Verifying Randomised Social Choice

Manuel Eberl[0000−0002−4263−6571]

Technische Universität München, 85748 Garching bei München, Germany

Abstract This work describes the formalisation of a recent result from Randomised Social Choice Theory in Isabelle/HOL. The original result had been obtained through the use of linear programming, an unverified Java program, and SMT solvers; at the time that the formalisation effort began, no human-readable proof was available. Thus, the formalisation with Isabelle eventually served as both independent rigorous confirmation of the original result and led to human-readable proofs both in Isabelle and on paper.
This presentation focuses on the process of the formalisation itself, the domain-specific tooling that was developed for it in Isabelle, and how the structured human-readable proof was constructed from the SMT proof. It also briefly discusses how the formalisation uncovered a serious flaw in a second peer-reviewed publication.

1 Introduction

First of all, it must be stressed that this presentation is not intended as an introduction to Social Choice Theory, nor will it repeat the detailed explanation of the proof of the main result in Brandl *et al.* [1] (of which I am also a co-author). I must also stress that my contribution consists *only* of the formalisation and the human-readable proof for that result, and the purpose of this paper is to present more details of this formalisation process and the technology behind it.

All the background theory of Social Choice Theory that I will mention later on is either folklore or comes from the work of Brandl *et al.*; again, see their presentation [1] for more details on this background. For the sake of self-containedness, the result and the necessary definitions from Social Choice Theory will nevertheless be sketched here very briefly, but without any deeper explanation or motivation. For this, the reader should consult the original presentation by Brandl *et al.*

I will attempt to strike a balance between readability and technical details. In particular, I attempt to stay close to the Isabelle definitions, but mostly without actually using Isabelle syntax except in cases where there is real benefit in doing so. The full formal Isabelle proof developments can be found in the *Archive of Formal Proofs* [2,3].

2 The Main Result

The main result that was formalised is a typical impossibility result from Social Choice Theory: these are of the form

'There exists no voting scheme for at least m voters and n alternatives that simultaneously has the following properties:'

A large variety of results like this exists for many different types of voting schemes and many different choices for the properties that they should have; two famous ones are Arrow's Theorem [4] and Gibbard's Theorem [5].

The setting that we shall focus on is that of *Social Decision Schemes* (SDSs): We fix some finite set $N = \{1, \ldots, m\}$ of *agents* (or *voters*) and a finite set of alternatives A with $|A| = n$. Each agent i has a *preference relation* $\succeq_i$ over the alternatives. In our setting, these preferences are total preorders, i. e. reflexive, transitive, and total relations. The vector $R = (\succeq_1, \ldots, \succeq_m)$ is called a *preference profile*. An SDS is then a function that, given such a preference profile, returns a *lottery*: a probability distribution of winning alternatives.

The theorem that was formalised is the following:

Theorem 1. *If $m \geq 4$ and $n \geq 4$, there exists no SDS that has the following properties:*[1]

Anonymity: *Invariance under renaming of agents*

Neutrality: *Invariance under renaming of alternatives*

SD-**Efficiency:** *If the SDS returns some lottery, it is optimal in the sense that there is no other lottery that all agents prefer to it.*

SD-**Strategyproofness:** *No one agent can, by themselves, obtain a better result by lying about their preferences (i. e. strategic voting is not possible).*

As we will see later, it is enough to prove the theorem for $m = n = 4$ because any SDS $f(R)$ with the above properties for $m \geq 4$ agents and $n \geq 4$ alternatives would give rise to another SDS (denoted as $f{\downarrow}(R)$) with the same properties for exactly 4 agents and alternatives (see Section 3 for details on this). The difficult part is therefore the case of exactly 4 agents and alternatives.

In a nutshell, the strategy Brandl *et al.* pursued to find a proof for this case was the following: Consider the set of all preference profiles for our 4 agents and alternatives. For each profile R (or pair of profiles R_1, R_2), there are certain conditions on the probabilities of the lottery $f(R)$ (resp. the lotteries $f(R_1)$ and $f(R_2)$) resulting from the four conditions (anonymity, neutrality, etc.) It happens that all these conditions can be written as formulæ in quantifier-free linear real arithmetic (QF-LRA), which is a decidable logic that SMT solvers can typically handle fairly efficiently.

Unfortunately, there are 31,640,625 profiles for $m = n = 4$ (or 471,956 modulo anonymity and neutrality), which results in far too many QF-LRA formulæ to compute and check. However, if there really is no SDS with these four properties, these conditions must be inconsistent. When this is the case, there is often a much smaller subset of conditions (or, equivalently, a smaller set of profiles) that is already inconsistent – an *unsatisfiable core*. If one could guess a small set of profiles that already leads to inconsistent conditions, one could pass only these to an SMT solver and obtain a proof of the contradiction more quickly.

[1] The meaning of these concepts will be made more precise later – in particular, what it *means* for an agent to prefer one lottery over another.

Brandl *et al.* used search heuristics to find such a set of profiles, which they then narrowed down to only 47. The search for profile sets likely to lead to a contradiction and the translation to QF-LRA formulæ in the SMT-LIB format was performed by an unverified Java program.

However, there are various problems with this. Computer proofs are notoriously controversial in the mathematical community and even in high-profile computer proofs such as the Kepler conjecture [6] or the Lorenz Attractor [7,8], problems with the computer code were later found (although they turned out to be repairable). In our case, some possible points of criticism are:

– One must trust the SMT solver.
– One must trust the Java program that computes the conditions arising from the profiles and encodes them into the SMT-LIB format.
– The proof cannot realistically be inspected or verified by a human.

The first problem is not too serious, since one can use several different independent SMT solvers to check the result. Some of them can produce proof objects that can be verified by simple independent checkers.

