A Calculus for Computing Structured Justifications for Election Outcomes

Arthur Boixel, Ulle Endriss, Ronald de Haan

Institute for Logic, Language and Computation (ILLC), University of Amsterdam
{a.boixel, u.endriss, r.dehaan}@uva.nl

Abstract

In the context of social choice theory, we develop a tableau-based calculus for reasoning about voting rules. This calculus can be used to obtain structured explanations for why a given set of axioms justifies a given election outcome for a given profile of voter preferences. We then show how to operationalise this calculus, using a combination of SAT solving and Answer Set Programming, to arrive at a flexible framework for presenting human-readable justifications to users.

1 Introduction

When a group needs to take a decision and its members have diverging preferences about how to rank the available options, we might use a voting rule to make that choice. But, as is well known, there are many different voting rules one could pick; and different rules will sometimes produce different election outcomes (Brams and Fishburn 2002). Agreeing on a specific rule can be difficult, and any one rule might look arbitrary to nonexperts.[1] A possible response to this dilemma, recently advocated by several authors (Cailloux and Endriss 2016; Procaccia 2019; Boixel and Endriss 2020; Peters et al. 2020), would be to attempt to *justify* election outcomes *directly from first principles*, which in the realm of social choice theory means to justify them in terms of *axioms*. Examples for axioms include *anonymity* and the *Pareto principle* (Zwicker 2016). The basic idea is that if, for a given profile of preferences, choosing a given outcome is the only way in which to be consistent with a given set of basic normative principles (i.e., axioms) that everyone can agree with, then this is the outcome we should choose.

Such justifications amount to complex arguments, and more research is required to make them understandable to users. In this paper, we take steps in this direction by developing an approach to automatically generate *structured explanations* for why a given set of axioms logically entails the acceptance of a given outcome for a given profile. To clarify what we mean by this, let's look at an example.

[1] For example, (a nonexpert might wonder) why should we assign scores exactly as prescribed by the famous Borda rule? Or why should we trust the majority rule to choose between two alternatives when it is not even well-defined for three alternatives?

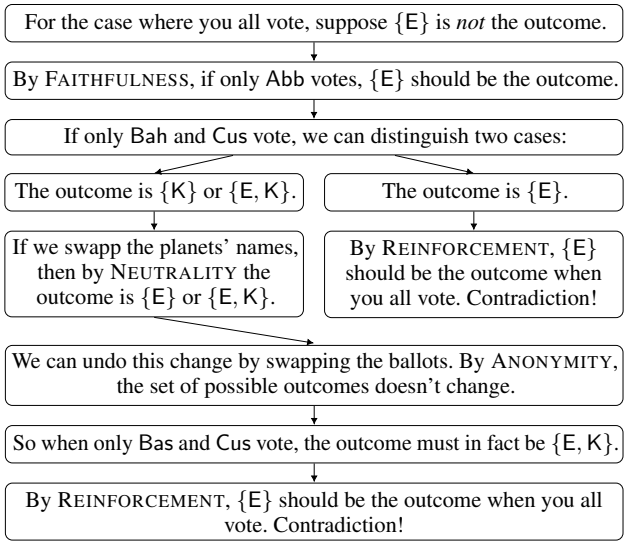

Figure 1: Informal argument for choosing E.

Example 1. Abb, Bah, and Cus, the members of the organising committee of the 42nd Galactic Singing Contest, need to decide where to hold it. (E)arth and (K)epler-452b have been shortlisted. Each member has their own preferences:

$$\mathsf{Abb} : \mathsf{E} \succ_{\mathsf{Abb}} \mathsf{K} \qquad \mathsf{Bah} : \mathsf{E} \succ_{\mathsf{Bah}} \mathsf{K} \qquad \mathsf{Cus} : \mathsf{K} \succ_{\mathsf{Cus}} \mathsf{E}$$

Intuitively, it seems that picking Earth, supported by a simple majority, would be a reasonable decision. But because they will be held accountable for the decision, the committee members instead want to justify their decision in terms of the normative principles enshrined in the Voting Section of the Galactic Constitution. There are four such principles:

- FAITHFULNESS: Should there be just a single voter, their most preferred alternative shall win.
- ANONYMITY: All voters shall be treated equally.
- NEUTRALITY: All alternatives shall be treated equally.
- REINFORCEMENT: If the choices made by two disjoint groups of voters overlap, then this overlap shall be the outcome when the members of both groups vote together.

Equipped with the latest technology, the committee relies on their supercomputer to find a justification. After setting up the

machine, they obtain the tree-like output depicted in Figure 1. Because it is easy to follow and relies on accepted normative principles, the committee feels comfortable announcing the decision: the Contest will take place on Earth. △

Figure 1 is an informal rendering of a formal proof showing that assuming that E is *not* the unique winner of the election would be in contradiction with our four axioms. So this proof *explains*, in a structured manner, how those axioms *justify* electing E.[2] In this paper, we develop a tableau-style calculus (D'Agostino 1999) to produce proofs of this kind—and, more generally, proofs that show that accepting certain axioms would be inconsistent with certain constraints on outcomes (such as not electing E in the profile above). Our calculus can be used with arbitrary axioms definable in the standard model of irresolute social choice functions with variable electorates (Arrow, Sen, and Suzumura 2002). We then show how to operationalise this calculus, using a combination of SAT solving (Biere, Heule, and van Maaren 2009) and Answer Set Programming (ASP) (Brewka, Eiter, and Truszczyński 2011), to generate justifications in practice. The use of ASP, in particular, allows for great flexibility in generating justifications that are tailored to the needs of specific audiences.

Related work. Cailloux and Endriss (2016) introduced the notion of axiomatic justification for an election outcome and designed an algorithm for generating justifications in terms of the axioms used by Young (1974) to characterise the Borda rule. Their explanations for how these axioms justify a given outcome take the form of Hilbert-style proofs. Peters et al. (2020) showed that this algorithm is optimal in the sense of producing the shortest possible proofs (for these axioms).

