On the Complexity of Finding Justifications for Collective Decisions

Arthur Boixel, Ronald de Haan
Institute for Logic, Language and Computation (ILLC), University of Amsterdam
a.boixel@uva.nl, me@ronalddehaan.eu

Abstract

In a collective decision-making process, having the possibility to provide non-expert agents with a justification for why a target outcome is a good compromise given their individual preferences, is an appealing idea. Such questions have recently been addressed in the computational social choice community at large—whether it was to explain the outcomes of a specific rule in voting theory or to seek transparency and accountability in multi-criteria decision making. Ultimately, the development of real-life applications based on these notions depends on their practical feasibility and on the scalability of the approach taken. In this paper, we provide computational complexity results that address the problem of finding and verifying justifications for collective decisions.

In particular, we focus on the recent development of a general notion of justification for outcomes in voting theory. Such a justification consists of a step-by-step explanation, grounded in a normative basis, showing how the selection of the target outcome follows from the normative principles considered. We consider a language in which normative principles can be encoded—either as an explicit list of instances of the principles (by means of quantifier-free sentences), or in a succinct fashion (using quantifiers). We then analyse the computational complexity of identifying and checking justifications. For the case where the normative principles are given in the form of a list of instances, verifying the correctness of a justification is DP-complete and deciding on the existence of such a justification is Σ_2^{P} -complete. For the case where the normative principles are given succinctly, deciding whether a justification is correct is in $\mathsf{NEXP} \wedge \mathsf{coNEXP}$, and NEXP-hard, and deciding whether a justification exists is in $\mathsf{EXP}^{\mathsf{NP}}$ and is NEXP-hard.

1 Introduction

In a recent paper, Boixel and Endriss (2020) developed a general notion of *justification* for collective decisions, and applied it to the setting of justifying outcomes in a voting scenario. Their approach, grounded in the axiomatic method of social choice theory, aims at providing non-expert agents with easily understandable arguments for showing that a target outcome is a reasonable compromise in a given situation. More precisely, a justification consists of (concrete instances of) normative principles—so-called *axioms* in social choice theory—showing *that* and *how* the selection of the target outcome follows from these principles in a specific situation.

While they propose a straightforward—and computationally demanding—algorithm to find such justifications, the precise computational complexity of their approach has not been studied. In this paper, we will do exactly this.

Our aim is to give insights into the potential and limits of algorithms for the approach of Boixel and Endriss (2020). In particular, we hope to lay the foundations for identifying in what (restricted) settings their approach can work efficiently in practice. Boixel and Endriss operationalise their approach by encoding axioms into a constraint modeling language (with potentially an exponential blow-up in the encoding), and then using constraint programming algorithms (for deciding satisfiability and for finding minimally unsatisfiable subsets) to find justifications. That is, they essentially use an exponential-time algorithm with an NP oracle. Our results will show that—in the general case—one cannot do significantly better than this, as the problem of deciding on the existence of a justification is NEXP-hard.

Concretely, we will study two computational tasks related to the setting of automatically justifying outcomes of a voting scenario. The first task is that of verifying whether a given justification is correct (for a given voting scenario). Boixel and Endriss defined various technical requirements that justifications should meet—we will define these requirements in Section 2. Another important condition for a justification to be convincing to non-expert agents is that its correctness can be checked efficiently. This indicates the importance of studying this first computational task. The second task that we study is that of identifying whether a correct justification exists, given a particular voting scenario. Clearly, this computational problem is central to the approach of using automated reasoning to find justifications.

Related work. There has only been little work on the notion of generating justifications for collective decisions. Cailloux and Endriss (2016) proposed an algorithm for finding a justification of a given outcome in the case where that outcome is the one given by the Borda rule. The notion they developed can be seen as a restriction of the one proposed by Boixel and Endriss (2020) and to date, the computational complexity of their approach is unknown. However,

in an unpublished work, Peters et al. (2020) prove that any outcome given by the Borda rule can be justified—using a characterisation of the Borda rule different from the classical one (Young 1974)—in $\mathcal{O}(m^2)$ steps where m is the number of alternatives. In the related field of multi-criteria decision making, Belahcene et al. (2018) seek to justify outcomes in a noncompensatory sorting model. They show that the problem of deciding whether an outcome can be represented in this model is NP-hard; the complexity of deciding whether an outcome is necessary, however, remains unknown. Belahcene et al. (2019) propose to justify outcomes obtained by aggregating pairwise comparative statements by generating arguments based on cancellation principles. Within this framework, finding a minimal justification is NP-complete.

As our aim in this paper is to use the toolbox of computational complexity to address the practical feasibility of the problem of finding justifications, our work may be seen as one particular instance of the application of tools developed in computer science to analyse collective decision making mechanisms (see, e.g., Brandt et al., 2016a). Computational complexity, in particular, has been successfully used to tackle several types of problems in social choice theory. Examples include studying the hardness of computing an outcome given a specific rule (see, e.g., Davenport and Kalagnanam, 2004; Hemaspaandra, Spakowski, and Vogel, 2005) to understand whether a given rule can be used in practice. Another successful application was to use complexity hardness results as a barrier against manipulation (see, e.g., Conitzer and Walsh, 2016).

Contribution. After defining a concrete logical language in which to encode classical normative principles from social choice theory, we prove four results. (1) We show that the problem of verifying the correctness of a justification is DP-complete for the specific case where one has immediate access to all concrete instances of the normative principles considered. (2) Under the same assumption, we show that the problem of deciding whether there exists a justification is Σ_2^{P}-complete. (3) We show that in the case where one is only given a succinct representation of the normative principles, verifying the correctness of a justification is in the class $\mathsf{NEXP} \wedge \mathsf{coNEXP} = \{\, A \cap B \mid A \in \mathsf{NEXP}, B \in \mathsf{coNEXP} \,\}$ and is NEXP-hard. Finally, (4) we show that in the setting where the principles are represented succinctly, deciding whether a justification exists is in $\mathsf{EXP}^{\mathsf{NP}}$ and NEXP-hard. An overview of our results can be found in Table 1.

The definition of justifications—as we will describe in detail in Section 2—contains both an existential and a universal quantification. Interestingly, the combination of these two quantifications manifests in different ways in the complexity results that we establish for the different computational problems (e.g., DP vs. Σ_2^{P})—see, e.g., Table 1.