The second problem could be solved by inspecting the generated SMT-LIB file by hand and checking that the generated inequalities are indeed the ones that follow from the 47 preference profiles – a very tedious task, but feasible.

The last problem, however, is difficult to address. While solvers like Z3 can print out unsatisfiability proofs, these proofs are typically fairly large and dense and provide little insight.

To address these concerns, Brandl *et al.* sought out my collaboration to formalise some version of this proof in Isabelle. Since Isabelle can replay SMT proof objects through its own kernel, we were confident that it should be possible to obtain *some* kind of proof of Theorem 1 in Isabelle. It was, however, completely unclear how to achieve the ultimate goal of finding a more structured and human-readable proof and whether such a proof even exists.

3 Definitions

First of all, I will define the basic notions that are required to state and prove the main result. For the Isabelle definitions, I followed the philosophy to keep definitions as simple and as close to the textbook definitions as possible – including syntax – or to at least prove more textbook-style versions as alternative definitions later on. In particular, I also placed great importance on proving various different views on more complex notions (e. g. Stochastic Dominance, *SD*-Efficiency, and Strategyproofness). This makes working with them more convenient as one can pick whichever form is most suitable in any given situation; it also increases the confidence that the definitions really *are* faithful to the textbook definitions.

I aim to present every notion precisely as it is defined in Isabelle, but for the sake of brevity and readability, I will mostly refrain from using actual Isabelle notation.

Most of the notions discussed here (such as *family of preorders*, *Social Decision Scheme*, *SD-efficient SDS*) are defined as *locales* [9]. These are a kind of named context supporting multiple inheritance that facilitates modular reasoning.

In the remainder of this section, I will list the most relevant definitions for Theorem 1.

Agents and Alternatives. In the Isabelle formalisation, agents and alternatives are opaque: We simply assume that we have finite non-empty sets N of some type ν *set* (agents) and A of some type γ *set* (alternatives).[2] For convenience, I invented the name *election* (with a locale of the same name) to describe a setting with a fixed set of agents and alternatives.

Preferences. Each agent i has their own *preference relation* $\succeq$, which is a *total preorder* (reflexive, transitive, and total – i. e. $x \succeq y \vee y \succeq x$ for all x, y). The collection of the preferences of all the agents forms a vector $(\succeq_1, \ldots, \succeq_m)$; this is called a *preference profile*.

In Isabelle, the preference relations are modelled as functions $\gamma \to \gamma \to \text{bool}$. Preference profiles are a modelled as functions $\nu \to \gamma \to \gamma \to \text{bool}$. By convention, any preference relation must return *false* if one of its inputs is not in A and any preference profile must return an empty relation if its input is not in N. This ensures extensionality in the sense that e. g. two preference relations are logically equal if they agree on all alternatives in A.

Lotteries. A *lottery* is a probability distribution of alternatives. Since there are only finitely many alternatives, the most convenient representation of this is as a *Probability Mass Function* (PMF). In Isabelle/HOL, the type γ *pmf* is defined as the type of all functions $f : \gamma \to \mathbb{R}$ such that $\forall x.\ f(x) \geq 0$ and $\sum_x f(x) = 1$. Various probability- and measure-theoretic concepts are defined for this type so that one can work with it in a fairly idiomatic way. We can simply define the set of *lotteries* as the set of values of type γ *pmf* whose support is a subset of A.

Anonymity and Neutrality. An SDS is *anonymous* if renaming the agents does not change the outcome, i. e. for any permutation π of N, we have $f(R \circ \pi) = f(R)$.

For neutrality, we first need to define what it means to rename an alternative. Let σ be a permutation of A. Then, if $\succeq$ is a preference relation, the relation $\succeq^\sigma$ obtained by renaming the alternatives with σ can be defined as $x \succeq^\sigma y \longleftrightarrow \sigma^{-1}(x) \succeq \sigma^{-1}(y)$. By renaming all preferences in a profile R like this, we obtain a profile R^σ. Additionally, we also have to take into account that the elements in the lottery returned by f must also be renamed. This can be accomplished with the function *map_pmf*, which is the *push-forward measure*, i. e. the covariant map function for PMFs. All in all, we obtain the condition

$$f(R^\sigma) = \text{map\_pmf}\ \sigma\ (f(R))\ .$$

[2] Readers who are used to systems like Coq or Lean might wonder why one does not simply use the entire types ν and γ. The reason for this is that we sometimes want to decrease or increase the number of agents and alternatives. Doing this without explicit carrier sets can be problematic in Isabelle.

Pareto preorder. A family (i. e. a vector) $R = (\succeq_1, \ldots, \succeq_n)$ of preorders gives rise to the *Pareto preorder* of that family, which is simply defined as the intersection of all the $\succeq_i$:

$$x \preceq_{\mathrm{Par}(R)} y \Longleftrightarrow \forall i.\ x \preceq_i y$$

Note that here, the $\succeq_i$ are not assumed to be total, since we will use Par for non-total relations later on. Even if the $\succeq$ *are* total, $\mathrm{Par}(R)$ typically is not.

We call x a *Pareto loser* w. r. t. R if there is y such that $x \prec_{\mathrm{Par}(R)} y$; in other words, there exists another alternative y that makes all agents at least as happy as x, and one of them strictly more happy.

Stochastic dominance. As was mentioned before, we need a notion of when an agent prefers one lottery to another, i. e. to lift a preference relation $\succeq$ on alternatives to one on lotteries. Such a notion is called a *lottery extension.* In general, the resulting relations on lotteries are *not* total. Lottery extensions are typically justified by making some reasonable assumption about the behaviour of agents and then concluding under what circumstances they *must* prefer one lottery over another. An extreme example to illustrate this would be that any agent should be expected to prefer the singleton lottery where their favourite alternative wins with probability 1 over any other lottery.