Boixel and Endriss (2020) developed a more general model of justifications that can be used with arbitrary sets of axioms, where—however—explanations are simply minimally unsatisfiable sets of constraints, i.e., unstructured objects that pinpoint *which* parts of the axioms involved force the outcome in question but that do not make explicit *how* they do so. Boixel and De Haan (2021) studied the computational complexity of this approach, and Nardi (2021) demonstrated that, through the use of heuristics, it can be optimised to an extent that allows for the efficient generation of (unstructured) justifications for small elections of practical interest.

Paper outline. The remainder of this paper is organised as follows. We start with some preliminaries in Section 2 and define our tableau-based calculus in Section 3. Building on the calculus, we present our notion of structured justification in Section 4 and show, in Section 5, how to retrieve such justifications in practice. We conclude in Section 6.

Code. The code used to operationalise the calculus is available online (Boixel, Endriss, and De Haan 2021).

2 Preliminaries

We start by recalling some basic notions from the theory of voting (Zwicker 2016). We then proceed to introduce additional notions that will be necessary to define the calculus.

2.1 Voting Theory

Let X of size $|X| = m$ be a finite set of *alternatives*, and let $\mathcal{L}(X)$ denote the set of all strict linear orders on X. The elements $\succ$ of $\mathcal{L}(X)$, strict rankings of the alternatives in X, are used to encode the preferences of individual agents.

Let N^* of size $|N^*| = n$ be a finite set of *agents*. A *profile of preferences* $\succ_N$ for an electorate $N \subseteq N^*$ is a mapping associating each agent $i \in N$ with a preference $\succ_i \in \mathcal{L}(X)$. The set $\bigcup_{N \in 2^{N^*} \setminus \{\emptyset\}} \mathcal{L}(X)^N$ of all possible profiles for all possible electorates is denoted by $\mathcal{L}(X)^+$.

A *voting rule* is a function $F : \mathcal{L}(X)^+ \to 2^X \setminus \{\emptyset\}$ that assigns to each profile $\succ_N \in \mathcal{L}(X)^+$ a nonempty set of alternatives $F(\succ_N)$, the *election winners* for that profile. Note that this definition of F as an *irresolute* rule accounts for the fact that there could be ties. Many different voting rules have been proposed in the literature and are used in practice. Well-known examples include the *plurality rule* and the *Borda rule* (Brams and Fishburn 2002).

2.2 Constraining Voting Rules

Different voting rules satisfy different normative principles, or *axioms*. In practice, we may think of an axiom as a collection of local restrictions on the behaviour of a voting rule, each focusing on concrete inputs for the rule. We call these restrictions the *instances* of the axiom in question. For example, the *anonymity* axiom enforces equal treatment of the voters. An instance of this axiom would refer to two concrete profiles (that are the same up to a permutation of the agents) and then enforce the fact that they should be assigned the same outcome. Note that an instance A' of an axiom A can itself be seen as a very simple axiom.

Given an axiom instance A', we use $\mathbb{P}(A') \subseteq \mathcal{L}(X)^+$ to denote the set of concrete profiles referred to in its definition. When A' is an instance of some axiom A with $\mathbb{P}(A') = \{\succ^1_{N_1}, \ldots, \succ^k_{N_k}\}$, i.e., when A' refers to k profiles with electorates $N_1, \ldots, N_k$, we sometimes use the expression $A(\succ^1_{N_1}, \ldots, \succ^k_{N_k})$ to refer to A' (e.g., in Figure 2).

Given an axiom (or an axiom instance) A, we define its *interpretation* $\mathbb{I}(A)$ as the set of voting rules that satisfy A. This notion naturally extends to sets of axioms; for a set of axioms $\mathcal{A}$, the set $\mathbb{I}(\mathcal{A}) = \bigcap_{A \in \mathcal{A}} \mathbb{I}(A)$ contains the voting rules that satisfy *all* the axioms in $\mathcal{A}$.

Axioms and their instances encode normative restrictions. We use *outcome statements* to encode more practical restrictions. An outcome statement $s = \langle \succ_N, \mathcal{O} \rangle$ constrains a voting rule to assign one of the outcomes in $\mathcal{O} \subseteq 2^X \setminus \{\emptyset\}$ to the profile $\succ_N \in \mathcal{L}(X)^+$.[3] Note that we can technically think of s as a very simple axiom. Thus, the notion of *interpretation* can also be applied to (sets of) statements. A nonempty set S of outcome statements is interpreted as follows:

$$\mathbb{I}(S) = \bigcap_{\langle \succ_N, \mathcal{O} \rangle \in S} \{ F : \mathcal{L}(X)^+ \to 2^X \setminus \{\emptyset\} \mid F(\succ_N) \in \mathcal{O} \}$$

[2] Note that here we do not need to refer to any specific voting rule. Indeed, settling on a voting rule would involve much greater a commitment, as it would entail also taking a position on the outcome to be selected for every other conceivable profile.

[3] Such outcome statements also feature in the work of Cailloux and Endriss (2016), where they take on the role of atomic propositions in a logical language to speak about voting rules.

We refer to statements of the form $s = \langle \succ_N, 2^X \setminus \{\emptyset\} \rangle$ as *vacuous*, given that they are satisfied by all voting rules. We furthermore refer to statements of the form $s = \langle \succ_N, \emptyset \rangle$ as *inconsistent*, given that they are satisfied by none.

3 A Calculus to Reason about Voting Rules

In this section we define a calculus that can be used to automate basic reasoning tasks regarding voting rules. We start by describing the problem such a calculus is supposed to solve and give several examples of potential applications. We then define the calculus and prove its correctness.

