Query Evaluation in Election Databases

Benny Kimelfeld
Technion
Haifa, Israel
bennyk@cs.technion.ac.il

Phokion G. Kolaitis
UC Santa Cruz
and IBM Research-Almaden
California, USA
kolaitis@soe.ucsc.edu

Muhammad Tibi
Technion
Haifa, Israel
m7mdtb@cs.technion.ac.il

ABSTRACT

Election databases are the main elements of a recently introduced framework that aims to create bridges between the computational social choice and the data management communities. An election database consists of incomplete information about the preferences of voters, in the form of partial orders, alongside with standard database relations that provide contextual information. Earlier work in computational social choice focused on the computation of possible winners and necessary winners that are determined by the available incomplete information and the voting rule at hand. The presence of the relational context, however, permits the formulation of sophisticated queries about voting rules, candidates, potential winners, issues, and positions on issues. Such queries can be given possible answer semantics and necessary answer semantics on an election database, where the former means that the query is true on some completion of the given partial orders and the latter means that the query is true on every such completion.

We carry out a systematic investigation of query evaluation on election databases by analyzing how the interaction between the partial preferences, the voting rules and the relational context impacts on the complexity of query evaluation. To this effect, we focus on positional scoring rules and unions of conjunctive queries. We establish a number of results that delineate the complexity of the possible answers and of the necessary answers for different positional scoring rules and for various classes of unions of conjunctive queries. Furthermore, we show that query evaluation is fixed-parameter tractable, where the parameter is the number of candidates in the election.

CCS CONCEPTS

• **Theory of computation → Incomplete, inconsistent, and uncertain databases**; **Database query languages (principles)**; • **Applied computing → Voting / election technologies**;

KEYWORDS

Necessary and Possible Answers; Computational Social Choice; Positional Scoring Rules.

1 INTRODUCTION

During the past two decades, computational social choice has emerged as an interdisciplinary area between social choice theory, economics, mathematics, logic, and computer science. Social choice theory studies how votes or, more broadly, preferences of individual members of a society can be aggregated in such a way that the society arrives at a collective decision. Social choice theory has a long history that spans several centuries, from the analysis of voting manipulation by Pliny the Younger in Ancient Rome to the study of particular voting rules by Jean-Charles de Borda and Marquis de Condorcet in the 18th Century (now known as the Borda rule and the Condorcet method, respectively), to, more recently, the ground-breaking work on dictatorial aggregation by Kenneth Arrow in the 1950s. (See [8] for a brief history of social choice theory.)

Computational social choice infuses an algorithmic perspective into social choice theory. In particular, the computational aspects of preference aggregation in an election have been a focal point of research in this area. It is often the case, however, that preferences are only partial, since, for example, a voter may be undecided between two candidates, or our knowledge of the voter's preference is incomplete. For this reason, Konczak and Lang [21] introduced the notions of *necessary winners* and *possible winners* as those candidates who win in *every* completion, and *at least one* completion, respectively, of the given partial preferences. This work eventually led to a classification of the computational complexity of the necessary and possible winners for a variety of voting rules [3, 4, 27].

Arguably, the *positional scoring rules* form the most extensively studied class of voting rules. Under such a rule, every candidate receives from every voter a score that is determined only by the position of the candidate in the voter's ranking. Hence, a positional scoring rule is defined by a sequence of *scoring vectors* that specify the score for each position (there is a scoring vector of length m, for each $m \geq 1$, where m stands for the number of candidates). A *winner* in an election is a candidate who achieves the highest total score from the voters. Thus, in general, there may be several winners in an election, in which case they are also referred to as *co-winners*. It is conceivable, however, that only one candidate achieves the highest total score from the voters, in which case that candidate is a *unique* winner. The *plurality* rule and the *veto* rule are two well known examples of positional scoring rules. The scoring vectors of the plurality rule are of the form $(1, 0, \dots, 0)$, while the scoring vectors of the veto rule are of the form $(1, 1, \dots, 1, 0)$. Thus, under the plurality rule, the winners are the candidates who are at top of the ranking of as many voters as possible, while, under the veto rule, the winners are the candidates who are the bottom of the ranking of as few voters as possible.

We focus on positional scoring rules whose scoring vectors are computable in polynomial time in the number of candidates. Under such rules, the necessary winners can be computed in polynomial time [21, 27]. The possible answers can be computed in polynomial time under the plurality and veto rules, but their computation is NP-complete for any other *pure* rule, as established in a sequence of studies [3, 4, 21, 27]. A positional scoring rule is *pure* if, for all $m > 1$, the scoring vector for m candidates is obtained from the scoring vector for $m - 1$ candidates by inserting a new score into the vector.

Elections or polls do not take place in a vacuum; instead, they take place in a context in which an abundance of additional information may be available, including information about the candidates (gender, age, education, wealth), information about issues and positions of candidates on issues, and information about campaign contributions (donors, amounts, recipients). Thus, one may be interested in formulating and answering queries that take into account both the given partial preferences and the contextual information available.

The preceding considerations motivated Kimelfeld, Kolaitis, and Stoyanovich [20] to introduce a new framework that aims to create, for the first time, bridges between the computational social choice and the database management communities.