Interestingly, the NEXP-hardness results that we establish indicate that the approach taken by Boixel and Endriss (2020) to automate the search for justifications—an exponential-time reduction to NP machinery—cannot be fundamentally improved, in the general case.

	CHECK-JUST	EXISTS-JUST
Quantifier-free fragment	DP-complete (Thm 3)	Σ_2^{P}-complete (Thm 6)
Full first-order language	in $\mathsf{NEXP} \wedge \mathsf{coNEXP}$ (Prop 4) NEXP-hard (Thm 5)	in $\mathsf{EXP}^{\mathsf{NP}}$ (Prop 7) NEXP-hard (Thm 8)

Table 1: Overview of the computational complexity results.

Paper outline. We start in Section 2 by recalling the definition of a justification as defined by Boixel and Endriss (2020). In the same section, we define a concrete logical language in which classical axioms from social choice theory can be encoded—and we identify two variants of this language (the quantifier-free fragment and the full language). We study the computational complexity of verifying the correctness of a justification in Section 3—for both variants of the language. In Section 4, we study the complexity of deciding whether a justification exists for a given scenario—again, for both variants of the language. In Section 5, we discuss what types of restrictions on the language one could place to yield tractability results.

2 Model

In this section, we start by recalling the language-independent notion of *justification* defined by Boixel and Endriss (2020)—as well as several relevant concepts from social choice theory. We then make this notion concrete by defining a many-sorted first-order logic language in which normative principles and voting rules can be encoded—allowing us to formally address computational complexity questions related to the notion of justifications.

What is a Justification?

Consider a group of agents reporting individual preferences over some alternatives. Given these preferences, they seek to reach a compromise and take a collective decision. Several options are then available to them. They could decide, for example, to use a specific voting rule to aggregate their personal preferences into a collective one. Another possibility would be to go through a deliberation process and examine the pros and cons of the various alternatives. It would be useful to obtain a formal justification for why a given outcome is reasonable for their situation. If a voting rule has been used, a step-by-step explanation of the outcome obtained, using arguments based on the normative principles characterising it (or any other appealing principles) might be useful. Such a justification will help to demystify the—seemingly—arbitrary result returned by the rule, hence improving the agents' confidence in the election. Alternatively, if the agents wish to deliberate, providing them with different justifications based on different desirable normative principles showing why some outcomes might be better than others could guide them through the process. Let us now more formally state what constitutes a justification.

Consider a scenario in which a set N^* of n *agents* report their individual *preferences* over a set X of m *alterna-*

tives. Their preferences are linear orders, elements of $\mathcal{L}(X)$. A *profile* of preferences $\succ_N$ for an *electorate* $N \subseteq N^*$ maps every agent $i \in N$ to an element $\succ_i \in \mathcal{L}(X)$. The set of all possible profiles for all nonempty electorates $\bigcup_{N \in 2^{N^*} \setminus \{\emptyset\}} \mathcal{L}(X)^N$ is denoted by $\mathcal{L}(X)^+$.

Now, a *voting rule* $F : \mathcal{L}(X)^+ \to 2^X \setminus \{\emptyset\}$ maps every possible profile to a nonempty winning set of alternatives. Every single voting rule does satisfy some normative properties, called *axioms* but violates others. The Borda rule for example is the unique voting rule satisfying all the following axioms: NEUTRALITY, REINFORCEMENT, FAITHFULNESS and CANCELLATION (Young 1974).[1] Whatever the language used to express an axiom A, its interpretation, the set of all voting rules satisfying this axiom, is denoted by $\mathbb{I}(A) \subseteq \mathcal{L}(X)^+ \to 2^X \setminus \{\emptyset\}$. This notion naturally extends to sets of axioms.

The following notion of justification relies on the concept of *instances* of an axiom—that we define for our language in this Section. An instance of an axiom encodes the restriction(s) it imposes for a concrete situation: restrictions on possible outcomes for a concrete profile for example. We write $A' \triangleleft A$ to express that axiom A' is an instance of axiom A. This notation extends to sets of axioms.

Definition 1 (Justification of an election outcome; Boixel and Endriss, 2020)**.** *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 set of alternatives. Then we say that a pair $\langle \mathcal{A}^{\mathrm{N}}, \mathcal{A}^{\mathrm{E}} \rangle$ of sets of axioms is a **justification**, with **normative basis** $\mathcal{A}^{\mathrm{N}}$ and **explanation** $\mathcal{A}^{\mathrm{E}}$, for the set X^* winning the election under profile $\succ_{N^*}$ if and only if the following conditions are satisfied:*

- ***Explanatoriness.*** *$\mathcal{A}^{\mathrm{E}}$ (but none of its proper subsets) can explain the desired outcome: $F(\succ_{N^*}) = X^*$ for every voting rule $F \in \mathbb{I}(\mathcal{A}^{\mathrm{E}})$, but $F(\succ_{N^*}) \neq X^*$ for some voting rule $F \in \mathbb{I}(\mathcal{A})$ for every strict subset $\mathcal{A} \subsetneq \mathcal{A}^{\mathrm{E}}$.*
- ***Relevance.*** *The explanation $\mathcal{A}^{\mathrm{E}}$ is an instance of the normative basis $\mathcal{A}^{\mathrm{N}}$: $\mathcal{A}^{\mathrm{E}} \triangleleft \mathcal{A}^{\mathrm{N}}$.*
- ***Adequacy.*** *All axioms in the normative basis $\mathcal{A}^{\mathrm{N}}$ belong to the corpus $\mathbb{A}$ of axioms provided: $\mathcal{A}^{\mathrm{N}} \subseteq \mathbb{A}$.*
- ***Nontriviality.*** *There exists at least one voting rule that satisfies all axioms in the normative basis: $\mathbb{I}(\mathcal{A}^{\mathrm{N}}) \neq \emptyset$.*

Out of the four requirements defined above, two of them—*relevance* and *adequacy*—seem pretty easy to satisfy in practice. Adequacy is easily satisfied as long as the algorithm works with the given corpus of axioms; relevance is easily satisfied as long as one keeps track of which axiom in the corpus gave rise to which concrete axiom instance.