3.1 Purpose

Given a finite set $\mathcal{A}$ of axiom instances and a finite set $\mathcal{S}$ of outcome statements with $\mathbb{I}(\mathcal{A}) \cap \mathbb{I}(\mathcal{S}) = \emptyset$, our calculus will allow us to generate a proof for this fact. In other words, it will allow us to demonstrate that—*and why*—there exists no voting rule that satisfies all the axiom instances in $\mathcal{A}$ while also agreeing with all the outcome statements in $\mathcal{S}$. Why is this useful? Let us review three possible applications:

- For $\mathcal{S} = \emptyset$, a proof of $\mathbb{I}(\mathcal{A}) \cap \mathbb{I}(\mathcal{S}) = \emptyset$ would be a proof of $\mathbb{I}(\mathcal{A}) = \emptyset$, i.e., the fact that no voting rule can satisfy all the axiom instances in $\mathcal{A}$. So this would show that, taken together, the axioms giving rise to $\mathcal{A}$ constitute an unsatisfiable set of requirements, i.e., this would amount to a proof of an *impossibility theorem*. Such impossibility theorems are central to social choice theory and have greatly contributed to our understanding of collective decision making (Arrow, Sen, and Suzumura 2002).[4]
- If $\mathcal{S}$ consists of statements of the form $\langle \succ_N, F(\succ_N) \rangle$ for some specific voting rule F and $\mathcal{A}$ only includes instances of one specific axiom A, then a proof of $\mathbb{I}(\mathcal{A}) \cap \mathbb{I}(\mathcal{S}) = \emptyset$ would demonstrate that F violates A. This can be useful, for instance, when trying to construct an argument against using F (Cailloux and Endriss 2016).
- If there is just a single outcome statement in $\mathcal{S}$ and that statement is of the form $\langle \succ_{N^*}, 2^X \setminus \{\emptyset, X^*\} \rangle$ for some set $X^* \subseteq X$, then that would show that accepting the axioms the elements of $\mathcal{A}$ are instances of forces us to accept X^* as the election outcome under profile $\succ_{N^*}$. Thus, these axioms would *justify* electing X^*. This application will be the main focus of the second part of this paper.

Importantly, in case $\mathbb{I}(\mathcal{A}) \cap \mathbb{I}(\mathcal{S})$ is *not* actually empty, any attempt of a proof using our calculus will (correctly) fail in finite time. While such a failed proof is a less natural object to interact with than a successful proof, it still technically provides a certificate for the fact that $\mathbb{I}(\mathcal{A}) \cap \mathbb{I}(\mathcal{S}) \neq \emptyset$.

[4]Note that a proof of $\mathbb{I}(\mathcal{A}) = \emptyset$ only excludes the existence of a voting rule satisfying the axioms on the specific profile sizes occurring in $\mathcal{A}$. This is all we need if we are interested in voting rules defined for variable electorates (so *including* the electorates occurring in $\mathcal{A}$). But in case we would be satisfied with finding a rule that works only for specific electorates (say, an electorate that is larger than any of those occurring in $\mathcal{A}$), additional work is required to obtain a full impossibility result. We can still use our calculus to prove the impossibility of satisfying the axioms on profiles of a fixed size. Such results have been used as "base cases" for inductive proofs in the literature on computer-aided proofs for impossibility theorems (Tang and Lin 2009; Geist and Peters 2017).

3.2 The Calculus

To prove the unsatisfiability of several properties of voting rules, we require some kind of refutation procedure. The *tableau methodology* (D'Agostino 1999) seems particularly suited to the task of developing such a procedure.

When using a tableau method as a calculus for a given logic, a *tableau* is a tree, the nodes of which are sets of formulas in that logic. To show that a given set Φ of such formulas is unsatisfiable, we construct a tree with $S_0 := \Phi$ at the root. We use so-called *expansion rules* to repeatedly add new nodes below the current leaf nodes. There are two kinds of rules. First, we can add a single new node below a leaf node S by copying S and adding an additional formula that is logically entailed by the formulas in S (e.g., if $(p \wedge q) \in S$, we might add $S \cup \{p\}$ as the child). Second, we can branch and add two sibling nodes below a leaf node to make a case distinction (e.g., if $(p \vee q) \in S$, we might add $S \cup \{p\}$ as the left child and $S \cup \{q\}$ as the right child). We *close* a branch whenever its leaf node is obviously unsatisfiable (e.g., when it includes both p and $\neg p$). We have found a proof for the unsatisfiability of Φ if we can close *all* branches.

We are now going to adapt this approach to our problem and formally define our calculus by describing what a tableau is, how to expand it, and how to close it. Broadly speaking, the outcome statements take on the role of formulas and the axiom instances license tailor-made expansion rules.

Recall that a *rooted tree* is a directed acyclic graph with a single source (the *root*) and a maximum in-degree of 1. Formally, for our purposes, a *tableau* is a rooted tree with nodes that are sets of outcome statements.

To show that $\mathbb{I}(\mathcal{A}) \cap \mathbb{I}(\mathcal{S}) = \emptyset$, we construct such a tableau as follows. We start with a very simple tableau consisting of just a single node $S_0 := \mathcal{S}$ containing all the outcome statements in $\mathcal{S}$. We then expand our tableau by successively applying *expansion rules*. New nodes are added, making some of the logical consequences of (complying with) the statements in $\mathcal{S}$ and the axiom instances in $\mathcal{A}$ explicit. We continue to expand until we obtain a tableau that makes the contradictory requirement of satisfying all properties in $\mathcal{S}$ and $\mathcal{A}$ self-evident (if possible). Every expansion rule appends one or two new nodes immediately below one of the leaf nodes of the current tableau. There are three such rules:

- **Axiom-driven expansion rule.** For any axiom instance $A \in \mathcal{A}$ and any profile $\succ_N \in \mathbb{P}(A)$, a branch ending in a leaf node S can be expanded by appending a node $S' = S \cup \{\langle \succ_N, \mathcal{O} \rangle\}$, where $\mathcal{O} = \{ F(\succ_N) \mid F \in \mathbb{I}(S) \cap \mathbb{I}(A) \}$, provided that $\mathbb{I}(S') \subsetneq \mathbb{I}(S)$.
- **Branching rule.** A branch ending in a leaf node S that includes a statement of the form $s = \langle \succ_N, \mathcal{O}_1 \cup \mathcal{O}_2 \rangle$, where $\mathcal{O}_1$ and $\mathcal{O}_2$ are nonempty sets with $\mathcal{O}_1 \cap \mathcal{O}_2 = \emptyset$, can be split by appending two nodes to S, namely $S_1 = \{\langle \succ_N, \mathcal{O}_1 \rangle\} \cup S \setminus \{s\}$ and $S_2 = \{\langle \succ_N, \mathcal{O}_2 \rangle\} \cup S \setminus \{s\}$. If a profile $\succ_N \in \mathbb{P}(\mathcal{A})$ does not currently appear in any of the outcome statements in S, it is still possible to use

the branching rule with respect to $\succ_N$ by assuming that S contains the vacuous statement $\langle \succ_N, 2^X \setminus \{\emptyset\}\rangle$.