The lottery extension we shall use is *Stochastic Dominance.* The definition is somewhat technical, namely

$$p \preceq_{SD(\succeq)} q \Longleftrightarrow \forall x{\in}A.\ \mathrm{P}_p[\{y \mid y \succeq x\}] \leq \mathrm{P}_q[\{y \mid y \succeq x\}]\ ,$$

i. e. for any alternative x, the probability of getting something at least as good as x in the lottery q is at least as big as that in p.

Another equivalent, but perhaps more intuitive definition is

$$p \preceq_{SD(\succeq)} q \Longleftrightarrow \forall u{\in}\mathrm{vnM}(\succeq).\ \mathrm{E}_p[u] \leq \mathrm{E}_q[u],$$

i. e. for all *von Neumann–Morgenstern utility functions* $u : A \to \mathbb{R}$ that are compatible with the preference relation $\succeq$, the lottery q must yield at least as much expected utility as p.

The idea behind *SD* is that agents are assumed to have a utility function and want to maximise their expected utility. We only know the agent's preference relation, but not the underlying utility function. However, if a lottery q yields at least as much expected utility as p for *all* utility functions that fit the agent's preferences, we expect the agent to consider q at least as good as p – and that is precisely what *SD* is.

Efficiency. Efficiency of an SDS, in general, means that the SDS never returns a lottery that can be improved upon in a way that satisfies all agents. One basic notion of Efficiency is *ex-post*-Efficiency, which states that for any profile R, the resulting lottery $f(R)$ must not contain a Pareto loser w. r. t. R.

SD-Efficiency, which is used in Theorem 1, is a stronger notion. Using the concepts defined above, we can define it very concisely: a lottery is called *SD-inefficient* w. r. t. a profile R if it is a Pareto loser w. r. t. $SD \circ R$, i. e. there is another profile R' that is weakly *SD*-preferred by all agents and strictly *SD*-preferred by at least one agent.

An SDS is called *SD-efficient* if it never returns an *SD*-inefficient lottery.

Strategyproofness. Strategyproofness captures the intuitive idea that agents should have no benefit from voting strategically. There are various notions of Strategyproofness; for our purposes, we only need (weak) *SD*-Strategyproofness. An SDS f is called *(weakly) SD-strategyproof* if, for any agent $i \in N$, any preference profile $R = (\succeq_1, \ldots, \succeq_m)$, and any preference relation $\succeq_i'$ we have:

$$f(R(i := \succeq_i')) \not\succ_{SD(\succeq_i)} f(R)$$

Intuitively, this means that no single agent can benefit from lying about their preferences. If they submit false preferences $\succeq_i'$ instead of their true preferences $\succeq_i$ (while all other preferences remain the same), the result can never be better (w. r. t. $SD(\succeq_i)$) than if they had submitted their true preferences.

Lifting. As was mentioned before, the impossibility result can be 'lifted' from m agents and n alternatives to m' agents and n' alternatives with $m' \geq m$, $n' \geq n$. The general idea is this: Given a preference profile R for m agents and n alternatives, we can extend this profile to n' alternatives by adding $n' - n$ new 'dummy' alternatives that are all equally good, but strictly worse than all the existing n alternatives. Then, we can extend the profile to m' agents by adding $m' - m$ new 'dummy' agents that are fully indifferent between all n' alternatives. We denote this 'lifted' version of R as $R{\uparrow}$.

Using this, we can 'lower' any SDS f for m' agents and n' alternatives to an SDS $f{\downarrow}$ on m agents and n alternatives by defining $f{\downarrow}(R) := f(R{\uparrow})$. In order for $f{\downarrow}$ to be well-defined, however, it must never return any of the dummy alternatives. Since the dummy alternatives are all Pareto losers, one way to ensure this is if f is *ex-post*-efficient. In this case, $f{\downarrow}$ is also *ex-post*-efficient.

Notably, if f is *ex-post*-efficient, this construction also preserves anonymity, neutrality, *SD*-Efficiency, and *SD*-Strategyproofness. This way, any impossibility result involving *ex-post*-Efficiency (or something stronger) and any combination of the above-mentioned properties can be lifted to a higher number of agents and alternatives.

Random (Serial) Dictatorship. Let us now turn to two interesting examples of concrete SDSs that I also formalised in Isabelle as an exercise to myself and to the library I developed: *Random Dictatorship* (*RD*) and *Random Serial Dictatorship* (*RSD*).

The former is normally only defined for the subset of preference profiles where each agent has a unique favourite alternative. In this case, *RD* picks an agent uniformly at random and returns that agent's favourite alternative as the winner. Since the present formalisation only allows SDSs over the *full* set of preference relations, I chose the obvious generalisation where one first picks an agent uniformly at random and then returns one of that agent's favourites uniformly at random if there are more than one. The Isabelle definition of *RD* is

$$\text{RD}\ R = \textbf{do}\ \{i \leftarrow \text{pmf\_of\_set}\ N;\ \text{pmf\_of\_set}\ (\text{Max\_wrt\_among}\ (R\ i)\ A)\}$$

where *pmf_of_set* X describes the uniform distribution over the set X and *Max_wrt_among* returns the maximal elements among the given set w. r. t. the given preference relation. For details on the monadic **do** notation, see e. g. Eberl *et al.* [10]. The SDS thus defined can then easily be proven to satisfy anonymity, neutrality, and *SD*-Strategyproofness (a stronger version of the latter than the one defined above even). It is, however, not *ex-post*-efficient.