The other two requirements are more interesting and will be the hardest to satisfy, thus making the task of generating justifications computationally hard. The formulation of the explanatoriness and nontriviality requirements can give us some intuition regarding how hard to satisfy those requirements are. The explanatoriness condition requires us to check that all voting rules satisfying the axiom instances in the explanation do select the target outcome under the target profile. We also need to make sure that the explanation cannot be any smaller. The nontriviality condition on the other hand requires us to search for a voting rule that satisfies all the axioms in the normative basis.

[1] For space reasons, we omit a description of these axioms. For more details, we refer to the literature (e.g., Zwicker, 2016).

A Language for Axioms and Justifications

To precisely study the computational complexity of several problems related to justifications, we will define a logical language in which to encode axioms and voting rules. For this, we will use a *many-sorted first-order* language with three sorts: one for the agents, one for the alternatives, and one for the profiles. The language we define here is highly inspired by the one used by (Endriss 2020) to model and analyse axiomatic properties of matching mechanisms.

Let Var_N, Var_X and Var_P be three disjoint, infinite and countable sets of variables. The following specification—in Backus-Naur form—defines the set of all sentences φ expressible in this language. In this specification, i denotes a variable in Var_N, x and y variables in Var_X and p a variable in Var_P. Additionally, we also assume that there is a sufficiently large number of constants in set $D_{n,m}$ to represent any concrete object (agents, alternatives and profile) in a given situation involving n agents and m alternatives. In the specification of the language, i_ℓ denotes a constant referring to an agent, x_ℓ denotes a constant referring to an alternative, and p_ℓ denotes a constant referring to a profile.

$$
\begin{aligned}
\varphi ::= {} & \forall_N i.\varphi \mid \forall_X x.\varphi \mid \forall_P p.\varphi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \\
& \mathit{pref}(t_i, t_x, t_x, t_p) \mid o(t_x, t_p) \\
t_i ::= {} & i_\ell \mid i \\
t_x ::= {} & x_\ell \mid x \\
t_p ::= {} & p_\ell \mid p
\end{aligned}
$$

We also consider the usual abbreviations—e.g., $\varphi_1 \vee \varphi_2$ for $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$. The language that we consider consists of all *sentences* that are generated by the above specification.

The interpretation of sentences in our language depends on the number n of agents involved in a specific scenario and on the number m of alternatives considered in this scenario. Variables in Var_N range over the set $\{i_1, \ldots, i_n\}$ of agents, variables in Var_X range over the set $\{x_1, \ldots, x_m\}$ of alternatives, and variables in Var_P range over the set $\{p_1, \ldots, p_l\}$ of profiles where $l = \sum_{k=1}^{n} \binom{n}{k} m!^k$ is the total number of profiles involving only—some of the—agents in N^*.

An *assignment* is a function $\alpha : \mathrm{Var}_N \cup \mathrm{Var}_X \cup \mathrm{Var}_P \cup D_{n,m} \to D_{n,m}$, where $D_{n,m} = \{i_1, \ldots, i_n\} \cup \{x_1, \ldots, x_m\} \cup \{p_1, \ldots, p_l\}$ is the combined set of agents, alternatives and profiles involved in a specific scenario. We require that $\alpha(i) \in \{i_1, \ldots, i_n\}$ for all $i \in \mathrm{Var}_N$, that $\alpha(x) \in \{x_1, \ldots, x_m\}$ for all $x \in \mathrm{Var}_X$, and that $\alpha(p) \in \{p_1, \ldots, p_l\}$ for all $p \in \mathrm{Var}_P$. Moreover, we require that for every constant $c \in D_{n,m}$ it holds that $\alpha(c) = c$. For any $x \in \mathrm{Var}_N \cup \mathrm{Var}_X \cup \mathrm{Var}_P$, assignments α and α' are called x-variants of each other if $\alpha(y) = \alpha'(y)$ for every variable $y \in \mathrm{Var}_N \cup \mathrm{Var}_X \cup \mathrm{Var}_P \setminus \{x\}$.

We write $f, \alpha \vDash \varphi$ to indicate that a voting rule f *satisfies sentence φ under assignment α*. This notion of satisfaction is defined inductively as follows:

- $f, \alpha \vDash \mathit{pref}(i, x, y, p)$ if agent $\alpha(i)$ ranks alternative $\alpha(x)$ above $\alpha(y)$ in profile $\alpha(p)$;
- $f, \alpha \vDash o(x, p)$ if alternative $\alpha(x)$ is selected as one of the winners according to rule f—that is, $\alpha(x) \in f(\alpha(p))$;
- $f, \alpha \vDash \neg\varphi$ if $f, \alpha \vDash \varphi$ is not the case;
- $f, \alpha \vDash \varphi \wedge \psi$ if both $f, \alpha \vDash \varphi$ and $f, \alpha \vDash \psi$ are the case;
- $f, \alpha \vDash \forall_N i.\varphi$ if $f, \alpha' \vDash \varphi$ for all i-variants α' of α;
- $f, \alpha \vDash \forall_X x.\varphi$ if $f, \alpha' \vDash \varphi$ for all x-variants α' of α; and
- $f, \alpha \vDash \forall_P p.\varphi$ if $f, \alpha' \vDash \varphi$ for all p-variants α' of α.

Furthermore, we will also make use of two other (shorthand) predicates. We let $\mathit{top}(x, i, p) = \forall_X y, \mathit{pref}(i, x, y, p)$, which is true if and only if $\alpha(x)$ is the most preferred alternative of $\alpha(i)$ in profile $\alpha(p)$. Moreover, we let $\omega(X^*, p) = \bigwedge_{x \in X^*} o(x, p) \wedge \bigwedge_{x \notin X^*} \neg o(x, p)$ for any $X^* \subseteq \{x_1, \ldots, x_m\}$, which states that X^* is the entire outcome selected under profile $\alpha(p)$.

Instances In the concept of justifications—as defined by Boixel and Endriss (2020) and previously described in this Section—normative bases and explanations are expressed in the same language, and the set $\mathcal{A}^{\text{E}}$ of sentences that forms an explanation is required to be an instance of the normative basis $\mathcal{A}^{\text{N}}$. Instances are the building blocks of axioms; an axiom can be seen as a collection of concrete restrictions imposed on some concrete objects (profiles, agents, etc.), these restrictions are the instances of the axiom considered. We will define what it means for a sentence (and a set of sentences) expressed in our first-order language to be an instance of another—with respect to a given scenario with n agents and m alternatives.

