemma *filter-lower-result-eq*:
y ∉ *X* ⟹ {#*x* ∈# *lower-result C*. *x* ∈ *X*#} = *lower-result* {#*x* ∈# *C*. *x* ∈ *lift-applist X*#}
⟨*proof*⟩

sublocale *lowered*: *card-stratproof-anon-papp n-voters parties* − {*y*} *committee-size lowered*
⟨*proof*⟩

sublocale *lowered*: *card-stratproof-weak-rep-anon-papp n-voters parties* − {*y*} *committee-size lowered*
⟨*proof*⟩

end

The following lemma is now simply an iterated application of the above. This allows us

to restrict a P-APP rule to any non-empty subset of parties.

lemma *card-stratproof-weak-rep-anon-papp-restrict-parties*:
assumes *card-stratproof-weak-rep-anon-papp n parties k r parties'* $\subseteq$ *parties parties'* $\neq \{\}$
shows $\exists r$. *card-stratproof-weak-rep-anon-papp n parties' k r*
⟨*proof*⟩

4.4 Decreasing the committee size

If we have a PAPP rule that satisfies weak representation and cardinality strategyproofness, for $l(k+1)$ voters, $m+1$ parties, and a committee size of $k+1$, we can turn it into one for lk voters, m parties, and a committee size of k.

This is done by again cloning a party x into a new party y and additionally adding l new voters whose preferences are $\{x, y\}$. We again replace any y occuring in the output committee with x. Weak representation then ensures that x occurs in the output at least once, and we then simply remove one x from the committee to obtain an output committee of size $k - 1$.

Consequently, if we have an impossibility result for a committee size of m, we can extend it to a larger committee size, but at the cost of introducing a new party and new voters, and with a restriction on the number of voters.

locale *decrease-committee-card-stratproof-weak-rep-anon-papp* =
card-stratproof-weak-rep-anon-papp l $*$ *(k + 1) insert y parties k + 1 r*
for *l k y* **and** *parties* :: *'a set* **and** *r* +
fixes *x* :: *'a*
assumes *xy*: *x* $\in$ *parties y* $\notin$ *parties*
assumes *k*: *k* > *0*
begin

definition *lift-applist* :: *'a set* $\Rightarrow$ *'a set* **where**
lift-applist X = (*if x* $\in$ *X then insert y X else X*)

definition *lift-profile* :: *'a set multiset* $\Rightarrow$ *'a set multiset* **where**
lift-profile A = *image-mset lift-applist A* + *replicate-mset l* {*x*, *y*}

definition *lower-result*
where *lower-result C* = *image-mset* (λ*z*. *if z* = *y then x else z*) *C* − {#*x*#}

definition *lowered* **where** *lowered* = *lower-result* ∘ *r* ∘ *lift-profile*

lemma *l*: *l* > *0*
⟨*proof*⟩

lemma *x-neq-y* [*simp*]: *x* $\neq$ *y y* $\neq$ *x*
⟨*proof*⟩

lemma *lift-profile-empty* [*simp*]: *lift-profile* {#} = *replicate-mset l* {*x*, *y*}
⟨*proof*⟩

lemma *lift-profile-add-mset* [*simp*]:
lift-profile (*add-mset X A*) = *add-mset* (*lift-applist X*) (*lift-profile A*)
⟨*proof*⟩

lemma *empty-in-lift-profile-iff* [*simp*]: {} ∈# *lift-profile A* ⟷ {} ∈# *A*
⟨*proof*⟩

lemma *size-lift-profile* [*simp*]: *size* (*lift-profile A*) = *size A* + *l*
⟨*proof*⟩

lemma *lift-applist-eq-self-iff* [*simp*]: *lift-applist X* = *X* ⟷ *x* ∉ *X* ∨ *y* ∈ *X*
⟨*proof*⟩

lemma *lift-applist-eq-self-iff'* [*simp*]: *lift-applist* (*X* − {*y*}) = *X* ⟷ (*x* ∈ *X* ⟷ *y* ∈ *X*)
⟨*proof*⟩

lemma *in-lift-applist-iff*: *z* ∈ *lift-applist X* ⟷ *z* ∈ *X* ∨ (*z* = *y* ∧ *x* ∈ *X*)
⟨*proof*⟩

lemma *count-lift-profile*:
assumes ∀ *Y*∈#*A*. *y* ∉ *Y*
shows *count* (*lift-profile A*) *X* =
(*if x* ∈ *X* ⟷ *y* ∈ *X then count A* (*X* − {*y*}) *else 0*) +
(*if X* = {*x*, *y*} *then l else 0*)
⟨*proof*⟩