The main conceptual contribution of [20] is the development of rigorous semantics of queries that involve both partial preferences and contextual information about candidates, issues, positions, and so on. To this effect, the notions of *necessary answers* and *possible answers* to queries were introduced as an extension of the notions of necessary winners and possible winners. To appreciate the difference between, say, necessary winners and necessary answers, consider the Boolean query q that asks whether or not there is a winner who is Republican. Given a set **P** of partial orders representing the (partial) preferences of voters, it is conceivable that no single candidate is a winner in every completion **T** of **P**, which implies that the set of necessary winners is the empty set. Yet, it is also conceivable that in every completion **T** of **P** one of the winners is a Republican, hence the necessary answers of the query q is "yes." In particular, this example shows that the necessary answers to queries cannot always be obtained from the necessary winners.

As regards technical contributions, a study of the necessary answers of conjunctive queries was initiated in [20] and some preliminary complexity results were obtained. In particular, it was shown that there are natural conjunctive queries involving winners and database relations such that computing the necessary answers of these queries under the plurality rule is coNP-complete. This contrasts sharply with earlier results in computational social choice to the effect that, as previously mentioned, there is a polynomial-time algorithm for computing the necessary winners under every positional scoring rule, including the plurality rule [21, 27].

Summary of results. We carry out a systematic investigation of the algorithmic aspects of query evaluation on *election* databases, that is, databases that consist of standard database relations and a set of partial orders representing the preferences of voters. We analyze how the interaction between the partial preferences, the voting rules and the relational context impacts on the computational complexity of query evaluation. We establish a number of results that delineate the complexity of the possible answers and of the necessary answers for various classes of unions of conjunctive queries and under different positional scoring rules.

The scope of our investigation is broader than that in [20] along several different dimensions. First, while only the necessary answers to queries were investigated in [20], here we investigate both the necessary answers and the possible answers to queries. Second, most of our results cover arbitrary pure positional scoring rules, thus they go well beyond the plurality rule and a few other voting rules studied in [20]. Third, we consider queries that involve not only winners, but also unique winners, while only queries involving winners (but not unique winners) were considered in [20].

To give a taste of our findings, we state here a few of the results obtained in this paper. For the possible answers, we show that if q is an arbitrary union of Boolean conjunctive queries that may involve winners and unique winners, then there are polynomial-time algorithms for computing the possible answers to q under the plurality rule or under the veto rule. This gives a fairly complete picture for the class of pure positional scoring rules, because the aforementioned NP-hardness results for the possible winners [3, 4, 27] imply the NP-hardness of the possible answers of unions of conjunctive queries under pure positional scoring rules other than plurality and veto.

As for the necessary answers, we consider the Boolean conjunctive query $q^k_{\text{wr}}()$ for every $k \geq 1$, where

$$q^k_{\text{wr}}() \;:-\; \textsc{Winner}(x_1), \ldots, \textsc{Winner}(x_k), R(x_1, \ldots, x_k)\,.$$

We show that there are polynomial-time algorithms for computing the necessary answers of q^1_{wr} under the plurality rule or the veto rule, but computing the necessary answers of q^1_{wr} is coNP-complete under every other pure positional scoring rule. We further show how the tractable cases generalize from q^1_{wr} to a wide class of unions of conjunctive queries where, in each disjunct, the Winner atoms are disconnected. In contrast, we prove that for every $k > 1$, computing the necessary answers of q^k_{wr} is coNP-complete under every pure positional scoring rule (including plurality and veto). For $k > 2$, this hardness applies to *every* positional scoring rule, pure or not.

Finally, we examine query evaluation using the lens of parameterized complexity and we show that evaluation of unions of conjunctive queries is fixed-parameter tractable, where the parameter is the number of candidates in the election.

Organization. The rest of the paper is structured as follows. In Section 2, we give the basic definition and terminology, and in particular, describe briefly the framework of election databases in [20]. In Sections 4 and 5, we study the possibility and necessity problems, respectively, for queries that involve winners but not *unique* winners. In Section 6, we extend our study to queries that involve unique winners. In Section 7, we study the parameterized complexity of the possibility and necessity problems, and then conclude in Section 8.

2 PRELIMINARIES

This section contains the definitions of the main concepts and background material.

2.1 Databases and Queries

A (relational) *schema* $\mathcal{S}$ is a collection of *relation symbols* with each relation symbol R in $\mathcal{S}$ having an associated arity that we denote by $ar(R)$. We assume a countably infinite set of *constants* that are used as database values. A *relation* is a set of tuples of constants, each having the same arity (length) that we denote by $ar(T)$. A *database* D (over the schema $\mathcal{S}$) is a collection of relations such that for each relation symbol R, the database D contains a relation R^D with $ar(R) = ar(R^D)$. The active domain of D, denoted by $adom(D)$, is the set of all constants occurring in relations in D.

A *query* is a function that maps databases to relations. Formally, a query q of arity $ar(q)$ is a function that maps every database over $\mathcal{S}$ to a finite relation $q(D)$ of arity $ar(q)$ on the active domain $adom(D)$. We say that each tuple in $q(D)$ is an *answer* to q on D. If the arity of q is zero, then we say that q is a *Boolean* query; in this case, $D \models q$ denotes that $q(D)$ consists of the empty tuple (), while $D \not\models q$ denotes that $q(D)$ is empty.