An *instance* is a sentence without any (universal) quantifiers. Moreover, we say that a sentence φ_1 is an *instance* of a sentence φ_2, written $\varphi_1 \lhd \varphi_2$, if φ_1 can be obtained from φ_2 by iterated application of the following steps: (i) removing an outermost quantifier $\forall_N i$, $\forall_X x$, or $\forall_P p$, and replacing every occurrence of the corresponding variable i, x, or p in the rest of the sentence by a single constant i_ℓ, x_ℓ, or p_ℓ (respectively); and (ii) replacing a proper subformula[2]:
(ii.a) of the form $\forall_N i.\psi$ by $\bigwedge_{1 \leq \ell \leq n} \psi[i \mapsto i_\ell]$;
(ii.b) of the form $\forall_X x.\psi$ by $\bigwedge_{1 \leq \ell \leq m} \psi[x \mapsto x_\ell]$;
(ii.c) of the form $\forall_P p.\psi$ by $\bigwedge_{1 \leq \ell \leq \sum_{k=1}^{n} \binom{n}{k} m!^k} \psi[p \mapsto p_\ell]$.

Generally axioms have the following shape: "FORALL *Object*, IF *Condition* is met, THEN *Restriction* holds for *Object*." For a given axiom, we seek to generate one concrete instance of it for each concrete *Object*, in order to only include the relevant instances in a justification. This way, justifications only feature the relevant concrete instances, rather than a sentence specifying the general axiom—the latter of which would be the case if we were to replace the outermost-most $\forall$ quantifiers with large conjunctions.

A set $\mathcal{A}^{\text{E}}$ of sentences is an *instance* of a set $\mathcal{A}^{\text{N}}$ of sentences, written $\mathcal{A}^{\text{E}} \lhd \mathcal{A}^{\text{N}}$, if for every $\varphi_1 \in \mathcal{A}^{\text{E}}$, there exists some $\varphi_2 \in \mathcal{A}^{\text{N}}$ such that $\varphi_1 \lhd \varphi_2$.

[2]With a proper subformula of φ, we mean any subformula that is not identical to the entire formula φ.

Note that this notion of instancehood satisfies the three requirements stipulated by Boixel and Endriss (2020): (1) Every instance of a sentence is itself a sentence. (2) A sentence is equivalent to the conjunction of all of its instances. (3) The number of instances of a sentence is finite.

Variants of the Language In the remainder of the paper, we will direct our attention to two variants of this language:

- *The full language,* where we allow all sentences in the first-order language that we defined. Intuitively, this language allows axioms to be expressed in a succinct way, where one does not need to explicitely list all instances of the axiom that refer to concrete profiles.
- *The quantifier-free fragment,* where we only allow sentences without quantifiers. Intuitively, this language requires one to list all relevant applications of the axioms to concrete objects (profiles, etc.).

Expressing Axioms The language that we defined is expressive enough to express a wide range of well-known axioms that have been studied in the social choice literature (see, e.g., (Brandt et al. 2016b; Zwicker 2016)). To illustrate this, we express the axiom of Pareto optimality.

Example 2 (Pareto optimality)**.** *A voting rule F is* Pareto optimal *if for all profiles $\succ_N$ the following is true: for any two alternatives $x, y \in X$ if it holds that each agent $i \in N$ prefers x over y, then $y \notin F(\succ_N)$. This property can be expressed in the first-order language as follows:* $\forall_P p.\forall_X x.\forall_X y.(\forall_N i.\ \mathit{pref}(i, x, y, p)) \rightarrow \neg o(y, p)$.

The following is an example of an instance of this sentence—given a voting scenario with n agents and m alternatives: $(\bigwedge_{1 \leq k \leq n} \mathit{pref}(i_k, a, b, p_1)) \rightarrow \neg o(b, p_1)$. *Here, the restriction is enforced on the concrete profile p_1 with respect to the concrete alternatives a and b.*

3 Checking Correctness of a Justification

Now that we have defined a concrete language in which to express axioms and their instances, we can use the toolbox of computational complexity in order to gain insights about the obstacles that may prevent us from reaching practical feasibility and the possible ways out.

We start by taking a look at the problem of deciding whether a given justification is correct. That is, we consider the following decision problem.

CHECK-JUST
INSTANCE: A voting scenario $\langle \succ_{N^*}, X^*, \mathbb{A} \rangle$ and a pair $\langle \mathcal{A}^{\text{N}}, \mathcal{A}^{\text{E}} \rangle$.
QUESTION: Is $\langle \mathcal{A}^{\text{N}}, \mathcal{A}^{\text{E}} \rangle$ a justification for the set X^* winning the election under profile $\succ_{N^*}$?

Note that even if the potential justification is given (and so is the corresponding normative basis), we still require the corpus of axioms $\mathbb{A}$ to be given as well. Having $\mathbb{A}$ is necessary to check the adequacy requirement (which is a computationally easy property to check).

The problem of checking whether a given justification is correct is relevant if we wish to automatically use the ap-

proach in practice. Indeed, to improve confidence in the system generating justifications, one would like to be able to easily verify that the system's output is correct. Any computational intractability result for the problem of checking a justification would be an obstacle for experts to verify the output of the system—even with powerful computational machinery—making it even harder for non-expert individuals to verify whether a justification is correct.

In the remainder of this section, we study the computational complexity of the problem CHECK-JUST when considering two different variants of our language in which to express axioms and their instances: the quantifier-free fragment and the full variant of our language.

The Quantifier-Free Fragment

We first consider the case where the normative principles in $\mathbb{A}$ can only be expressed using quantifier-free sentences. Thus each axiom can be directly seen as an instance of a more general one as it can only refer to concrete objects (agents, alternatives, and profiles). In this scenario axioms consist of the conjunction of their concrete instances. Encoding axioms in this way is useful in practice as it enables the easy use of powerful tools such as SAT solvers.[3]

We show that the problem CHECK-JUST is DP-complete in the case where the axioms in the corpus $\mathbb{A}$ are given as quantifier-free sentences. The complexity class $\mathsf{DP} = \mathsf{NP} \wedge \mathsf{coNP}$[4] lies at the second level of the Boolean Hierarchy (Papadimitriou and Yannakakis 1982; Cai et al. 1988).