- **Simplification rule.** A branch ending in a leaf node S that includes two distinct statements $s_1 = \langle \succ_N, \mathcal{O}_1\rangle$ and $s_2 = \langle \succ_N, \mathcal{O}_2\rangle$ can be expanded by appending to S the node $S' = \{\langle \succ_N, \mathcal{O}_1 \cap \mathcal{O}_2\rangle\} \cup S \setminus \{s_1, s_2\}$.

For a given branch, the axiom-driven expansion rule makes explicit, in terms of outcome statements, the consequences of an axiom instance A as far as profile $\succ_N$ is concerned. Take for example a set S including a statement s_1 saying that for profile $\succ_N = (a \succ_1 b \succ_1 c, b \succ_2 a \succ_2 c)$ we must select an outcome from $\{\{a\}, \{b\}\}$. Then we may want to use the axiom-driven rule associated with an instance of the anonymity axiom to enforce that $\succ_N$ and $\succ'_N = (b \succ_1 a \succ_1 c, a \succ_2 b \succ_2 c)$ take the same outcome. Applying this rule allows us to expand the tree with a new node S' containing the additional statement $s_2 = \langle \succ'_N, \{\{a\}, \{b\}\}\rangle$.[5]

The branching rule allows us to make a case distinction regarding the outcomes still available for a profile $\succ_N$. If the outcome for $\succ_N$ must be in $\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2$, then it must either be in $\mathcal{O}_1$ or in $\mathcal{O}_2$. The branching rule lets us treat these cases separately on two distinct branches.

The simplification rule only makes cosmetic changes and will be useful in practice to (i) clean up a set of statements and (ii) make explicit the inconsistency of such a set.

Let us call a tableau constructed in the manner described *rooted in* $\mathcal{S}$ if its root node consists of the set $\mathcal{S}$, and let us call it *licensed by* $\mathcal{A}$ if all applications of the axiom-driven expansion rule rely on axiom instances in $\mathcal{A}$. A tableau is called *saturated* (relative to $\mathcal{A}$ and $\mathcal{S}$) if no further rules can be applied to it. Furthermore, a tableau is called *closed* if every branch ends in a leaf node that includes at least one inconsistent outcome statement. Otherwise it is *open*.

Note that we have not said anything about the order in which to apply expansion rules. Different strategies will lead to different trees, and not all closed trees will have the same explanatory power. For now, we only focus on the correctness of the calculus, but we will return to this issue in Section 5.

Example 2. Recall the informal proof of Example 1. We now present its formal counterpart, the closed tableau of Figure 2. It is rooted in $\mathcal{S} = \{\langle \succ_{\{\mathsf{A},\mathsf{B},\mathsf{C}\}}, \{\{\mathsf{K}\}, \{\mathsf{E},\mathsf{K}\}\}\rangle\}$, containing a single statement that amounts to ruling out outcome $\{\mathsf{E}\}$ in case all three agents vote. It is licensed by a set $\mathcal{A}$ containing one axiom instance for each of our four axioms (FAI, ANO, NEU, REI) and imposing restrictions on four concrete profiles:

- Profile $\succ_{\{\mathsf{A},\mathsf{B},\mathsf{C}\}}$ containing all the ballots.
- Profile $\succ_{\{\mathsf{A}\}}$ containing only Abb's ballot.
- Profile $\succ_{\{\mathsf{B},\mathsf{C}\}}$ containing the ballots of Bah and Cus.
- Profile $\succ'_{\{\mathsf{B},\mathsf{C}\}}$ containing the ballots of Bah and Cus with the alternatives (or equivalently: the voters) switched.

To save space, most applications (except for two) of the simplification rule are left implicit. Whenever an axiom-driven expansion rule has been used, the corresponding edge is annotated with the associated axiom instance. The branching rule is used once to make a case distinction between outcomes for profile $\succ_{\{\mathsf{B},\mathsf{C}\}}$. Because the two leaves of the tree both contain an inconsistent statement, stating that no outcome can be assigned to the full profile, the tableau is closed. △

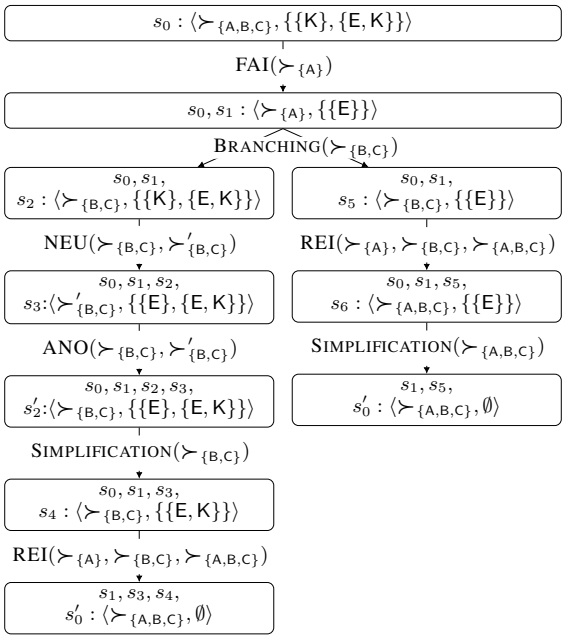

Figure 2: The closed tableau of Example 2.

3.3 Correctness of the Calculus

To prove that there exists no voting rule that satisfies all the outcome statements in a given finite set $\mathcal{S}$ and all the axiom instances in a given finite set $\mathcal{A}$, we construct a tableau rooted in $\mathcal{S}$ and licensed by $\mathcal{A}$. We are now going to show that—whatever the order in which we apply our expansion rules—this process will always terminate eventually,[6] culminating in either a closed tableau (in case of success) or a saturated but open tableau (in case of failure), and that the final tableau is closed if and only if there exists no such voting rule.