lemma *y-notin-lower-result* [*simp*]: *y* ∉# *lower-result C*
⟨*proof*⟩

lemma *lower-result-subset*: *set-mset* (*lower-result C*) ⊆ *insert x* (*set-mset C* − {*y*})
⟨*proof*⟩

lemma *lower-result-subset'*: *set-mset C* ⊆ *insert y parties* ⟹ *set-mset* (*lower-result C*) ⊆ *parties*
⟨*proof*⟩

lemma *size-lower-result* [*simp*]:
assumes *x* ∈# *C* ∨ *y* ∈# *C*
shows *size* (*lower-result C*) = *size C* − *1*
⟨*proof*⟩

lemma *size-lower-result'*: *size* (*lower-result C*) = *size C* − (*if x* ∈# *C* ∨ *y* ∈# *C then 1 else 0*)
⟨*proof*⟩

lemma *count-lower-result*:

count (*lower-result C*) *z* =
(*if z* = *y then 0*
else if z = *x then count C x* + *count C y* − *1*
else count C z) (**is** - = *?rhs*)
⟨*proof*⟩

lemma *in-lower-resultD*:
z ∈# *lower-result C* ⟹ *z* = *x* ∨ *z* ∈# *C*
⟨*proof*⟩

lemma *in-lower-result-iff*:
z ∈# *lower-result C* ⟷ *z* ≠ *y* ∧ (*if z* = *x then count C x* + *count C y* > *1 else z* ∈# *C*)
(**is** - = *?rhs*)
⟨*proof*⟩

lemma *filter-lower-result-eq*:
assumes *y* ∉ *X*
shows {#*z* ∈# *lower-result C*. *z* ∈ *X*#} = *lower-result* {#*z* ∈# *C*. *z* ∈ *lift-applist X*#}
⟨*proof*⟩

sublocale *lowered*: *anon-papp-election l* ∗ *k parties k*
⟨*proof*⟩

lemma *is-pref-profile-lift-profile* [*intro*]:
assumes *lowered.is-pref-profile A*
shows *is-pref-profile* (*lift-profile A*)
⟨*proof*⟩

lemma *is-committee-lower-result*:
assumes *is-committee C x* ∈# *C* ∨ *y* ∈# *C*
shows *lowered.is-committee* (*lower-result C*)
⟨*proof*⟩

lemma *x-or-y-in-r-lift-profile*:
assumes *lowered.is-pref-profile A*
shows *x* ∈# *r* (*lift-profile A*) ∨ *y* ∈# *r* (*lift-profile A*)
⟨*proof*⟩

sublocale *lowered*: *anon-papp l* ∗ *k parties k lowered*
⟨*proof*⟩

sublocale *lowered*: *weak-rep-anon-papp l* ∗ *k parties k lowered*
⟨*proof*⟩

sublocale *lowered*: *card-stratproof-anon-papp l* ∗ *k parties k lowered*
⟨*proof*⟩

sublocale *lowered*: *card-stratproof-weak-rep-anon-papp l* ∗ *k parties k lowered*

⟨*proof*⟩

end

For Weak *Proportional* Representation, the lowering argument to decrease the committee size is somewhat easier since it does not involve adding a new party; instead, we simply add l new voters whose preferences are $\{x\}$.

locale *decrease-committee-card-stratproof-weak-prop-rep-anon-papp* =
card-stratproof-weak-prop-rep-anon-papp $l * (k + 1)$ *parties* $k + 1$ r
for l k **and** *parties* :: $'a$ *set* **and** r +
fixes x :: $'a$
assumes x: $x \in$ *parties*
assumes k: $k > 0$
begin

definition *lift-profile* :: $'a$ *set multiset* ⇒ $'a$ *set multiset* **where**
lift-profile $A = A +$ *replicate-mset* l $\{x\}$

definition *lower-result*
where *lower-result* $C = C - \{\#x\#\}$

definition *lowered* **where** *lowered* = *lower-result* ∘ r ∘ *lift-profile*

lemma l: $l > 0$
⟨*proof*⟩

lemma *lift-profile-empty* [*simp*]: *lift-profile* $\{\#\}$ = *replicate-mset* l $\{x\}$
⟨*proof*⟩

lemma *lift-profile-add-mset* [*simp*]:
lift-profile (*add-mset* X A) = *add-mset* X (*lift-profile* A)
⟨*proof*⟩

lemma *empty-in-lift-profile-iff* [*simp*]: $\{\} \in\#$ *lift-profile* $A \longleftrightarrow \{\} \in\#$ A
⟨*proof*⟩

lemma *size-lift-profile* [*simp*]: *size* (*lift-profile* A) = *size* $A + l$
⟨*proof*⟩

lemma *count-lift-profile*:
count (*lift-profile* A) X = *count* A X + (*if* $X = \{x\}$ *then* l *else* 0)
⟨*proof*⟩