Random Serial Dictatorship is another generalisation of RD to the full set of preference profiles which additionally fulfils *ex-post*-Efficiency. Here, one first chooses a random permutation $i_1, \ldots, i_m$ of the agents and then lets each agent, in that order, delete all those among the remaining alternatives that they 'do not want' (i. e. keep only those that they prefer most among the remaining ones). Then, one returns one of the remaining alternatives (among which all agents are indifferent) uniformly at random. One possible Isabelle definition is

$$\begin{aligned}\text{RSD}\ N\ A\ R = \textbf{do}\ & is \leftarrow \text{pmf\_of\_set}\ (\text{permutations\_of\_set}\ N)\\ & \text{pmf\_of\_set}\ (\text{foldr}\ (\lambda i\ A'.\ \text{Max\_wrt\_among}\ (R\ i)\ A')\ is\ A)\end{aligned}$$

where *permutations_of_set* N returns the set of all lists that contain each element of N exactly once. An alternative recursive definition is

$$\begin{aligned}\text{RSD}\ N\ A\ R = \textbf{if}\ N = \emptyset\ & \textbf{then}\ \text{pmf\_of\_set}\ A\\ & \textbf{else}\ \ \textbf{do}\ i \leftarrow \text{pmf\_of\_set}\ N\\ & \qquad\qquad \text{RSD}\ (N \setminus \{i\})\ (\text{Max\_wrt\_among}\ (R\ i)\ A)\ R\end{aligned}$$

The actual definition in Isabelle uses the generic combinator *fold_random_permutation* from the Isabelle library that allows traversing a set in random order. This directly yields the above two definitions as corollaries and allows the user to use whichever form is more convenient.

RSD can then be proven to be anonymous, neutral, strongly *SD*-strategyproof, and *ex-post*-efficient. The proofs of the first two are fairly simple; the other two are somewhat more involved. Writing these non-trivial proofs about a *concrete* SDS like *RSD* served as a good first 'stress test' of the Social Choice library and increased the confidence that the formal definitions were as intended. This is likely the first formalisation of *RD* and *RSD* and their properties.

4 Gathering Consequences from Profiles

Having established the necessary definitions, we can now approach the proof of the main result (Theorem 1). First of all, let us explore how to take the four conditions – anonymity, neutrality, *SD*-Efficiency, *SD*-Strategyproofness – and derive all of the arising conditions for a fixed set of profiles, from which we can then hopefully derive a contradiction. Suppose we have an SDS f for some fixed set of m agents and n alternatives. As mentioned before, these four conditions can be fully characterised by QF-LRA formulæ. The variables in these formulæ are the probabilities returned by f for each profile R. We denote these variables as $p_{R,x}$ (the probability that $f(R)$ returns the winner x).

Let us now go through the different types of conditions. Again, I will only sketch the precise constructions here; for more details, see Brandl *et al.* [1].

Anonymity and Neutrality. Anonymity can be handled implicitly by simply considering all preference profiles that differ only by a renaming of agents as equal. An alternative view is to look at a preference profile as a *multiset* of preference relations instead of as a *vector*.

For neutrality, one can similarly consider all profiles equal that differ only by a renaming of alternatives. Here, the only way to implement this in practice seems to be to choose an arbitrary representative among the $n!$ candidates, e. g. the one with the lexicographically smallest representation.

Orbit conditions. The above does not completely capture anonymity and neutrality; what is still missing are the so-called *orbit conditions* that arise from profile automorphisms: If a permutation σ of alternatives maps a profile R to itself (modulo anonymity), it is clear that by neutrality, all alternatives in an orbit of σ must receive the same probability (e. g. if $\sigma = (a\ b\ c)(d)$, we have $p_{R,a} = p_{R,b} = p_{R,c}$). These orbit conditions tend to arise when R has rich symmetries. Together with the efficiency conditions, they will be extremely useful in the proof later since they greatly restrict the possible values for f.

SD-Strategyproofness. This is easy to capture in QF-LRA: For any pair of profiles R, $\bar{R}$ we must check if $\bar{R}$ differs from R only by the preferences of one agent i. If that is the case, let $\succeq_i$ resp. $\bar{\succeq}_i$ denote the preference relation of agent i in R resp. $\bar{R}$. We must then have $\neg f(\bar{R}) \succ_{SD(\succeq_i)} f(R)$ and $\neg f(R) \succ_{SD(\bar{\succeq}_i)} f(\bar{R})$. When the definition of Stochastic Dominance is unfolded, these conditions simply reduce to a combination of equations and inequalities in the $p_{R,x}$ and $p_{\bar{R},x}$.

Of course, equality must be seen modulo anonymity and neutrality here, and if a renaming of alternatives was necessary for the manipulation, this renaming must also be taken into account.

SD-Efficiency. This is the most difficult condition to handle. Here, the key insight by Brandl *et al.* is that if a lottery is *SD*-efficient w. r. t. a profile R, then all other lotteries with the same support or a smaller support (w. r. t. inclusion) are *also SD*-efficient. We can therefore define the notion of an *SD*-efficient support: A set $X \subseteq A$ is called an *SD-efficient support* if the lotteries that have support X are *SD*-efficient. Whether such a set X is an *SD*-efficient support can simply be encoded as a linear programming problem.

Therefore, we only need to find all the inclusion-minimal *SD*-inefficient supports $X_1, X_2, \dots$ (of which there are $< 2^m$). The condition that some lottery p be *SD*-efficient w. r. t. R then reduces to its support not being a superset of any of these minimal *SD*-inefficient supports, i. e. $\forall k.\ \exists x {\in} X_k.\ p_{R,x} = 0$. This is a conjunction of disjunctions, and thus a QF-LRA formula. Of course, this support-set characterisation of *SD*-Efficiency is also fully verified in the system.

Another interesting fact is that a singleton support $\{x\}$ is *SD*-inefficient iff x is a Pareto loser. This directly implies that *SD*-Efficiency is stronger than *ex-post*-Efficiency, and it means that we do not have to employ linear programming for singleton sets; we can simply check if the element is a Pareto loser.