Theorem 3. *In the case where the axioms in the corpus $\mathbb{A}$ are given as quantifier-free sentences, the problem* CHECK-JUST *is* DP*-complete.*

Proof (sketch). We can obtain a DP algorithm by adapting the Σ_2^{P} algorithm in the proof of Theorem 6 (in Section 4) to the problem CHECK-JUST: instead of guessing a justification, we use the (candidate) justification given in the input. If we do this, the single NP oracle in the algorithm call does not depend on any guesses—yielding a DP algorithm.

To show DP-hardness, we describe (the general lines of) a reduction from the DP-complete problem CRITICAL-SAT (Papadimitriou and Wolfe 1986). In this problem, the input is a propositional formula $\varphi = \bigwedge_{i=1}^{m} c_i$ in 3CNF with m clauses and n variables, and the question is to decide whether φ is minimally unsatisfiable—that is, whether it holds that (1) φ is unsatisfiable and (2) removing any clause from φ yields a satisfiable formula.

Given φ, we construct an equivalent instance of CHECK-JUST as follows. We let $X = \{\top, \bot, \triangle\}$ be the set of alternatives. To each variable $v \in \mathit{Vars}(\varphi)$, we associate a profile $\succ_v$, and we let $\succ_{N^*} = \succ_{v_1}$. We take $X^* = \{\triangle\}$ to be the target outcome. We let $\mathcal{A}^{\mathrm{N}} = \mathcal{A}^{\mathrm{E}} = \{\, A_{c_i} \mid c_i \in \varphi \,\} \cup \{A_\triangle\}$—where the axioms A_{c_i} and $A_\triangle$ are defined as follows. Intuitively, the axioms A_{c_i} encode the clauses of φ, and the axiom $A_\triangle$ enforces that either (i) all profiles $\succ_v$ are assigned to some truth value—corresponding to a truth assignment for φ—or (ii) all profiles are assigned to the outcome $\{\triangle\}$. For each clause $c_i = (\ell_1 \vee \ell_2 \vee \ell_3)$ of φ, we let $A_{c_i} = \sigma(\ell_1) \vee \sigma(\ell_2) \vee \sigma(\ell_3)$, where $\sigma(\ell) = \omega(\{\top\}, \succ_v) \vee \omega(\{\triangle\}, \succ_v)$ if $\ell = v$ is a positive literal, and $\sigma(\ell) = \omega(\{\bot\}, \succ_v) \vee \omega(\{\triangle\}, \succ_v)$ if $\ell = \neg v$ is a negative literal. We let $A_\triangle = (\bigwedge_{v \in \mathit{Vars}(\varphi)} \omega(\{\triangle\}, \succ_v)) \vee (\bigwedge_{v \in \mathit{Vars}(\varphi)} (\omega(\{\top\}, \succ_v) \vee \omega(\{\bot\}, \succ_v)))$.

One can verify that the constructed instance satisfies the explanatoriness condition if and only if φ is minimally unsatisfiable—and the instance always satisfies the other three conditions of a justification. □

The Full Language

We continue with studying the complexity of CHECK-JUST for the full variant of our language. We begin by showing that the problem CHECK-JUST is contained in an exponential-time analogue $\mathsf{NEXP} \wedge \mathsf{coNEXP}$ of the class DP.

Proposition 4. *In the case where the axioms in the corpus $\mathbb{A}$ are given as arbitrary sentences in the first-order language, the problem* CHECK-JUST *is in* $\mathsf{NEXP} \wedge \mathsf{coNEXP}$.

Proof (idea). The proof is entirely analogous to the DP membership proof for the case of quantifier-free sentences (Theorem 3). When the axioms are given as quantifier-free sentences, guessing a voting rule F consists of a polynomial number of guesses (one for each mentioned profile). Therefore, checking whether there exists some $F \in \mathbb{I}(\mathcal{A})$ with certain properties, for some set $\mathcal{A}$ of sentences, is an NP problem. In the case where the axioms are given as arbitrary sentences, the number of relevant profiles that need to be considered can be exponential. Therefore, checking whether exists some $F \in \mathbb{I}(\mathcal{A})$ with certain properties in this case is a NEXP problem. Similarly, the coNP problem in the case of quantifier-free sentences becomes a coNEXP problem for the full variant of our language. □

In the case where axioms are specified as arbitrary first-order sentences, CHECK-JUST is NEXP-hard. For space reasons, we omit a proof of this result—one can show this by straightforwardly modifying the proof of Theorem 8 that we will establish in Section 4.

Theorem 5. *In the case where the axioms in the corpus $\mathbb{A}$ are given as arbitrary sentences in the first-order language, the problem* CHECK-JUST *is* NEXP*-hard.*

4 Existence of a Justification

We continue by studying the problem of deciding whether a justification exists for a given scenario.

EXISTS-JUST

INSTANCE: A voting scenario $\langle \succ_{N^*}, X^*, \mathbb{A} \rangle$.

QUESTION: Does there exist a justification $\langle \mathcal{A}^{\mathrm{N}}, \mathcal{A}^{\mathrm{E}} \rangle$ for $\langle \succ_{N^*}, X^*, \mathbb{A} \rangle$ such that $\mathcal{A}^{\mathrm{N}} \subseteq \mathbb{A}$, $\mathcal{A}^{\mathrm{E}} \triangleleft \mathcal{A}^{\mathrm{N}}$, $\mathcal{A}^{\mathrm{N}}$ is *non-trivial*, and $\mathcal{A}^{\mathrm{E}}$ satisfies the *explanatoriness* condition?

This problem captures the question of whether there exists

[3] See the original paper of Tang and Lin (2009) for more information on how to encode axioms in propositional logic.

[4] For any two classes C and D the class $C \wedge D$ is defined as $\{\, A \cap B \mid A \in C, B \in D \,\}$. Note that $C \wedge D$ differs from $C \cap D$.

a justification for a given input. The lower bounds that we will establish for the complexity of this decision problem will carry over to the more general problem of computing a justification $\langle \mathcal{A}^{\mathrm{N}}, \mathcal{A}^{\mathrm{E}} \rangle$, if it exists.

The Quantifier-Free Fragment

We begin by showing that the problem EXISTS-JUST is Σ_2^{P}-complete in the case where the axioms in the corpus $\mathbb{A}$ are given as quantifier-free sentences.