Lemma 1 (Termination)**.** *Any application of expansion rules leads, within a finite number of steps, to a tableau that is either closed or saturated.*

Proof. The expansion rules ensure that for any two nodes S and S', with S being the parent of S', we have either $\mathbb{I}(S') \subsetneq \mathbb{I}(S)$ or both $\mathbb{I}(S) = \mathbb{I}(S')$ and $|S'| < |S|$. Thus, as the root $S_0 = \mathcal{S}$ describes a finite set $\mathbb{I}(\mathcal{S})$ of voting rules, all branches must be of finite length. □

Lemma 2 (Soundness)**.** *If we can construct a closed tableau that is rooted in $\mathcal{S}$ and licensed by $\mathcal{A}$, then there exists no voting rule that satisfies both the outcome statements in $\mathcal{S}$ and the axiom instances in $\mathcal{A}$.*

[5] We stress that, in principle, it is possible to think of axioms that would have consequences for more than one profile at a time. In such a case, we would apply the axiom-driven expansion rule multiple times for the same axiom but different profiles.

[6] Yet, in line with the complexity results of Boixel and De Haan (2021), the size of the tableau can be exponential in n and m.

Proof. We will prove the contrapositive: if we are in a node S such that $\mathbb{I}(S) \cap \mathbb{I}(\mathcal{A}) \neq \emptyset$, then applying any expansion rule will lead to the creation of at least one new node S' such that $\mathbb{I}(S') \cap \mathbb{I}(\mathcal{A}) \neq \emptyset$. Consider such a node S and let us review all possible ways of expanding the tableau from S.

An application of the *simplification rule* is only cosmetic and does not affect the soundness of the approach.

An application of the *branching rule* creates a case distinction regarding the possible outcomes a profile can be assigned to. At least one of the new branches will remain open.

Finally, consider an application of the *axiom-driven* expansion rule with respect to an instance A and a profile $\succ_N$. When applying the rule we only make changes to the outcomes available for $\succ_N$. Moreover, we make sure that *all* of the outcomes $F(\succ_N)$ such that $F \in \mathbb{I}(S) \cap \mathbb{I}(A)$ remain available for $\succ_N$. We know by assumption that $\mathbb{I}(S) \cap \mathbb{I}(A) \neq \emptyset$. Thus, there is still some rule satisfying both the new set of statements and the instance A. □

Lemma 3 (Completeness)**.** *If there exists no voting rule that satisfies both the outcome statements in $\mathcal{S}$ and the axiom instances in $\mathcal{A}$, then we will always be able to obtain a closed tree rooted in $\mathcal{S}$ and licensed by $\mathcal{A}$.*

Proof. We shall prove the contrapositive. Take any open saturated tree rooted in $\mathcal{S}$ and licensed by $\mathcal{A}$. We need to show that there is at least one rule in $\mathbb{I}(\mathcal{S}) \cap \mathbb{I}(\mathcal{A})$. By virtue of being open, our tree has a branch ending in a leaf node S^* that is not inconsistent. As it is saturated, we know that:

1. There must be *at least* one statement for each profile in $\mathbb{P}(\mathcal{A})$, as otherwise the axiom-driven expansion rule or the branching rule would still be applicable.
2. All the sets of outcomes referred to in the statements in S^* must be singletons, as otherwise the branching rule would still be applicable.
3. There must be *at most* one statement for each profile occurring at all, as otherwise the simplification rule would still be applicable.

As S^* is not inconsistent, there exists a rule $F^* \in \mathbb{I}(S^*)$. This rule returns the outcomes prescribed by the singleton outcome statements in S^* on all profiles mentioned in S^*, and makes some arbitrary choices for all other profiles. Given that we have $\mathbb{I}(S') \subseteq \mathbb{I}(S)$ for any two nodes with S being the parent of S', we immediately obtain that $F^* \in \mathbb{I}(\mathcal{S})$. So it remains for us to show that $F^* \in \mathbb{I}(\mathcal{A})$.

For the sake of contradiction, suppose F^* violates some $A \in \mathcal{A}$. Recall that the tree is saturated, so we should *not* be able to apply the expansion rule associated with A on S^*.

Note that, if F^* violates A, then so does any other rule in $\mathbb{I}(S^*)$. Indeed, all rules in $\mathbb{I}(S^*)$ agree on the outcomes to select for profiles in $\mathbb{P}(S^*)$, in particular, they all select the same outcomes for profiles in $\mathbb{P}(A)$. So $\mathbb{I}(S^*) \cap \mathbb{I}(A) = \emptyset$.

Consider any profile $\succ_N^* \in \mathbb{P}(A)$. From the previous observation it follows that the set $\mathcal{O} = \{F(\succ_N^*) \mid F \in \mathbb{I}(S^*) \cap \mathbb{I}(A)\}$ is empty. Hence, we can use the axiom-driven expansion rule associated with A on S^* and $\succ_N^*$ and reach a new node S' that contains a statement $s' = \langle \succ_N^*, \emptyset \rangle$. Note that, as $\mathbb{I}(S') = \emptyset$, we have $\mathbb{I}(S') \subsetneq \mathbb{I}(S^*)$ and this is a valid rule application. So we were able to apply an expansion rule on a supposedly saturated tree, which is a contradiction. □

Lemma 3 says that, if there exists no voting rule satisfying $\mathcal{S}$ and $\mathcal{A}$, then every way of applying expansion rules to the initial tableau must lead to a closed tableau. Thus, if $\mathcal{S}$ and $\mathcal{A}$ are indeed incompatible, then our calculus will always find a proof eventually. Note that soundness and completeness together entail that, if *some* sequence of rule applications leads to a closed tableau, then *all* sequences do. So we are free to apply expansion rules in any order we wish and will always obtain the same result. This opens up the way for devising heuristics for using our calculus in practice (see Section 5). We summarise our findings as follows.