Lottery conditions. Lastly, we still need to take into account that the $p_{R,x}$ are not independent real variables: since they represent probabilities, they are subject to the conditions $p_{R,x} \geq 0$ and $\sum_{x \in A} p_{R,x} = 1$.

5 Tooling

5.1 External Tools and Trusted Code Base

I will now give a brief overview of the two external tools that were used in this project. Neither of them are trusted, i. e. the correctness of the final result does not depend on them. First, however, I would like to clarify what exactly the trusted code base of the result is.

Isabelle is based on a simple intuitionistic logic known as Isabelle/Pure, on top of which the *object logic* HOL is then axiomatised. The basic inference rules of *Pure* are provided as ML functions by the Isabelle kernel, which is the only part of Isabelle that can actually produce theorems[3]. All other parts of Isabelle (e. g. all of its various proof automation tools) can only prove theorems by interfacing with this kernel, so that the trusted code base is effectively only the kernel (and the code for parsing and pretty-printing). A bug in any other part of Isabelle or in my own ML code should therefore, in principle, never lead to an inconsistency. To reiterate: all proofs in this work go through the kernel. There is no use of computational reflection, there are no no external computations, and no trusted external tools.

Now, let me clarify what the two external tools *were* used for and in what form.

Z3. This is a well-known SMT solver. It is bundled with the Isabelle distribution and integrated via the *smt* proof method [11], which translates Isabelle/HOL goals into the SMT-LIB format, calls Z3, and attempts to reconstruct an Isabelle proof from the Z3 proof. Here, *Isabelle proof* does not mean Isabelle proof text. *smt* does not produce Isabelle code; it rather constructs Isabelle theorems by emulating the Z3 proof rules with basic logical inference. A replayed Z3 proof therefore appears as a single opaque invocation of the *smt* method in Isabelle proof text. Like the Z3 proofs, these reconstructed proofs are very large and low-level and therefore not human-readable. They are, however, just as trustworthy as any other Isabelle proof since the *smt* method has to go through Isabelle's kernel in order to create theorems.

[3] Except for *oracles*, which I do not use here.

As will be explained in Section 6, this *smt* method was very helpful in deriving the 'human-readable' version of the proof of Theorem 1; however the final proof does not contain any invocations of it anymore.

QSOpt_ex. This software is a Linear Programming solver written in C that uses exact rational arithmetic [12,13], i. e. it outputs the exact optimal solutions as rational numbers without any rounding errors. It was developed by Applegate *et al.* using their non-exact solver QSopt as a basis and uses a combination of fast, non-exact floating point operations and exact rational computations based on GMP arbitrary-precision rational numbers. This is important because we want to use the solution returned by the solver to construct witness lotteries, and even a small rounding error would lead Isabelle to reject such a witness.

However, I do not use this version of QSopt_ex since I was unable to compile the code. Fortunately, there is a fork by Jon Lund Steffenson [14] that provides a number of improvements, particularly to the build system. I created rudimentary bindings to interface with this version of QSopt_ex from Isabelle/ML by writing the problems into a problem description file in the LP format, invoking QSopt_ex on it, and parsing the result file.

For our purposes, we only need to *compute* the optimal solutions, but we do not have to *prove* that they are optimal. QSopt_ex is used to check if a support is *SD*-inefficient and – when it is – to compute a witness for this inefficiency (i. e. another lottery that is strictly better w. r. t. Pareto-*SD*). If there were a bug in QSopt_ex, this would either lead to an unprovable goal when trying to use the witness or it would cause us to miss some inefficient supports and therefore give us less information about the consequences of *SD*-Efficiency. It can, by construction, never lead to any inconsistency.

Note that we do not need to show the optimality of the solutions found by QSopt_ex in Isabelle; it is only required on a meta level for the completeness of the approach. We do need to prove the correctness of the solutions, however, and this can easily be done by Isabelle's general-purpose automation.

5.2 Automation in Isabelle/HOL

While all of the many facts following from our four properties for the given set of preference profiles could easily be derived and proven in Isabelle by a human, this would have resulted in a considerable amount of work and boilerplate proofs. Moreover, this work would have to be re-done for a new proof of a related statement or even if the underlying preference profiles changed slightly, which discourages experimentation. The goal was therefore to develop specialised automation for this in Isabelle that is capable of replacing the unverified Java program by Brandl *et al.*, thereby turning Isabelle into a capable IDE for randomised Social Choice proofs of this kind.

Isabelle itself is written in Standard ML and contains a sophisticated ML system based on Poly/ML that allows compiling and adding new code at run-time. Users can add custom proof methods written in ML to automate proof steps and commands to automatically define constants, derive facts, etc. I developed a number of such Isabelle commands to automate the fact gathering described before:

preference_profiles defines preference profiles and automatically proves their well-definedness. The notation is similar to that found in textbooks: to specify e. g. the preference relation $1 \succ \{2, 3\} \succ 4$ (1 is better than 2 or 3 and 2 and 3 are equally good and better than 4) one would write $1, [2, 3], 4$.

derive_orbit_equations computes the orbit conditions for a set of given preference profiles and proves them automatically. . For each orbit, a canonical representative x is chosen and the orbit conditions have the form $f(R)(y) = f(R)(x)$, where $y \neq x$ is some other element on the orbit. This makes it possible to use the orbit conditions directly as rewrite rules for Isabelle's simplifier, since the equations are normalising.

find_inefficient_supports computes Pareto losers and *SD*-inefficient supports and automatically proves the corresponding conditions for *ex-post*- and *SD*-efficient SDSs. In order to find *SD*-inefficient supports and prove their inefficiency, the ML code invokes the external Linear Programming solver *QSOpt_ex*.

prove_inefficient_supports takes a list of *ex-post*- and *SD*-inefficient supports where each *SD*-inefficient support is annotated with a witness lottery (i. e. a lottery that is strictly *SD*-preferred to the uniform distribution on the inefficient support). This witness lottery can be read directly from the solution of the corresponding linear program.