Theorem 6. *In the case where the axioms in the corpus $\mathbb{A}$ are given as quantifier-free sentences, the problem* EXISTS-JUST *is Σ_2^{P}-complete.*

Proof (sketch). We begin by showing membership in Σ_2^{P}. We describe a nondeterministic polynomial-time algorithm with access to an NP oracle that decides the problem. The algorithm first guesses a normative basis $\mathcal{A}^{\mathrm{N}} \subseteq \mathbb{A}$, together with a voting rule $F \in \mathbb{I}(\mathcal{A}^{\mathrm{N}})$ that witnesses that $\mathcal{A}^{\mathrm{N}}$ is nontrivial. Since there are only polynomially many different profiles that are relevant, guessing the rule F can be done in nondeterministic polynomial time. Moreover, the algorithm guesses a set $\mathcal{A}^{\mathrm{E}}$ and verifies that $\mathcal{A}^{\mathrm{E}} \lhd \mathcal{A}^{\mathrm{N}}$. Since $\mathcal{A}^{\mathrm{N}}$ contains no quantifiers, guessing $\mathcal{A}^{\mathrm{E}}$ can also be done in nondeterministic polynomial time. Then, for each $\mathcal{A} \subsetneq \mathcal{A}^{\mathrm{E}}$ with $|\mathcal{A}| = |\mathcal{A}^{\mathrm{E}}| - 1$ the algorithm guesses a voting rule $F \in \mathbb{I}(\mathcal{A})$ with $F(\succ_{N^*}) \neq X^*$. Finally, the algorithm uses the NP oracle to verify that there exists no $F \in \mathbb{I}(\mathcal{A}^{\mathrm{E}})$ with $F(\succ_{N^*}) \neq X^*$—that is, that for all $F \in \mathbb{I}(\mathcal{A}^{\mathrm{E}})$ it holds that $F(\succ_{N^*}) = X^*$. The algorithm accepts if and only if all checks succeed. It is straightforward to verify that the algorithm accepts for some sequence of nondeterministic choices if and only if there exists a suitable justification $\langle \mathcal{A}^{\mathrm{N}}, \mathcal{A}^{\mathrm{E}} \rangle$.

Next, we will show Σ_2^{P}-hardness by reducing from the Σ_2^{P}-complete problem $\exists\forall$-3SAT (Stockmeyer 1976; Wrathall 1976). In this problem, the input is a propositional formula $\varphi(A, B)$ in 3DNF ranging over two distinct sets of variables A and B, and the question is whether there exists a truth assignment $\alpha : A \to \{0, 1\}$ such that for all truth assignments $\beta : B \to \{0, 1\}$ it holds that $\varphi[\alpha \cup \beta]$ is true.

Let $\varphi(A, B)$ be an instance of $\exists\forall$-3SAT. We consider a sufficiently large set of agents, and we let $X = \{\top, \bot, \triangle\}$ be the set of alternatives. We associate each propositional variable $v \in \mathit{Vars}(\varphi)$ with a profile $\succ_v$. Moreover, we let $X^* = \{\triangle\}$ and $\succ_{N^*} = \succ_{v_1}$ where v_1 corresponds to a variable in the set B.

As set of (quantifier-free) axioms we take $\mathbb{A} = A^+ \cup A^- \cup \{A_\triangle\}$, where $A_\triangle$ is defined below, where $A^+ = \{A_v^+ \mid v \in A\}$ contains an axiom $A_v^+ = \omega(\{\top\}, \succ_v)$ for each profile $\succ_v$ linked to some $v \in A$, encoding the fact that the truth value $\top$ should be assigned to v, and where $A^- = \{A_v^- \mid v \in A\}$ contains an axiom $A_v^- = \omega(\{\bot\}, \succ_v)$ for each profile $\succ_v$ linked to some $v \in A$, encoding the fact that the truth value $\bot$ should be assigned to v. Note that if we are able to assign an outcome to each profile $\succ_v$ with $v \in A$, satisfying either A_v^+ or A_v^-, then this value assignment corresponds to a truth assignment $\alpha : A \to \{0, 1\}$ where for all v in A, $\alpha(v) = 1$ if and only if $\omega(\{\top\}, \succ_v)$ is true.

Intuitively, the axiom $A_\triangle$ encodes that, given a truth assignment $\alpha : A \to \{0, 1\}$, then either there exists a truth assignment $\beta : B \to \{0, 1\}$ making φ false (together with α), or all profiles $\succ_v$ for variables $v \in B$ are assigned to the outcome $\{\triangle\}$. We let $A_\triangle = \big(\bigwedge_{v \in B} \omega(\{\triangle\}, \succ_v)\big) \vee \big(\neg\psi \wedge \bigwedge_{v \in B}(\omega(\{\top\}, \succ_v) \vee \omega(\{\bot\}, \succ_v))\big)$, where $\psi = \bigvee_{d_i \in \varphi, d_i = (\ell_1 \wedge \ell_2 \wedge \ell_3)} (\sigma(\ell_1) \wedge \sigma(\ell_2) \wedge \sigma(\ell_3))$, $\sigma(\ell) = \omega(\{\top\}, \succ_v)$ if $\ell = v$ is a positive literal, and $\sigma(\ell) = \omega(\{\bot\}, \succ_v)$ if $\ell = \neg v$ is a negative literal.

One can verify that there exists some $\mathcal{A}^{\mathrm{N}} \subseteq \mathbb{A}$ and $\mathcal{A}^{\mathrm{E}} \lhd \mathcal{A}^{\mathrm{N}}$ (justifying the fact the $\{\triangle\}$ is the only possible outcome for profile $\succ_{v_1}$) such that (a) $\mathcal{A}^{\mathrm{N}}$ satisfies the nontriviality condition and (b) $\mathcal{A}^{\mathrm{E}}$ satisfies the explanatoriness condition if and only if (a) there exists some suitable α such that (b) for all β it holds that $\varphi[\alpha \cup \beta]$ is true. □

The Full Language

For the case where axioms are specified as arbitrary sentences in our language, the problem EXISTS-JUST is in $\mathsf{EXP}^{\mathsf{NP}}$ and NEXP-hard. We begin by showing membership in $\mathsf{EXP}^{\mathsf{NP}}$. The complexity class $\mathsf{EXP}^{\mathsf{NP}}$ has also been named Δ_2^{exp}—at the second level of the Exponential-Time Hierarchy as defined by Dawar, Gottlob, and Hella (1998)—and is an exponential-time analogue of the class $\Delta_2^{\mathrm{p}} = \mathsf{P}^{\mathsf{NP}}$.