A *conjunctive query* (CQ) q over the schema $\mathcal{S}$ is a query definable by a first-order formula of the form

$$\exists \mathbf{y}_1 \cdots \exists \mathbf{y}_m \theta(\mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_m),$$

where θ is a conjunction of atomic formulas with variables among those in $\mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_m$. In the sequel, conjunctive queries will be written as logic rules, i.e., as expressions of the form

$$q(\mathbf{x}) :\!- R_1(\mathbf{t}_1), \ldots, R_n(\mathbf{t}_m)$$

where each R_i is a relation symbol of $\mathcal{S}$, each $\mathbf{t}_i$ is a tuple of variables and constants with the same arity as R_i, and $\mathbf{x}$ is a tuple of k variables from $\mathbf{t}_1, \ldots, \mathbf{t}_m$. We call $q(\mathbf{x})$ the *head* of q, and $R_1(\mathbf{t}_1), \ldots, R_n(\mathbf{t}_m)$ the *body* of q; each $R_i(\mathbf{t}_i)$ is an *atom* of q. The variables occurring in the body but not in the head are existentially quantified. The answers to q on a database D are the projections to $\mathbf{x}$ of all homomorphisms from q to D.

A *union of conjunctive queries* (UCQ) is an expression q of the form $q_1 \cup \cdots \cup q_\ell$, where each q_i is a conjunctive query and all q_is have the same arity (i.e., the arity $ar(q)$ of q). If D is a database, then $q(D) = q_1(D) \cup \cdots \cup q_\ell(D)$.

2.2 Voting Profiles and Voting Rules

We now recall the basic terminology of voting theory in computational social choice. For additional background material, we refer the reader to the handbook of computational social choice [7] and, in particular, to the chapter on incomplete information in voting [6]. Let $C = \{c_1, \ldots, c_m\}$ be a set of *candidates*, and let $V = \{v_1, \ldots, v_n\}$ be a set of voters. A *complete voting profile* is a tuple $\mathbf{T} = (T_1, \ldots, T_n)$, where each T_i is a total order over C, representing the ranking (preference) of voter v_j of the candidates in C. A *voting rule r* is a function that maps every complete profile $\mathbf{T}$ into a set $\mathbf{W}(r, \mathbf{T}) \subseteq C$ of *winners*. We say that a candidate c is a *winner* if $c \in \mathbf{W}(r, \mathbf{T})$; we also say that c is a *unique winner* if $\mathbf{W}(r, \mathbf{T}) = \{c\}$. We write $\mathbf{U}(r, \mathbf{T})$ to denote the set of unique winners; note that $\mathbf{U}(r, \mathbf{T})$ is either a singleton or the empty set. On occasion, we will use the term "co-winner" to emphasize that we refer to a winner who is not necessarily unique.

In this paper, we focus on the class of *positional scoring rules*, which is arguably the most extensively studied class of voting rules in computational social choice.

$\mathbf{T}^1$				
T_1	D	T	H	B
T_2	H	T	B	D
T_3	D	H	B	T
T_4	D	B	H	T
T_5	B	H	T	D
$r(m,\cdot)$	1	1	0	0

Winner
Hillary
Donald

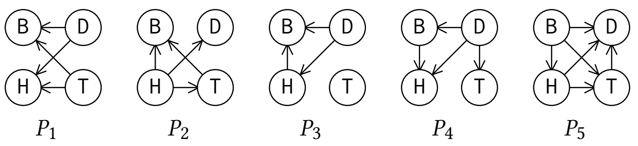
UWinner

$\mathbf{T}^2$				
T_1	D	T	B	H
T_2	H	D	T	B
T_3	D	H	T	B
T_4	D	B	T	H
T_5	B	H	T	D
$r(m,\cdot)$	1	1	0	0

Winner
Donald

UWinner
Donald

(a) Complete voting profiles $\mathbf{T}^1 = (T^1_1, \dots, T^1_5)$, $\mathbf{T}^2 = (T^2_1, \dots, T^2_5)$ over the candidates Bernie, Donald, Hillary and Ted, together with corresponding sets of winners and unique winners. Candidates are denoted by the first letter of their name (e.g., D for Donald); the order is from left to right, the leftmost being the most preferred.



(b) A partial voting profile $\mathbf{P} = (P_1, \dots, P_5)$; an edge $c \to d$ denotes that c is preferred to d.

Cand		
name	party	birth
Bernie	D	1941
Donald	D	1947
Hillary	R	1946
Ted	R	1970

Donate		
donor	type	cand
Soros	indv	Hillary
Trump	indv	Donald
UAPE	PAC	Hillary
UAPE	PAC	Ted
Wilks	indv	Ted

Position		
cand	issue	pos
Hillary	TPP	yes
Donald	TPP	no
Bernie	TPP	no
Ted	TPP	no
Hillary	PPACA	yes
Donald	PPACA	no
Bernie	PPACA	yes
Ted	PPACA	no

(c) A database D over a schema $\mathcal{S}$

Figure 1: Examples of profiles, a partial profile, and a database. Viewed bottom up, this an example of an election database $(D, \mathbf{P})$ and two expansions $D \sqcup \mathbf{T}^1$ and $D \sqcup \mathbf{T}^2$.