Theorem 4 (Correctness)**.** *Our tableau-based calculus is a sound and complete decision procedure to check whether there exists a voting rule that satisfies certain axiom instances and complies with certain outcome statements.*

If we fail to find a closed tableau $\mathcal{T}$ rooted in a set $\mathcal{S}$ and licensed by a set $\mathcal{A}$, Theorem 4 implies that there exists at least one voting rule satisfying both the axiom instances in $\mathcal{A}$ and the statements in $\mathcal{S}$. A (partial) description of such a rule can be found in the leaf of any open branch. Hence, the tableau methodology can also be seen as a search procedure. When constructing a tableau we are in fact searching for *models* making some claim false. Each branch of the tableau constitutes a partial description of such models that are refined during the expansion stage. Being able to close the tableau means that no such model exists, thereby proving the validity of the claim. This view is perfectly in line with what we are trying to do here: to prove that a set of axiom instances $\mathcal{A}$ and a set of outcome statements $\mathcal{S}$ are inconsistent, we try and fail to find a voting rule satisfying both.

4 Using the Calculus to Obtain Justifications

In this section we take advantage of the calculus previously described to define a notion of *structured* justification for collective decisions. We start by defining this notion and explain the differences with the notion of *unstructured* justification introduced by Boixel and Endriss (2020). Finally, we show how the calculus can be used to compute such justifications.

4.1 Structured Justification

Consider a voting scenario involving some agents with preferences stored in a profile $\succ_{N^*}$. Assuming the agents (or others we wish to convince) care about the axioms in some corpus $\mathbb{A}$, how can we justify that some target outcome X^* is the right one to select? Intuitively, we are looking for a step-by-step proof showing *why* satisfying certain axioms in $\mathbb{A}$ enforces the selection of X^*. In other words, we need to prove that there exists no voting rule satisfying these axioms that selects an outcome different from X^*.

We now define a notion of *structured justification*. Our definition is directly inspired by the original definition of Boixel and Endriss (2020) for an (unstructured) justification.

Definition 1 (Structured justification)**.** *Let $\mathbb{A}$ be a corpus of axioms for voting rules $F : \mathcal{L}(X)^+ \to 2^X \setminus \{\emptyset\}$, let $\succ_{N^*}$ be a profile, and let $X^* \subseteq X$ be a target outcome for that profile.*

We say that a triple $\langle \mathcal{A}^{\text{N}}, \mathcal{A}^{\text{E}}, \mathcal{T} \rangle$ of a set of axioms, a set of axiom instances, and a tableau is a ***structured justification*** *for the set X^* winning the election under profile $\succ_{N^*}$ if and only if the following four conditions are satisfied:*

1. ***Explanatoriness.*** *$\mathcal{T}$ is a closed tableau that is rooted in $\mathcal{S} = \{\langle \succ_{N^*}, 2^X \setminus \{X^*, \emptyset\}\rangle\}$ and licensed by $\mathcal{A}^{\text{E}}$.*
2. ***Relevance.*** *Every axiom instance in $\mathcal{A}^{\text{E}}$ instantiates some axiom in $\mathcal{A}^{\text{N}}$.*
3. ***Adequacy.*** *All axioms in $\mathcal{A}^{\text{N}}$ belong to $\mathbb{A}$: $\mathcal{A}^{\text{N}} \subseteq \mathbb{A}$.*
4. ***Nontriviality.*** *There exists at least one voting rule that satisfies all of the axioms in $\mathcal{A}^{\text{N}}$: $\mathbb{I}(\mathcal{A}^{\text{N}}) \neq \emptyset$.*

We call $\mathcal{A}^{\text{N}}$ the ***normative basis****, $\mathcal{T}$ the* ***explanation tableau****, and $\mathcal{A}^{\text{E}}$ the* ***explanation basis****.*

The explanatoriness requirement ensures that the tableau $\mathcal{T}$ is a proper proof for the fact that it is impossible to satisfy the axiom instances in $\mathcal{A}^{\text{E}}$ if we select an outcome different from X^* for $\succ_{N^*}$, i.e., if we comply with the special statement in $\mathcal{S}$. Relevance and adequacy ensure that any argument featured in the proof, i.e., any axiom-driven expansion rule used to construct $\mathcal{T}$, derives from a normative principle in $\mathbb{A}$. Finally, the normative basis should be such that there exists a least one voting rule satisfying all the axioms in it. Together with the explanatoriness requirement this entails that these axioms enforce the selection of the target outcome.

The explanation basis $\mathcal{A}^{\text{E}}$ consists of axiom instances that together enforce the selection of X^*. So if we were to provide our audience only with $\langle \mathcal{A}^{\text{N}}, \mathcal{A}^{\text{E}} \rangle$, this would still constitute some kind of justification. This is essentially the notion of justification proposed by Boixel and Endriss (2020). But such justifications have limited explanatory power. While satisfying the instances in $\mathcal{A}^{\text{E}}$ does enforce the selection of X^*, it may be unclear *why* that is so. The burden of understanding how to use (the restrictions imposed by) the instances to obtain an insightful explanation is left to the user. Trading this unstructured set of axiom instances $\mathcal{A}^{\text{E}}$ for a closed tableau $\mathcal{T}$ presenting a step-by-step proof is a way to address this issue.[7]

Example 3. Recall the informal proof presented in Example 1 and its formal counterpart, the closed tableau $\mathcal{T}$ of Example 2. Consider the corpus $\mathbb{A} = \{\text{FAI}, \text{NEU}, \text{ANO}, \text{REI}\}$. The tableau $\mathcal{T}$ is closed, rooted in a set $\mathcal{S}$ containing a statement preventing E from winning in profile $\succ_{\{\mathsf{A},\mathsf{B},\mathsf{C}\}}$, and licensed by a set of axiom instances $\mathcal{A}^{\text{E}}$ derived from the normative basis $\mathcal{A}^{\text{N}} = \mathbb{A}$. So $\langle \mathcal{A}^{\text{N}}, \mathcal{A}^{\text{E}}, \mathcal{T} \rangle$ is a structured justification showing why making E the unique winner in profile $\succ_{\{\mathsf{A},\mathsf{B},\mathsf{C}\}}$ is the only way to satisfy the axioms in $\mathcal{A}^{\text{N}}$.[8] △

4.2 Obtaining Justifications

We now show how to use our calculus to obtain a structured justification. We then discuss the limitations of this approach and give some pointers on how to make it more efficient.