The idea is to compute the inefficient supports and their witnesses once with *find_inefficient_supports*, which outputs a hyperlink that can be clicked to automatically insert a corresponding invocation of *prove_inefficient_supports* with all the witnesses filled in as needed. This makes the final proof document completely independent from the external LP solver.

derive_strategyproofness_conditions takes a list of preference profiles and computes all possible manipulations of all profiles in this list that yield another profile in the list. It then derives and proves all the conditions that arise from these manipulations for a (weakly) strategyproof SDS. The user can specify an optional distance threshold to restrict the search to small manipulations (measured as the cardinality of the symmetric difference of the relations). For our purposes, a distance of 2 is sufficient.

Note again that this ML code is *untrusted*: I did not verify it and – as explained in Section 5.1 – there is, in fact, no need to verify it.

All of this automation is available in the *Archive of Formal Proofs* entry on randomised Social Choice [2]. The automation also provides ML interfaces so that for future similar projects, one could easily implement the entire pipeline of candidate set generation, derivation of all the QF-LRA conditions, and the invocation of the SMT solver inside Isabelle, turning it into a convenient and extensible IDE for randomised Social Choice.

6 The Formal Proof of Theorem 1

The formal proof of the main result begins with a definition of the setting: I define a locale called *sds_ impossibility* for the setting of an anonymous, neutral, *SD*-efficient and *SD*-strategyproof SDS for $m \geq 4$ agents I and $n \geq 4$ alternatives N. Building on this, I then define a sublocale called *sds_ impossibility_ 4_ 4* that additionally assumes that $I = \{A, B, C, D\}$ and $N = \{a, b, c, d\}$ where the four agents and alternatives are distinct. Our goal is to prove *False* in the context of the latter locale and then use the lifting machinery described in Section 3 to derive *False* in the first locale.

For illustration purposes, I will track the total number of *degrees of freedom* in our problem, i. e. the number of real variables $p_{R,x}$ that are not constrained by an equation. In the beginning, we have 141 degrees of freedom (4 for each profile, minus 1 eliminated since the probabilities must sum to 1).

The automatic part. In the context of the locale *sds_ impossibility_ 4_ 4*, the machinery described in Section 5.2 is invoked: The 47 preference profiles listed in the proof by Brandl *et al.* are defined using the *preference_ profiles* command. The orbit and strategyproofness conditions are derived fully automatically – we only have to supply the list of profiles that we are interested in to the corresponding commands. For the efficiency conditions, we need to run *find_ inefficient_ supports* once; for the full set of profiles, this takes about 7 s. The final proof document only contains the invocation of *prove_ inefficient_ supports* generated by it.

This automatic part is fairly quick: The proofs of the well-definedness of the profiles and all the other conditions take about 20 s altogether. The result returned by these commands is:

- 12 equations of the form $p_{R,x} = p_{R,y}$ from orbit conditions
- 24 equations of the form $p_{R,x} = 0$ from Pareto losers
- 9 conditions of the form $p_{R,x} = 0 \lor p_{R,y} = 0$ from *SD*-inefficient supports
- 256 conditions from Strategyproofness (of which we will use only 85)

Each orbit and Pareto-loser condition immediately eliminates one degree, and 5 of the *SD*-Efficiency conditions also each eliminate one degree immediately due to orbit conditions. This leaves us with 100 degrees of freedom. Using the *smt* method mentioned in Section 5.1, we can then already prove *False* in Isabelle from all these conditions fully automatically within about 8 s.

Deriving a human-readable proof. As mentioned before, one of the goals of the project was to obtain a *structured* proof that a human can follow and, in principle, check every step. I will now describe how I proceeded to find such a proof.

As a first step, the 5 support conditions mentioned above that eliminate a degree have to be identified by hand. They happen to have the form $p_{R,x} = 0 \lor p_{R,y} = 0$ where we know $p_{R,x} = p_{R,y}$ from an orbit condition, so that we can conclude $p_{R,x} = p_{R,y} = 0$. Naturally, this process could also be automated, but seeing as there are only 5 conditions like this, it is hardly worth the effort.

I then naïvely tried to reason 'forward' from the conditions by combining various Strategyproofness conditions and the 4 remaining unused support conditions. It seemed particularly desirable to me to find exact values for variables (e. g. $p_{R_{39},b} = 0$ or $p_{R_{36},a} = 1/2$) since this immediately greatly simplifies all Strategyproofness conditions in which that variable appears. Any value thus determined can be added to Isabelle's simplifier so that one can easily see what remains of any given condition after all the values that were already determined have been plugged in.

My general approach to derive these new equalities was then initially to pick two corresponding Strategyproofness conditions (i. e. two profiles R_1 and R_2 that differ only by one agent's manipulation modulo a renaming of alternatives). Then I hand these – together with lottery conditions and possibly support conditions – to Isabelle's `auto` method. In some cases, the assumptions are then automatically simplified to some useful equation like $p_{R_{36},b} = 0$ or $p_{R_{18},c} = p_{R_9,c}$ or at least an inequality like $p_{R_5,d} \geq 1/2$. This worked for quite a while, but eventually, I was unable to find any new information like this.

I then turned towards the SMT solver for guidance. The situation at this point is that there are some structured proofs of facts and we hand these facts (along with many Strategyproofness conditions) to the SMT solver to derive *False*. The way forward was to attempt to *pull out* facts from the set of facts given to the SMT solver. To do this, I conjectured values of variables (e. g. $p_{R_{42},a} = 0$) that seemed likely to be useful (e. g. because they would simplify many other conditions). Of course, since the conditions are inconsistent, *any* conjecture like this is provable in our context, but a 'good' conjecture can be derived from a small subset of the conditions.