A *positional scoring rule* r is a function that maps every number m to a *scoring vector* $(r(m,1), \dots, r(m,m))$ of natural numbers, called the *score values*, such that $r(m,1) \geq r(m,2) \dots \geq r(m,m)$. Here, m is the number of candidates and $r(m,j)$ is the score that a candidate is awarded whenever she is at the jth position of a voter. We denote by $r(\cdot,m)$ the vector $(r(1,m), \dots, r(m,m))$. Suppose that $\mathbf{T} = (T_1, \dots, T_n)$ is a complete voting profile. The score $s(T_i, c)$ of a candidate c on T_i is the value $r(m,j)$ where j is the position of candidate c in T_i. When the positional scoring rule r is applied to $\mathbf{T} = (T_1, \dots, T_n)$, it assigns to each candidate c the sum $\sum_{i=1}^{n} s(T_i, c)$ as the *score* of c. A candidate is a *winner* if her score is greater or equal to the score of all candidates. Consequently, a candidate is a *unique winner* if her score is strictly greater than that of all other candidates. We now give several examples of well known positional scoring rules.

- The *plurality* rule $(1, 0, \dots, 0)$, where the winners are the candidates that voters most frequently rank first.
- The k-*approval* rule $(1, \dots, 1, 0, \dots, 0)$ that starts with k ones and then 0's, where the winners are the candidates that voters most frequently rank among the top k.
- The *veto* rule $(1, \dots, 1, 0)$, where the winners are the candidates that voters least frequently rank last.
- The k-*veto* rule that starts with 1's and ends with k zeros, where the winners are the candidates that voters least frequently rank among the bottom k.
- The *Borda* rule $(m-1, m-2, \dots, 0)$, where the score of a candidate is the position itself minus 1 in reverse order.

We assume that the scoring rule r is such that the score values $r(m,i)$ are computable in polynomial time in m. Hence, $\mathbf{W}(r, \mathbf{T})$ and $\mathbf{U}(r, \mathbf{T})$ are also computable in polynomial time in the size of T. To avoid trivialities, we assume that every $r(\cdot,m)$ contains at least two different score values, that is, $r(m,1) > r(m,m)$ holds for all $m > 1$. It follows that $\mathbf{W}(r, \mathbf{T})$ is always nonempty, while $\mathbf{U}(r, \mathbf{T})$ is either a singleton or the empty set (the latter is the case when $|\mathbf{W}(r, \mathbf{T})| > 1$). We also assume that for all $m > 1$, the score values in $r(\cdot,m)$ are co-prime numbers (i.e., their biggest common divisor is 1), since multiplying all score values by the same number has no impact on the outcome of the rule.

The rule r is *pure* if for every $m \geq 2$, the scoring vector $r(\cdot,m)$ is obtained from $r(\cdot,m-1)$ by inserting a score value at some position. All aforementioned positional scoring rules (plurality, k-approval, veto, k-veto, and Borda) are pure.

Example 2.1. Our running example is taken from the 2016 US presidential elections. There are four candidates, *Bernie*, *Donald*, *Hillary*, and *Ted*, and five voters. Figure 1a shows two voting profiles $\mathbf{T}^1$ and $\mathbf{T}^2$, each consisting of five linear orders (presented as sequences) over the candidates. (The other parts of Figure 1 will be discussed later on.) The positional scoring rule is 2-approval (which, in this case, is also 2-veto), and it is shown below each profile. The scores of the candidates *Bernie*, *Donald*, *Hillary*, and *Ted* in $\mathbf{T}^1$ are, respectively, 2, 3, 3 and 2. Hence, $\mathbf{W}(r, \mathbf{T}^1)$ consists of Donald and Hillary, while $\mathbf{U}(r, \mathbf{T}^1)$ is empty since there is no unique winner. In $\mathbf{T}^2$, the scores of these candidates are, respectively, 2, 4, 3 and 1; thus, $\mathbf{W}(r, \mathbf{T}^2)$ consists of just Donald. Since Donald is the unique winner, we have that $\mathbf{U}(t, \mathbf{T}^2)$ also consists of Donald. □

2.3 Partial Profiles

Often, our knowledge about the voter preference is only partial. Missing information in preferences is commonly modeled using a *partial order*, that is, a relation $\succeq$ that is reflexive, ($a \succeq a$), transitive, ($a \succeq b$ and $b \succeq c$ imply $a \succeq c$), and antisymmetric, ($a \succeq b$ and $b \succeq a$ imply $a = b$), but not necessarily total (it may be the case that neither $a \succeq b$ nor $b \succeq a$ holds). A *completion* of a partial order is a total order that extends that partial order. A partial order may have exponentially many completions.

A *partial voting profile* is a tuple $\mathbf{P} = (P_1, \dots, P_n)$, where each P_i is a partial order of the set C of candidates, representing the partial ranking (partial preference) of voter v_j on the candidates. A *completion* of a partial voting profile $\mathbf{P} = (P_1, \dots, P_n)$ is a complete voting profile $\mathbf{T} = (T_1, \dots, T_n)$ such that each T_i is a completion of the partial order P_i. The notions of *possible* and *necessary* winners were introduced by Konczak and Lang [21].

Let r be a voting rule and $\mathbf{P}$ a partial voting profile.