lemma *size-lower-result* [*simp*]:
assumes $x \in\#$ C
shows *size* (*lower-result* C) = *size* $C - 1$
⟨*proof*⟩

lemma *size-lower-result'*: *size* (*lower-result C*) = *size C* − (*if x* ∈# *C then 1 else 0*)
⟨*proof*⟩

lemma *count-lower-result*:
count (*lower-result C*) *z* = *count C z* − (*if z* = *x then 1 else 0*)
⟨*proof*⟩

lemma *in-lower-resultD*:
z ∈# *lower-result C* ⟹ *z* ∈# *C*
⟨*proof*⟩

lemma *in-lower-result-iff*:
z ∈# *lower-result C* ⟷ (*if z* = *x then count C x* > *1 else z* ∈# *C*)
(**is** - = *?rhs*)
⟨*proof*⟩

sublocale *lowered*: *anon-papp-election l* ∗ *k parties k*
⟨*proof*⟩

lemma *is-pref-profile-lift-profile* [*intro*]:
assumes *lowered.is-pref-profile A*
shows *is-pref-profile* (*lift-profile A*)
⟨*proof*⟩

lemma *is-committee-lower-result*:
assumes *is-committee C x* ∈# *C*
shows *lowered.is-committee* (*lower-result C*)
⟨*proof*⟩

lemma *x-in-r-lift-profile*:
assumes *lowered.is-pref-profile A*
shows *x* ∈# *r* (*lift-profile A*)
⟨*proof*⟩

sublocale *lowered*: *anon-papp l* ∗ *k parties k lowered*
⟨*proof*⟩

sublocale *lowered*: *weak-prop-rep-anon-papp l* ∗ *k parties k lowered*
⟨*proof*⟩

sublocale *lowered*: *card-stratproof-anon-papp l* ∗ *k parties k lowered*
⟨*proof*⟩

sublocale *lowered*: *card-stratproof-weak-prop-rep-anon-papp l* ∗ *k parties k lowered*
⟨*proof*⟩

end

end

5 Lifting the Impossibility Result to Larger Settings

theory *PAPP-Impossibility*
imports *PAPP-Impossibility-Base-Case Anonymous-PAPP-Lowering*
begin

In this section, we now prove the main results of this work by combining the base case with the lifting arguments formalized earlier.

First, we prove the following very simple technical lemma: a set that is infinite or finite with cardinality at least 2 contains two different elements x and y.

lemma *obtain-2-elements*:
assumes *infinite* $X \vee$ *card* $X \geq 2$
obtains x y **where** $x \in X$ $y \in X$ $x \neq y$
⟨*proof*⟩

We now have all the ingredients to formalise the first main impossibility result: There is no P-APP rule that satisfies Anonymity, Cardinality-Strategyproofness, and Weak Representation if $k \geq 3$ and $m \geq k + 1$ and n is a multiple of $2k$.

The proof simply uses the lowering lemmas we proved earlier to first reduce the committee size to 3, then reduce the voters to 6, and finally restrict the parties to 4. At that point, the base case we proved with SAT solving earlier kicks in.

This corresponds to Theorem 1 in the paper.

theorem *papp-impossibility1*:
assumes $k \geq 3$ **and** *card parties* $\geq k + 1$ **and** *finite parties*
shows ¬*card-stratproof-weak-rep-anon-papp* $(2 * k * l)$ *parties k r*
⟨*proof*⟩

If Weak Representation is replaced with Weak Proportional Representation, we can strengthen the impossibility result by relaxing the conditions on the number of parties to $m \geq 4$.

This works because with Weak Proportional Representation, we can reduce the size of the committee without changing the number of parties. We use this to again bring k down to 3 without changing m, at which point we can simply apply our previous impossibility result for Weak Representation.

This corresponds to Theorem 2 in the paper.

corollary *papp-impossibility2*:
assumes $k \geq 3$ **and** *card parties* ≥ 4 **and** *finite parties*
shows ¬*card-stratproof-weak-prop-rep-anon-papp* $(2 * k * l)$ *parties k r*
⟨*proof*⟩

end

References

[1] P. Lammich. The GRAT tool chain – efficient (UN)SAT certificate checking with formal correctness guarantees. In S. Gaspers and T. Walsh, editors, *Theory and Applications of Satisfiability Testing – SAT 2017, Proceedings*, volume 10491 of *Lecture Notes in Computer Science*, pages 457–463. Springer, 2017.

[2] N. Wetzler, M. Heule, and W. A. H. Jr. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In C. Sinz and U. Egly, editors, *Theory and Applications of Satisfiability Testing – SAT 2014, Proceedings*, volume 8561 of *Lecture Notes in Computer Science*, pages 422–429. Springer, 2014.