I therefore used the *smt* method to check how many conditions suffice to prove my conjecture. When this set was sufficiently small, I proved the conjecture using the *smt* method, added it to the set of facts given to *smt* in the final proof of *False*, and removed as many of its preconditions as possible from that set in order to determine whether the conjecture was indeed a useful one – the goal, after all, is to make the final 'monolithic' proof step smaller.

With this approach, I was able to easily shrink the final proof step until it disappeared completely. I then proceeded to 'flesh out' all the small facts proven with the *smt* method into structured Isabelle proofs, which was fairly easy since they were all relatively small and Isabelle has good automation for linear arithmetic. The end result was a very linear proof without any 'big' case distinctions, which is remarkable considering that there are over 60 disjunctions in the conditions altogether. At this stage, the proof was clear and detailed enough to derive a rigorous and human-readable (albeit rather lengthy) pen-and-paper proof, which is printed in the appendix of the paper by Brandl *et al.* [1].

7 A Mistake in a Related Result

A previous paper by Brandl *et al.* contained a proof of a weaker version of Theorem 1. The difference is that this weaker theorem additionally assumes that the SDS in question must also be an extension of Random Dictatorship in the sense that it returns the same result as RD if each agent has a unique favourite alternative (i. e. whenever RD is defined).

Corollary 1. *If $m \geq 4$ and $n \geq 4$, there exists no SDS that is an extension of RD and has the following properties: Anonymity, Neutrality, SD-Efficiency, SD-Strategyproofness.*

For the motivation behind this result, see the original presentation by Brandl *et al.* [15]. For our purposes, it should only be said that the proof for this theorem was relatively short and human-readable (it involves only 13 profiles). It was therefore decided to first formalise this weaker theorem in Isabelle (in the hope that it would be considerably easier) and then move on to the proof of Theorem 1.

Like their later proof of Theorem 1, the main part here is also the base case $m = n = 4$ and then employs the lifting argument described in Section 3. I was able to formalise the base case of $m = n = 4$ quickly and without any problems, although it already became apparent that tool support such as that described in Section 5.1 would be very useful.

However, once I attempted to formalise the lifting step (which Brandl *et al.* only described very roughly in a single paragraph since it is usually not very interesting), it became apparent that the lifting argument breaks down in this case: What Brandl *et al.* missed is that unlike the other four properties, the property 'f is an extension of RD' does not 'survive' the lifting, i. e. if f is an RD-extension, it is possible that $f{\downarrow}$ is *not* an RD-extension anymore.

Brandl *et al.* acknowledged this mistake and published a corrigendum [16] in which they suggest to add the additional requirement that f must ignore fully indifferent agents. The result and its problems were superseded by the later correct proof of Theorem 1 anyway. Nevertheless, I find it notable that the formalisation process found a previously undiscovered mistake in a peer-reviewed published work – in particular, a mistake that could only be repaired by introducing additional assumptions.

8 Related Work

Brandl *et al.* [1,15] already give a good overview of work related to Theorem 1 in Social Choice Theory. Geist & Peters [17] give an overview of computer-aided methods in Social Choice Theory in general. I shall therefore restrict this section to formalisations of results from broader Social Choice Theory in theorem provers.

Nipkow [18,19] formalised Arrow's theorem and the Gibbard–Satterthwaite theorem. Gammie [20,21] formalised some more results such as Arrow's theorem, May's theorem, Sen's liberal paradox, and stable matchings. All of these use Isabelle/HOL. The only formalisation of Social Choice Theory outside of Isabelle that I am aware of is one of Arrow's theorem in Mizar by Wiedijk [22].

Brandt *et al.* [23,24] recently built upon my work to formalise another, simpler impossibility result in Isabelle/HOL: that there is no Social Choice Function (SCF) for at least 3 agents / alternatives that fulfils Anonymity, Fishburn-Strategyproofness, and Pareto-Efficiency. The main differences to this work are:

– SCFs return a *set* of winners, not a lottery. The problem can thus be encoded into SAT and SMT is not needed.
– The proof involves only 21 preference profiles instead of 47 and only 33 Strategyproofness conditions instead of 85.
– They do not attempt to construct a human-readable proof and instead use Isabelle's built-in SAT solver to obtain the contradiction in the end.

Due to the different setting of SCFs, most of the specialised automation developed for SDSs could unfortunately not be reused. The *preference_profiles* command and the substantial amount of library material on preferences, however, could be reused. The general structure of the proof (locales, definitions of various notions related to SDSs/SCFs, lifting) was also sufficiently similar that a considerable amount of material on SDSs could easily be adapted. Due to the much smaller size of the proof, the derivation of the SAT conditions from the preference profiles was done by hand since it would have been significantly more work to adapt the automation to SCFs.

It is worth noting that, in contrast to my work here, all examples listed in this section were only concerned with *non-probabilistic* Social Choice Theory. The present work is therefore probably the first published formalisation concerning *randomised* Social Choice Theory.

9 Conclusion

Based on work by Brandl *et al.* [15,25], I have written a fully machine-checked proof of the incompatibility of *SD*-Strategyproofness and *SD*-Efficiency using the Isabelle/HOL theorem prover and, based on this, a 'human-readable' proof. In the process, I have also developed a high-level formalisation of basic concepts of randomised Social Choice Theory and proof automation that automatically defines and derives facts from given preference profiles. Both of these can be used for similar future projects.