- The set $\mathrm{PW}(r, \mathbf{P})$ of the *possible winners* with respect to (w.r.t.) r and $\mathbf{P}$ is the union of the sets $\mathbf{W}(r, \mathbf{T})$, where $\mathbf{T}$ varies over all completions of $\mathbf{P}$.
- The set $\mathrm{NW}(r, \mathbf{P})$ of the *necessary winners* w.r.t. r and $\mathbf{P}$ is the intersection of the sets $\mathbf{W}(r, \mathbf{T})$, where $\mathbf{T}$ varies over all completions of $\mathbf{P}$.
- The set $\mathrm{PU}(r, \mathbf{P})$ of the *possible unique winners* w.r.t. r and $\mathbf{P}$ is the union of the sets $\mathbf{U}(r, \mathbf{T})$, where $\mathbf{T}$ varies over all completions of $\mathbf{P}$.
- The set $\mathrm{NU}(r, \mathbf{P})$ of the *necessary unique winners* w.r.t. r and $\mathbf{P}$ is the intersection of the sets $\mathbf{U}(r, \mathbf{T})$, where $\mathbf{T}$ varies over all completions of $\mathbf{P}$.

In other words, a candidate c is a possible winner (respectively, a possible unique winner) with respect to r and $\mathbf{P}$ if c is a winner (respectively, a unique winner) in at least one completion $\mathbf{T}$ of $\mathbf{P}$. Also, c is a necessary winner (respectively, a necessary unique winner) with respect to r and $\mathbf{P}$ if c is a winner (respectively, a unique winner) in every completion $\mathbf{T}$ of $\mathbf{P}$. Observe that $\mathrm{NW}(r, \mathbf{P})$, $\mathrm{PU}(r, \mathbf{P})$ and $\mathrm{NU}(r, \mathbf{P})$ can be empty, but $\mathrm{PW}(r, \mathbf{P})$ is never empty (since every completion has at least one winner). Moreover, there can be any number of *possible* unique winners, but there can be at most one *necessary* unique winner.

Example 2.2. Continuing our running example, Figure 1b depicts a partial voting profile $\mathbf{P}$ that consists of five partial orders of the five voters (represented as directed acyclic graphs) over the four candidates. It is easy to verify that both $\mathbf{T}^1$ and $\mathbf{T}^2$ in Figure 1a are completions of $\mathbf{P}$.

From the discussion in Example 2.1, we conclude that Donald and Hillary are both possible winners. Note that there is a completion in which Ted is a winner (e.g., Ted gets a unit score value from the first four voters), and, in fact, the unique winner. Bernie, however, cannot win in any completion. To see this, observe that he must be ranked no higher than third by the first three voters (so, gets a zero score from these voters), thus the maximum score he can get is 2. Among the ten units granted by the five voters, at least one candidate must get a score of 3, and therefore, will surpass Bernie. Via a similar reasoning, we conclude that the maximum score that Hillary can get is 3, and hence, it is impossible for her to be the single winner (since there is always another candidate with 3 or more units). Thus, Hillary is a possible winner, but not a possible *unique* winner.

Finally, we observe that for every one of the four candidates, there is a completion where the candidate does not win. (In particular, $\mathbf{T}^2$ provides such an example for every candidate other than Donald, so it remains to find one just for Donald.) Therefore, no candidate is a necessary winner.

In conclusion, it holds that

- $\mathrm{NW}(r, \mathbf{P}) = \mathrm{NU}(r, \mathbf{P}) = \emptyset$;
- $\mathrm{PW}(r, \mathbf{P}) = \{\text{Donald}, \text{Hillary}, \text{Ted}\}$;
- $\mathrm{PU}(r, \mathbf{P}) = \{\text{Donald}, \text{Ted}\}$. □

On the face of the definitions, computing possible and necessary (unique or not) winners requires exponential time since a partial order may have exponentially many completions. There is a substantial body of research on the computational complexity of the necessary and the possible winners for a variety of voting rules. The following *complete* classification of the complexity for *all* pure positional scoring rules was obtained through the work of Konczak and Lang [21], Xia and Conitzer [27], Betzler and Dorn [4], and Baumeister and Rothe [3].

Theorem 2.3. [Classification Theorem] *Let r be a pure positional scoring rule.*

(1) *If r is the plurality rule or the veto rule, then there is a polynomial-time algorithm for computing* $\mathrm{PW}(r, \mathbf{P})$ *and* $\mathrm{PU}(r, \mathbf{P})$*, given a partial voting profile* $\mathbf{P}$*. Otherwise, the following two problems are NP-complete: given a partial voting profile* $\mathbf{P}$ *and a candidate c, determine whether $c \in \mathrm{PW}(r, \mathbf{P})$, and determine whether $c \in \mathrm{PU}(r, \mathbf{P})$.*
(2) *There is a polynomial-time algorithm for computing both* $\mathrm{NW}(r, \mathbf{P})$ *and* $\mathrm{NU}(r, \mathbf{P})$*, given a partial voting profile* $\mathbf{P}$.

Put Theorem 2.3 differently, it is NP-hard to compute the possible winners for all pure positional scoring rules, with the exception of plurality and veto where the possible winners can be found in polynomial time. In contrast, the necessary winners can be computed in polynomial time for every pure positional scoring rule; in fact, this tractability result holds for *every* positional scoring rule (pure or not) [27].