Proposition 7. *In the case where the axioms in the corpus $\mathbb{A}$ are given as arbitrary sentences in the first-order language, the problem* EXISTS-JUST *is in* $\mathsf{EXP}^{\mathsf{NP}}$.

Proof (sketch). We describe an exponential-time algorithm with an NP oracle that solves the problem. The algorithm iterates over all (exponentially many) sets $\mathcal{A}^{\mathrm{N}} \subseteq \mathbb{A}$. For each $\mathcal{A}^{\mathrm{N}}$, it writes out $\mathcal{A}^{\mathrm{N}}$ as an equivalent quantifier-free sentence $\mathcal{A}^{\mathrm{N}}_{\mathrm{qf}}$. Then, it checks (1) whether $\mathcal{A}^{\mathrm{N}}_{\mathrm{qf}}$ is nontrivial. This check can be done with a single (exponential-size) query to the NP oracle. It further checks (2) that every rule $F \in \mathbb{I}(\mathcal{A}^{\mathrm{N}}_{\mathrm{qf}})$ has the property that $F(\succ_{N^*}) = X^*$. This can also be done with a single (exponential-size) query to the NP oracle: the algorithm asks the oracle if there exists some $F \in \mathbb{I}(\mathcal{A}^{\mathrm{N}}_{\mathrm{qf}})$ such that $F(\succ_{N^*}) \neq X^*$. The algorithm outputs "yes" if and only if for some $\mathcal{A}^{\mathrm{N}} \subseteq \mathbb{A}$ both checks (1) and (2) succeed. This algorithm is correct because if $\mathcal{A}^{\mathrm{N}}$ is *nontrivial* and forces the outcome to be selected, then there must exist some $\mathcal{A}^{\mathrm{E}} \lhd \mathcal{A}^{\mathrm{N}}$ (and thus also some subset-minimal $\mathcal{A}^{\mathrm{E}}$) that satisfies *explanatoriness*. □

The algorithm described in the proof of Proposition 7 is in line with the approach taken by Boixel and Endriss (2020), where they encode the problem of finding a justification into a constraint satisfaction problem in exponential time, and then use the NP machinery of constraint satisfaction.

We now turn our attention to the NEXP-hardness of the problem. To show that the problem EXISTS-JUST is NEXP-hard, we will reduce from the NEXP-complete problem SUCCINCT-3COL (Veith 1997). Here, the problem is to decide if a given undirected graph $G = (V, E)$ is 3-colorable—that is, whether there exists a function $\chi : V \to \{1, 2, 3\}$ such that for each $\{v, v'\} \in E$ it holds that $\chi(v) \neq \chi(v')$. However, the graph is specified by a representation that can be exponentially more succinct. We assume that $|V| = 2^n$ for some $n \in \mathbb{N}$ and that $V =$

$\{0,1\}^n$. The graph G is then given by a propositional logic formula φ_G on the variables $x_1, \ldots, x_n, y_1, \ldots, y_n$ that specifies which vertices are connected by an edge in E in the following way. Let $v, v' \in V$ be two vertices. Then v, v' are connected by an edge if $\varphi_G(\alpha_v, \alpha_{v'})$ evaluates to true, where α_v is the truth assignment that instantiates the variables $x_1, \ldots, x_n$ according to $v \in \{0,1\}^n$ and $\alpha_{v'}$ is the truth assignment that instantiates the variables $y_1, \ldots, y_n$ according to $v' \in \{0,1\}^n$. The input for the problem SUCCINCT-3COL consists of a propositional formula φ_G on the variables $x_1, \ldots, x_n, y_1, \ldots, y_n$ that specifies a graph $G = (V, E)$ with 2^n vertices, and the question is whether G is 3-colorable.

Theorem 8. *In the case where the axioms in the corpus $\mathbb{A}$ are given as arbitrary sentences in the first-order language, the problem* EXISTS-JUST *is* NEXP*-hard.*

Proof. We show NEXP-hardness by giving a reduction from SUCCINCT-3COL. Take an instance of SUCCINCT-3COL, consisting of a graph G with 2^n vertices represented succinctly by φ_G. We construct a corresponding instance $\langle \succ_{N^*}, X^*, \mathbb{A} \rangle$ of EXISTS-JUST as follows.

We introduce n individuals, and the set $X = \{r, g, b\}$ of alternatives. This gives rise to 6^n possible preference profiles. We will be interested in those profiles where every agents ranks alternative r as their most preferred alternative—the 2^n profiles with this property we will call *interesting* profiles. We associate each of the 2^n vertex in G with one of these interesting profiles. As target profile $\succ_{N^*}$ we take one of the non-interesting profiles, and the target outcome is $X^* = \{r\}$. The set $\mathbb{A} = \{A\}$ contains a single axiom $A = (\forall_P p. \forall_P p_1. \forall_P p_2. \forall_N i.\ A_1 \wedge A_2) \wedge \omega(\{r\}, \succ_{N^*})$, where the subformulas A_1 and A_2 are defined below.

The subformula A_1 enforces that the outcome given to each profile consists of a singleton. The intuition behind this is that each vertex of G can only be assigned to a single color. We let $A_1 = \omega(\{r\}, p) \vee \omega(\{g\}, p) \vee \omega(\{b\}, p)$.

The subformula A_2 encodes that two adjacent vertices in G cannot be associated with the same color—and thus the two corresponding interesting profiles cannot be assigned the same outcome: We let $A_2 = \mathit{top}(r, i, p_1) \wedge \mathit{top}(r, i, p_2) \wedge A_{\varphi_G} \rightarrow \bigwedge_{c \in \{r,g,b\}} (\neg\omega(\{c\}, p_1) \vee \neg\omega(\{c\}, p_2))$, where A_{φ_G} is used to determine whether vertices in G associated to profiles p_1 and p_2 are adjacent. The subformula A_{φ_G} is obtained from φ_G by replacing each occurrence of x_i by $\mathit{pref}(i, g, b, p_1)$ and each occurrence of y_i by $\mathit{pref}(i, g, b, p_2)$. Finally, the subformula $\omega(\{r\}, \succ_{N^*})$ of A enforces that $\{r\}$ must be selected for profile $\succ_{N^*}$.