Suppose there exists a structured justification, grounded in a normative basis $\mathcal{A}^{\text{N}}$, for selecting X^* under profile $\succ_{N^*}$. Let $\mathcal{A}$ be the set containing *all* the axiom instances for the axioms in $\mathcal{A}^{\text{N}}$. By assumption, for any voting rule F satisfying the instances in $\mathcal{A}$ it is the case that F fails to comply with $\mathcal{S} = \{\langle \succ_{N^*}, 2^X \setminus \{X^*, \emptyset\}\rangle\}$. Theorem 4 now tells us that any tree rooted in $\mathcal{S}$ and licensed by $\mathcal{A}$ can be closed. So we can use the calculus to compute such a closed tree $\mathcal{T}$ and thus obtain a justification $\langle \mathcal{A}^{\text{N}}, \mathcal{A}, \mathcal{T} \rangle$. Recall the requirements of Definition 1. As the tree is closed, explanatoriness is satisfied. Adequacy and relevance trivially hold as well. Only the nontriviality of $\mathcal{A}^{\text{N}}$ still remains to checked, which can be done using a SAT solver (Geist and Peters 2017).

Although technically correct, this direct approach may not be the most efficient one in practice. Indeed, a single axiom can give rise to many instances, most of which are likely useless. Moreover, working with the entire set of axiom instances offers little to no control regarding the arguments used to construct the tableau. By following this approach we rely on the calculus to both (i) identify this set of arguments and (ii) transform it into a comprehensible step-by-step proof.

An alternative approach is to *first* compute a minimal explanation basis $\mathcal{A}^{\text{E}}$ (e.g., by encoding the problem in propositional logic and using a SAT-based approach), and to *then* construct a tableau $\mathcal{T}$ licensed by this small set $\mathcal{A}^{\text{E}}$ rather than the much larger set $\mathcal{A}$ of all axiom instances of the normative basis selected for exploration. In other words, we can use the approach of Boixel and Endriss (2020), refined by Nardi (2021), as a black-box algorithm to first compute an unstructured justification $\langle \mathcal{A}^{\text{N}}, \mathcal{A}^{\text{E}} \rangle$ and then use our calculus to find a tableau $\mathcal{T}$ licensed by the set $\mathcal{A}^{\text{E}}$ thus obtained.

5 Structured Justifications in Practice

Earlier, we explained how to use our calculus to structure a previously computed unstructured justification. The purpose of this section is twofold. We start by showing how to operationalise this approach using ASP (Brewka, Eiter, and Truszczyński 2011; Gebser et al. 2012; Gelfond 2008).[9] We then argue that the ASP approach is very flexible and gives us a lot of control regarding the justifications we want to find.

5.1 Simulating the Calculus in ASP

ASP offers a declarative problem-solving paradigm that is based on logic programming. One can express a search problem using facts, rules, and constraints, and a solver then provides solutions (in the form of answer sets).

The search problem. We face the following search problem. Given a set $\mathcal{A}^{\text{E}}$ of axiom instances, minimally enforcing the selection of some outcome X^* under profile $\succ_{N^*}$,[10] we need to find a closed tableau showing *why* this is the case. ASP allows us to model the specification of a given search problem independently of the concrete instance we wish to

[7] Boixel and Endriss (2020) also require $\mathcal{A}^{\text{E}}$ to be *minimal*, in the sense of no strict subset still being able to enforce the choice of X^*. Because the smallest tableau might not always provide the best explanation, we do not impose a corresponding minimality requirement on $\mathcal{T}$ (see Section 5 for a discussion of this point).

[8] Requirements 1, 2, and 3 from Definition 1 are trivially satisfied. The non-triviality requirement is also satisfied: the plurality rule (Zwicker 2016), defined for two alternatives and (up to) three agents, satisfies all the axioms in $\mathcal{A}^{\text{N}}$.

[9] For our implementation (Boixel, Endriss, and De Haan 2021) we used clingo as ASP grounder/solver (Gebser et al. 2008).

[10] We can use SAT solving techniques to obtain such a set (Boixel and Endriss 2020; Nardi 2021).

solve. By using a fixed ASP program, we can thus virtually structure any set of axiom instances $\mathcal{A}^{\text{E}}$; we simply need to complete the program with facts encoding those instances.

The generate-and-test approach. Classically, an ASP program is designed following a two-step process (see, e.g., Gebser et al. 2012). First, we describe what potential solutions look like (the *generate*-step). Then, we encode the requirement that only correct solutions—the ones satisfying our requirements—should be kept (the *test*-step).

In our case, potential solutions are rooted trees containing some extra information. We encode these as a finite set of nodes (with a single root) linked via edges forming a tree. Each node is associated with some outcome statements and each edge is linked to a concrete application of an expansion rule. This encodes solutions that have the appropriate general structure but that are not guaranteed to be correct. For example, trees may be open, rooted in the wrong set of statements, or using an incorrect expansion rule between two nodes.

To finish the *test*-part, we encode various additional requirements. We ensure that trees are rooted in the correct set of statements. We ensure that each leaf of the tree contains at least one inconsistent statement. The most demanding part is to ensure that any edge linking a node N_1 with another node N_2 is associated with an expansion rule and that the statements associated with both nodes correspond to a correct application of the rule. We do this by checking that N_1 is associated with statements on which the expansion rule can be applied, and that the set of statements associated with N_2 correctly reflects the consequences of the rule application. This makes it straightforward to add support for new axioms.

5.2 Finding *Optimal* Proof Trees

As we saw, a single ASP program can be used to structure any set of axiom instances into a proof tree. The approach is flexible, and adding or removing support for a given axiom can be done easily. This approach does not only allow us to find *correct* proof trees, but—as we will argue—it also enables finding *optimal* proof trees, according to whichever criteria a specific audience might be interested in.

Selecting optimal proof trees. To structure a given set of axiom instances $\mathcal{A}^{\text{E}}$ into *some* tableau $\mathcal{T}$ is straightforward: start from the root, apply expansion rules (pseudo-)randomly, and stop once all branches are closed. While this naïve approach works, it gives us little to no control regarding the characteristics of the tree found, and refining the order in which to apply rules (e.g., delaying branching as much as possible) does not significantly improve on this base line.