3 ELECTION DATABASES

Here, we review the framework of Kimelfeld, Kolaitis, and Stoyanovich [20] that brings together computational social choice and relational databases. The aim is to have a unifying setting for analyzing partial voting profiles in the context of a database that holds information about candidates, voters, and issues.

Informally, an *election database* consists of a partial profile $\mathbf{P}$ and a database D that provides contextual information about the candidates (and beyond), as illustrated in Figure 1.[1] If r is a voting rule, then each completion $\mathbf{T}$ of $\mathbf{P}$ gives rise to an *expansion* of D obtained by augmenting D with the unary relations $\mathbf{W}(r, \mathbf{T})$ and $\mathbf{U}(r, \mathbf{T})$ of the winners and unique winners.

We now give the precise definition of an election database.

Let $\mathcal{S}$ be a schema and let Winner and UWinner be two new unary relation symbols that will be interpreted by the set of the winners and the set of unique winners, respectively. Let $\mathcal{S}^e$ be the schema obtained by augmenting $\mathcal{S}$ with these two unary relation symbols, that is, $\mathcal{S}^e = \mathcal{S} \cup \{\textsc{Winner}, \textsc{UWinner}\}$.

Definition 3.1 (Election Database). An *election database* over a schema $\mathcal{S}$ is a pair $(D, \mathbf{P})$, where D is a database over $\mathcal{S}$ and $\mathbf{P}$ is a partial voting profile. Let r be a voting rule. Each completion $\mathbf{T}$ of $\mathbf{P}$ gives rise to an *expansion* $D \sqcup \mathbf{T}$ of $(D, \mathbf{P})$ under r such that:

- $D \sqcup \mathbf{T}$ is a database over the relational schema $\mathcal{S}^e = \mathcal{S} \cup \{\textsc{Winner}, \textsc{UWinner}\}$;
- $R^{D \sqcup \mathbf{T}} = R^D$, for every relation symbol R of $\mathcal{S}$;
- $\textsc{Winner}^{D \sqcup \mathbf{T}} = \mathbf{W}(r, \mathbf{T})$;
- $\textsc{UWinner}^{D \sqcup \mathbf{T}} = \mathbf{U}(r, \mathbf{T})$.

Example 3.2. The schema $\mathcal{S}$ of our running example consists of the relation symbols *Cand*, *Position* and *Donate* that are instantiated in the database D of Figure 1c. The *Cand* relation includes information about the candidates (e.g., Bernie was born in 1941 and is currently with the Democratic Party), the *Position* relation contains the positions of the candidates on issues (e.g., Hillary supports the *Trans-Pacific Partnership* (TPP), and Donald opposes the *Patient Protection and Affordable Care Act* (PPACA), known as *Obamacare*), and the *Donate* relation contains information about donations to candidate (e.g., the individual Soros donated to Hillary, while the political action committee (PAC) UAPE donated to Ted). Combined with the partial profile $\mathbf{P}$ described in the figure (and discussed in Example 2.2), we get the election database $(D, \mathbf{P})$.

Recall the completions $\mathbf{T}^1$ and $\mathbf{T}^2$ of Figure 1a. Shown to the right of each completion are the Winner and UWinner relations of the corresponding expansion. In $\mathbf{T}^1$, the UWinner relation is empty, since the winners are Donald and Hillary. Yet, in $\mathbf{T}^2$, the Winner and UWinner relations are the same, since there is a unique winner (namely, Donald). □

[1]The information in the figure is taken from Diffen (https://www.diffen.com) and The New York Times (https://www.nytimes.com/interactive/2016/us/elections/top-presidential-donors-campaign-money.html).

3.1 Necessary and Possible Answers

Next, we recall the definitions of the necessary and possible query answers from [20]. Let $\mathcal{S}$ be an schema, $(D, \mathbf{P})$ an election database, q a query over $\mathcal{S}^e$, and r a voting rule. Let $\mathbf{a}$ be a tuple of constants of arity $ar(q)$.

- We say that a tuple $\mathbf{a}$ is a *possible answer* if $\mathbf{a} \in q(D \sqcup \mathbf{T})$ for some expansion $D \sqcup \mathbf{T}$.
- We say that a tuple $\mathbf{a}$ is a *necessary answer* if $\mathbf{a} \in q(D \sqcup \mathbf{T})$ for every expansion $D \sqcup \mathbf{T}$.

In terms of data complexity, the problem of computing the possible and necessary answers of CQs and UCQs amounts to the evaluation of Boolean queries.

Definition 3.3 (Possibility/Necessity Problem). Let $\mathcal{S}$ be a schema, q a Boolean query over $\mathcal{S}^e$ and r a voting rule.

- $\mathsf{Possibility}(q, r)$ is the following decision problem: given $(D, \mathbf{P})$, is q possibly true (i.e., does $D \sqcup \mathbf{T} \models q$ for some expansion $D \sqcup \mathbf{T}$ of $(D, \mathbf{P})$)?
- $\mathsf{Necessity}(q, r)$ is the following decision problem: given $(D, \mathbf{P})$, is q necessarily true (i.e., does $D \sqcup \mathbf{T} \models q$ for every expansion $D \sqcup \mathbf{T}$ of $(D, \mathbf{P})$)?