One can verify that the only relevant $\mathcal{A}^{\mathrm{N}} = \{A\} \subseteq \mathbb{A}$ satisfies the nontriviality condition if and only if G is 3-colorable, and that the explanatoriness condition can be met by an appropriate (minimal) $\mathcal{A}^{\mathrm{E}}$. □

5 Further Restrictions on the Language

The complexity results that we established in this paper all indicate computational intractability—one worse than the other. In general, when given an arbitrary set of first-order sentences that express axioms, finding (and checking) justifications for a given outcome requires nondeterministic exponential time (Theorems 5 and 8). If we restrict the language and forbid the use of quantifiers—which makes the language exponentially less succinct—the problem of finding justifications remains intractable (Theorem 6).

For using the automated search for justifications in practical settings, worst-case running time (e.g., polynomial-time) guarantees would be useful. The results in this paper are all based on arbitrary logic sentences. To get tractability results, one would have to look at various restrictions on the sets of sentences that are given as axioms. Our results already give some indications for what restrictions might (or might not) work to get tractability results. For example, for the case of quantifier-free axioms that refer to at most two profiles, we can modify the proof of Theorem 8 to show that finding justifications is still NP-hard.

Corollary 9. *In the case where each axiom in $\mathbb{A}$ is a quantifier-free sentence that mentions at most two profiles, the problem* EXISTS-JUST *is* NP*-hard.*

On the other hand, if each axiom is quantifier-free and mentions only a single profile, the problems EXISTS-JUST and CHECK-JUST are polynomial-time computable. However, this is a very limited setting.

We believe that the most interesting and promising restricted languages to consider are those that are based on structural restrictions—e.g., using various types of width measures on graphs (see, e.g., Cygan et al., 2015).

6 Conclusion

In the context of providing non-expert agents with justifications of collective decisions, we proved several intractability results. In case of succinct representations of the normative principles, checking the correctness of a given justification or deciding whether there exists one are NEXP-hard problems. In the more favorable case where one has direct access to all the instances of these normative principles, deciding on the existence of a justification is already Σ_2^{P}-complete. These results, summarised in Table 1, indicate that automatically justifying any outcome using any normative principle in real-life scenarios is not feasible across the board.

There are several directions for future research that one could explore. For example, the Σ_2^{P}-completeness result of Theorem 6 suggests that encoding the problem into Answer Set Programming (ASP) might be a useful direction. For another direction, as sketched in Section 5, it might be possible to make this notion of justification useful in practice by designing efficient solving strategies for settings with restrictions on the normative principles considered. For example, finding justifications that only use intraprofile axioms (Fishburn 1987)—although of very limited appeal—can be done tractably. A slightly more interesting restriction would be to allow axioms that refer to at most two different profiles. Many appealing axioms from social choice theory fit in this category (e.g., anonymity, neutrality, strategyproofness). In this case (possibly with some mild further restrictions), finding a justification could be in NP and finding practically efficient solving approaches for this case could be conceivable.

References

Belahcene, K.; Chevaleyre, Y.; Labreuche, C.; Maudet, N.; Mousseau, V.; and Ouerdane, W. 2018. Accountable Approval Sorting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-2018)*.

Belahcene, K.; Labreuche, C.; Maudet, N.; Mousseau, V.; and Ouerdane, W. 2019. Comparing options with argument schemes powered by cancellation. In *28th International Joint Conference on Artificial Intelligence (IJCAI-19)*, 1537–1543. Macao, Macau SAR China. doi:10.24963/ijcai.2019/213. URL https://hal.archives-ouvertes.fr/hal-02133034.

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.

Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds. 2016a. *Handbook of Computational Social Choice*. Cambridge University Press.

Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D. 2016b. Introduction to Computational Social Choice. In Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds., *Handbook of Computational Social Choice*, chapter 1. Cambridge University Press.

Cai, J.; Gundermann, T.; Hartmanis, J.; Hemachandra, L. A.; Sewelson, V.; Wagner, K. W.; and Wechsung, G. 1988. The Boolean Hierarchy I: Structural Properties. *SIAM J. Comput.* 17(6): 1232–1252.

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.

Conitzer, V.; and Walsh, T. 2016. Barriers to Manipulation in Voting. In Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds., *Handbook of Computational Social Choice*, chapter 6. Cambridge University Press.

Cygan, M.; Fomin, F. V.; Kowalik, L.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer.

Davenport, A.; and Kalagnanam, J. 2004. A computational study of the Kemeny rule for preference aggregation. In *Proceedings of the 19th National Conference on Artifical Intelligence (AAAI'04)*, volume 4, 697–702.

Dawar, A.; Gottlob, G.; and Hella, L. 1998. Capturing relativized complexity classes without order. *Mathematical Logic Quarterly* 44(1): 109–122.

Endriss, U. 2020. Analysis of One-to-One Matching Mechanisms via SAT Solving: Impossibilities for Universal Axioms. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI-2020)*.

Fishburn, P. C. 1987. *Interprofile conditions and impossibility*, volume 18. Taylor & Francis.

Hemaspaandra, E.; Spakowski, H.; and Vogel, J. 2005. The complexity of Kemeny elections. *Theoretical Computer Science* 349(3): 382–391.

Papadimitriou, C. H.; and Wolfe, D. 1986. The complexity of facets resolved. *J. of Computer and System Sciences* 37: 2–13.

Papadimitriou, C. H.; and Yannakakis, M. 1982. The complexity of facets (and some facets of complexity). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing - STOC '82*, 255–260. San Francisco, California, United States: ACM Press.

Peters, D.; Procaccia, A. D.; Psomas, A.; and Zhou, Z. 2020. Explainable voting. URL https://dominik-peters.de/publications/explainable.pdf. Unpublished manuscript.

Stockmeyer, L. J. 1976. The polynomial-time hierarchy. *Theoretical Computer Science* 3(1): 1–22.

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

Veith, H. 1997. Languages represented by Boolean formulas. *Information Processing Letters* 63(5): 251–256.

Wrathall, C. 1976. Complete Sets and the Polynomial-Time Hierarchy. *Theoretical Computer Science* 3(1): 23–33.

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. Cambridge University Press.