One could devise a tailor-made algorithm for finding optimal tableaux with respect to a given criterion. However, such an approach is likely not productive for practical use—especially not when the optimality criteria are changing or not fully known in advance. Any tailor-made algorithm would have to be adapted for any (small) change in the criteria, and any such adaptation likely takes significant effort. In particular, this would make experimenting with different optimality criteria tedious and time-consuming.

Instead, the declarative approach of ASP provides a much more flexible solution. One can straightforwardly (and quickly) add optimisation statements that express given optimality criteria—and update these when needed. For example, finding tableaux with a minimum number of nodes can be done by simply adding $\texttt{\#minimize}\,\{N : \texttt{node}(N)\}$.

Moreover, the declarative approach of ASP allows us to combine different optimisation criteria, with different priorities. For example, the tree in Figure 2 has a minimum number of nodes, and—amongst all such trees—introduces new profiles to the proof as late as possible.

One argument in favour of tailor-made algorithms is that one might be able to offer better worst-case running-time guarantees than for the ASP approach. However, ASP solvers are incredibly efficient in many cases, and worst-case running-time guarantees might not be crucial for use in practice. For example, using the ASP approach, we were able to find proof trees such as the one of Figure 2 in a few of minutes—even without significant use of advanced performance-improving ASP encoding techniques such as symmetry breaking.

Identifying optimality criteria. An interesting topic for future research is to establish which optimisation criteria lead to an optimal (human) understanding of explanations. One might expect that short trees that use as little branching as possible fit the bill—and this is indeed our first impression, after running our proof-of-concept implementation on various examples. There might, however, be different or additional criteria that lead to better-understandable explanations, and these criteria may differ from one application to another.

As already suggested by Boixel and Endriss (2020), finding out what makes an explanation understandable is an empirical question that could be investigated by means of crowdsourcing experiments. The flexibility of our ASP approach can provide a basis for such a research agenda. Following a data-driven axiomatic approach (d'Eon and Larson 2020), one might also get try to get insights into the type of arguments (or axioms) that are most convincing for a given audience.

6 Conclusion

We defined a tableau-based calculus to reason about voting rules. With modest adaptations, such a calculus could also be used to reason about mechanisms studied in other domains; examples include matching and participatory budgeting. We highlighted several applications for which such a calculus can prove useful. In particular, we used it to provide a notion of structured justification for election outcomes.

Using a combination of SAT solving techniques and ASP we showcased what such justifications could look like. The ASP approach is quite flexible and enables the setting up of real-life experiments that could improve our understanding of what makes for a *good* justification in practice.

Another important direction of future research will be to investigate how to automatically transform a formal justification obtained through the calculus, i.e, a closed tableau, into an easy-to-follow argument expressed in natural language.

References

Arrow, K. J.; Sen, A. K.; and Suzumura, K., eds. 2002. *Handbook of Social Choice and Welfare*, volume 1. Elsevier.

Biere, A.; Heule, M.; and van Maaren, H., eds. 2009. *Handbook of Satisfiability*. IOS Press.

Boixel, A.; and Endriss, U. 2020. Automated Justification of Collective Decisions via Constraint Solving. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2020)*. IFAAMAS.

Boixel, A.; Endriss, U.; and De Haan, R. 2021. Supplementary Material for "A Calculus for Computing Structured Justifications for Election Outcomes". Zenodo. Available at https://doi.org/10.5281/zenodo.5767215.

Boixel, A.; and De Haan, R. 2021. On the Complexity of Finding Justifications for Collective Decisions. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI-2021)*.

Brams, S. J.; and Fishburn, P. C. 2002. Voting Procedures. In Arrow, K. J.; Sen, A. K.; and Suzumura, K., eds., *Handbook of Social Choice and Welfare*, volume 1, chapter 4, 173–236. Elsevier.

Brewka, G.; Eiter, T.; and Truszczyński, M. 2011. Answer set programming at a glance. *Communications of the ACM*, 54(12): 92–103.

Cailloux, O.; and Endriss, U. 2016. Arguing about Voting Rules. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2016)*. IFAAMAS.

D'Agostino, M. 1999. Tableau Methods for Classical Propositional Logic. In D'Agostino, M.; Gabbay, D. M.; Hähnle, R.; and Posegga, J., eds., *Handbook of Tableau Methods*, chapter 2, 45–123. Elsevier.

d'Eon, G.; and Larson, K. 2020. Testing axioms against human reward divisions in cooperative games. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2020)*. IFAAMAS.

Gebser, M.; Kaminski, R.; Kaufmann, B.; Ostrowski, M.; Schaub, T.; and Thiele, S. 2008. A User's Guide to gringo, clasp, clingo, and iclingo. Technical report, University of Potsdam.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2012. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Geist, C.; and Peters, D. 2017. Computer-Aided Methods for Social Choice Theory. In Endriss, U., ed., *Trends in Computational Social Choice*, chapter 13, 249–267. AI Access.

Gelfond, M. 2008. Answer Sets. In van Harmelen, F.; Lifschitz, V.; and Porter, B., eds., *Handbook of Knowledge Representation*, chapter 7, 285–316. Elsevier.

Nardi, O. 2021. *A Graph-Based Algorithm for the Automated Justification of Collective Decisions*. Master's thesis, ILLC, University of Amsterdam.

Peters, D.; Procaccia, A. D.; Psomas, A.; and Zhou, Z. 2020. Explainable Voting. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems (NeurIPS-2020)*.

Procaccia, A. D. 2019. Axioms Should Explain Solutions. In Laslier, J.-F.; Moulin, H.; Sanver, M. R.; and Zwicker, W. S., eds., *The Future of Economic Design*, 195–199. Springer.

Tang, P.; and Lin, F. 2009. Computer-aided Proofs of Arrow's and other Impossibility Theorems. *Artificial Intelligence*, 173(11): 1041–1053.

Young, H. P. 1974. An Axiomatization of Borda's Rule. *Journal of Economic Theory*, 9(1): 43–52.

Zwicker, W. S. 2016. Introduction to the Theory of Voting. In Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds., *Handbook of Computational Social Choice*, chapter 2, 23–56. Cambridge University Press.