Example 3.4. We now illustrate queries. For a candidate c, the following query asks whether c is a winner.

$$q_c() \ {:-}\ \textsc{Winner}(c) \tag{1}$$

In particular, q_c is possible if and only if c is a possible winner, and q_c is necessary if and only if c is a necessary winner.

The following queries apply to our running example (Figure 1). The next one asks whether there is a Republican winner. (We use the convention of an underscore instead of a variable with a single occurrence.)

$$q_1() \ {:-}\ \textsc{Winner}(x)\,,\ \mathit{Cand}(x, \mathsf{R}, \_)$$

Hence, $\mathsf{Possibility}(q_1, r)$ asks whether it is possible to have a Republican winner, and $\mathsf{Necessity}(q_1, r)$ asks whether we necessarily have a Republican winner. As explained in Example 2.2, in every completion it is the case that either

Donald or Ted are winners. Hence, in the running example, both $\mathsf{Possibility}(q_1, r)$ and $\mathsf{Necessity}(q_1, r)$ are true. The next query asks whether there is a unique winner who is also a Republican.

$$q_1'() \text{ :- } \textsc{UWinner}(x)\,,\ \mathit{Cand}(x, \mathsf{R}, \_)$$

As shown in Figure 1a, it Donald is a possible unique winner, and hence, $\mathsf{Possibility}(q, r)$ is true. However, $\mathsf{Necessity}(q, r)$ is false, simply because there is an expansion in which the UWinner relation is empty (i.e., there are no unique winners).

The following query asks whether there are two winners who disagree on Obamacare.

$$\begin{aligned} q_2() \text{ :- } & \mathit{Position}(x, \mathsf{PPACA}, \mathsf{yes})\,,\ \mathit{Position}(y, \mathsf{PPACA}, \mathsf{no})\,,\\ & \textsc{Winner}(x)\,,\ \textsc{Winner}(y) \end{aligned}$$

$\mathsf{Possibility}(q_2, r)$ is true as witnessed by the completion $\mathbf{T}^1$ in Figure 1a, while $\mathsf{Necessity}(q_2, r)$ is false due to $\mathbf{T}^2$.

Finally, the following query asks whether there are two winners, one Democratic and one Republican, who get donations from the same source.

$$\begin{aligned} q_3() \text{ :- } & \mathit{Cand}(x, \mathsf{R}, \_)\,,\ \mathit{Cand}(y, \mathsf{D}, \_),\ \textsc{Winner}(x)\,,\\ & \textsc{Winner}(y)\,,\ \mathit{Donate}(z, \_, x)\,,\ \mathit{Donate}(z, \_, y) \end{aligned}$$

It is possible to build a completion where both Hillary and Ted (who both receive donations from UAPE) are winners, hence, $\mathsf{Possibility}(q_3, r)$ is true; moreover, both completions of Figure 1a witness that $\mathsf{Necessity}(q_3, r)$ is false. □

Example 3.5. For every positive integer k, let q^k_{wr} be the following Boolean conjunctive query:

$$q^k_{\mathsf{wr}}() \text{ :- } \textsc{Winner}(x_1), \ldots, \textsc{Winner}(x_k), R(x_1, \ldots, x_k) \quad (2)$$

This query asks whether there is a sequence of k winners that constitutes a tuple of R.

In the remainder of the paper, we will make repeated use of the family of the queries q^k_{wr}, $k \geq 1$. □

Note that a query q over $\mathcal{S}^{\mathrm{e}}$ may involve both the Winner and UWinner relation symbols. When q is a UCQ, we use the terms Winner *atoms* and UWinner *atoms* to refer to atoms that involve, respectively, the Winner and the UWinner relation symbol. In the sections that follow, we investigate the complexity of the decision problems $\mathsf{Possibility}(q, r)$ and $\mathsf{Necessity}(q, r)$, where q is a Boolean UCQ.

We note that other notions of databases and queries over partial orders have been studied [1, 14, 18], but none of them involves elections or social choice.

4 QUERIES WITH WINNER ATOMS: POSSIBLE ANSWERS

Next, we investigate the complexity of computing the possible answers for UCQs that involve Winner atoms only (but no UWinner atoms). Thus, in what follows in this section, we make the blanket assumption that the term UCQ refers to a union of conjunctive queries whose atoms are Winner atoms or atoms with relation symbols from the schema $\mathcal{S}$.

4.1 Tractability

According to Theorem 2.3, the possible winners can be computed in polynomial time when the positional scoring rule is plurality or veto. In the remainder of this section, we show that this tractability result extends to the possible answers of every UCQ (and beyond).

To establish tractability for plurality and veto, we use the tractability of a polygamous version of the perfect-matching problem. This problem is a special case of generalizations of perfect matching that are known to be in polynomial time [12, 25]. This special case has a simple proof, which we now give to make the subsequent results as self-contained as possible.

Definition 4.1 (Polygamous Matching). Polygamous Matching is the following decision problem: Given a bipartite graph $G = (V \uplus U, E)$ and natural numbers $\alpha_u \leq \beta_u$ for all $u \in U$, is there a subset of E where each $v \in V$ is incident to exactly one edge and every $u \in U$ is incident to at least α_u edges and at most β_u edges?