This work was also an interesting case study in how interactive theorem provers (like Isabelle) and powerful automated theorem provers (like Z3 and other SMT solvers) can be used not only to formally verify existing mathematical theorems, but also to find completely new and – more or less – human-readable proofs for conjectures. For human mathematicians, simplifying large terms and combining large numbers of complicated linear equations and inequalities is tedious and error-prone, but specialised computer programs (such as SMT solvers or Isabelle's decision procedures for linear arithmetic) excel at it. Using an interactive proof system such as Isabelle has the great advantage that

– it is virtually impossible to make a mistake in a proof,
– one receives immediate feedback on everything, and
– it is easy to check whether an idea works out or not.

The last two points are, in my opinion, often not stressed enough when talking about interactive theorem proving. With a paper proof, changing parts of the proof (e. g. simplifying the presentation or removing unnecessary assumptions) is usually a tedious and error-prone process. With the support of an interactive theorem prover, the consequences of any change become visible immediately, which can make experimentation and 'proof prototyping' much more appealing.

I also believe that this work shows that there is an opportunity for fruitful collaboration between domain experts and interactive proof experts. Together, even brand-new research-level mathematical results can – at least sometimes – be formalised. This can improve the confidence in the correctness of the result tremendously, and, more importantly, it is an excellent way to find and rectify mistakes (as was the case here).

Acknowledgments

I would like to thank Florian Brandl, Felix Brandt, and Christian Geist for bringing the field of randomised Social Choice to my attention as a target for formalisation, and for their continued assistance. I also thank Florian Brandl and Felix Brandt for commenting on a draft of this document.

References

1. Brandl, F., Brandt, F., Eberl, M., Geist, C.: Proving the incompatibility of efficiency and strategyproofness via SMT solving. Journal of the ACM **65**(2) (January 2018) 6:1–6:28
2. Eberl, M.: Randomised social choice theory. Archive of Formal Proofs (May 2016) Formal proof development.
3. Eberl, M.: The incompatibility of *SD*-efficiency and *SD*-strategy-proofness. Archive of Formal Proofs (May 2016) formal proof development.
4. Arrow, K.J.: A difficulty in the concept of social welfare. Journal of Political Economy **58**(4) (1950) 328–346
5. Gibbard, A.: Manipulation of schemes that mix voting with chance. Econometrica **45** (02 1977) 665–681
6. Hales, T.C., Adams, M., Bauer, G., Dang, D.T., Harrison, J., Hoang, T.L., Kaliszyk, C., Magron, V., McLaughlin, S., Nguyen, T.T., Nguyen, T.Q., Nipkow, T., Obua, S., Pleso, J., Rute, J., Solovyev, A., Ta, A.H.T., Tran, T.N., Trieu, D.T., Urban, J., Vu, K.K., Zumkeller, R.: A formal proof of the Kepler conjecture. arXiv **1501.02155** (2015)
7. Tucker, W.: The Lorenz Attractor Exists (revised March 10, 1999). PhD thesis, Uppsala universitet (1999)
8. Viana, M.: What's new on Lorenz strange attractors? The Mathematical Intelligencer **22**(3) (Jun 2000) 6–19
9. Ballarin, C.: Locales: A module system for mathematical theories. Journal of Automated Reasoning **52**(2) (2014) 123–153
10. Eberl, M., Hölzl, J., Nipkow, T.: A verified compiler for probability density functions. In Vitek, J., ed.: Proceedings of the 24th European Symposium on Programming, Springer Berlin Heidelberg (2015) 80–104
11. Böhme, S.: Proof reconstruction for Z3 in Isabelle/HOL. In: 7th International Workshop on Satisfiability Modulo Theories (SMT '09). (2009)

12. Espinoza, D.G.: On Linear Programming, Integer Programming and Cutting Planes. PhD thesis, Georgia Institute of Technology (2006)
13. Applegate, D.L., Cook, W., Dash, S., Espinoza, D.G.: Exact solutions to linear programming problems. Operations Research Letters **35**(6) (2007) 693 – 699
14. Steffensen, J.L.: QSopt_ex – an exact linear programming solver (2014)
15. Brandl, F., Brandt, F., Suksompong, W.: The impossibility of extending Random Dictatorship to weak preferences. Economics Letters **141** (2016) 44 – 47
16. Brandl, F., Brandt, F., Suksompong, W.: Corrigendum to "The impossibility of extending Random Dictatorship to weak preferences" [Econom. Lett. 141 (2016) 44–47]. Economics Letters **145** (2016) 295
17. Geist, C., Peters, D.: Computer-aided methods for social choice theory. In Endriss, U., ed.: Trends in Computational Social Choice. AI Access (2017) 249–267
18. Nipkow, T.: Arrow and Gibbard–Satterthwaite. Archive of Formal Proofs (September 2008) `http://isa-afp.org/entries/ArrowImpossibilityGS.html`, formal proof development.
19. Nipkow, T.: Social choice theory in HOL. Journal of Automated Reasoning **43**(3) (Oct 2009) 289–304
20. Gammie, P.: Some classical results in social choice theory. Archive of Formal Proofs (November 2008) `http://isa-afp.org/entries/SenSocialChoice.html`, formal proof development.
21. Gammie, P.: Stable matching. Archive of Formal Proofs (October 2016) `http://isa-afp.org/entries/Stable_Matching.html`, formal proof development.
22. Wiedijk, F.: Formalizing Arrow's theorem. Sadhana **34**(1) (Feb 2009) 193–220
23. Brandt, F., Saile, C., Stricker, C.: Voting with ties: Strong impossibilities via sat solving. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. AAMAS '18, Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems (2018) 1285–1293
24. Brandt, F., Eberl, M., Saile, C., Stricker, C.: The incompatibility of Fishburn-strategyproofness and Pareto-efficiency. Archive of Formal Proofs (March 2018) `http://isa-afp.org/entries/Fishburn_Impossibility.html`, formal proof development.
25. Brandl, F., Brandt, F., Geist, C.: Proving the incompatibility of efficiency and strategyproofness via SMT solving. Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI) (2016)