Proposition 4.2. *Polygamous Matching is solvable in polynomial time.*

Proof. We reduce Polygamous Matching to the problem of determining whether a *perfect* matching exists (which is solvable in polynomial time, e.g., via the Hungarian algorithm or maximum network flows). Given an input G, α, β to Polygamous Matching, we construct a bipartite graph $G' = (V' \uplus U', E')$ as follows. We obtain U' from U by replacing every node u with β_u distinct *copies* of u. The set V' is the union of V and a set of $|U'| - |V|$ of new *dummy* nodes. (If $|U'| < |V|$, then no polygamous matching exists.) Next, we construct E' by considering every node $u \in U$ and connecting all its copies in U' to the all the neighbors $v \in V$ that u has in g; in addition, we connect $\beta_u - \alpha_u$ of the copies of u to all the dummy nodes in V'. The reader can verify that G is a "yes" instance of Polygamous Matching if and only if G' has a perfect matching. In particular, to obtain a polygamous matching from a perfect matching M of G', we take all edges (v, u) such that $v \in V$ and M connects v to a copy of u. □

We will also use the following lemma, which we prove using Proposition 4.2. Hereafter, we will denote by $\min(v)$ the set of candidates c that the voter v does not prefer to any other candidate (hence, such a c can be last in a completion), and by $\max(v)$ the set of candidates c that are not preceded by any other voter (hence, such a c can be first in a completion).

Lemma 4.3. *Let r be the plurality rule or the veto rule. Then the following problem is solvable in polynomial time: Given a set C of candidates, a partial profile $\mathbf{P}$ over C, and a subset $S \subseteq C$, is there a completion $\mathbf{T}$ such that $S \subseteq \mathbf{W}(r, \mathbf{T})$?*

Proof. We begin with the plurality rule. We solve the problem by considering every possible score $s = 1, \dots, n$ (where n is the number of voters), and test whether there is a completion $\mathbf{T}$ such that every candidate in S gains *precisely* s votes, and every candidate outside of S gains *at most* s votes. If the answer is true for at least one s, we return true; otherwise, we return false.

So, the question at hand is whether we can assign to each voter one of her top candidates so that every candidate in S is assigned to precisely s voters, and every candidate outside of S is assigned to at most s voters. This problem is a special case of Polygamous Matching (Definition 4.1), where

- V is the set of voters,
- U is the set of candidates,
- E connects c and v, whenever $c \in \max(v)$,
- $\alpha_u = \beta_u = s$, for every $u \in S$,
- $\alpha_u = 0$ and $\beta_u = s$, for every $u \in C \setminus S$.

We then conclude the proof via Proposition 4.2.

For the veto rule, the proof is analogous. Here, the question is whether we can assign to each voter one of her *bottom* candidates so that every candidate in S is assigned to precisely $n - s$ voters, and every candidate outside of S is assigned to *at least* $n - s$ voters. In particular, V and U are the same as in the case of plurality; furthermore,

- E connects c and v whenever $c \in \min(v)$,
- $\alpha_u = \beta_u = n - s$, for every $u \in S$,
- $\alpha_u = n - s$ and $\beta_u = n$, for every $u \in C \setminus S$.

This concludes the proof. □

We can now prove the main result for this section.

Theorem 4.4. *Let r be the plurality rule or the veto rule. If q is a Boolean UCQ, then* Possibility(q, r) *is solvable in polynomial time.*

Proof. Consider an input $(D, \mathbf{P})$ of Possibility(q, r). Then q is possible for $(D, \mathbf{P})$ if and only if one of its CQs is possible for $(D, \mathbf{P})$; thus, we can solve the problem separately for each CQ of q. So, we will assume that q is a CQ (i.e., it has one disjunct) to begin with.

Let $adom(D)$ be the active domain of D (i.e., the set of constants that occur in relations in D, let C be the set of candidates, and let $\tau(q)$ be the set of variables and constants that occur in q. By a *potential homomorphism*, we refer to a mapping $\mu : \tau(q) \to adom(D) \cup C$ that satisfies the following conditions:

(1) $\mu(t) = t$, whenever t is a constant;
(2) μ maps every non-Winner fact to a fact of D;
(3) $\mu(t) \in C$, whenever t occurs in a Winner atom (i.e., the atom Winner(t)) of q.

Then q is possible for $(D, \mathbf{P})$ if and only if there is a potential homomorphism μ and a completion $\mathbf{T}$ of $\mathbf{P}$ such that $\mu(t)$ is a winner for every term t of the third condition. More formally, for a potential homomorphism μ, let us denote by $W(\mu)$ the set of all candidates $\mu(t)$ such that t occurs in a Winner atom. Then, for a completion $\mathbf{T}$ of $\mathbf{P}$, we have that μ is a homomorphism from q to $D \sqcup \mathbf{T}$ (hence, $D \sqcup \mathbf{T} \models q$) if and only if $W(\mu) \subseteq \mathbf{W}(r, \mathbf{T})$.

In view of the above, the algorithm simply constructs all possible μ (in polynomial time) and, for each μ, tests whether there exists a completion $\mathbf{T}$ such that $W(\mu) \subseteq \mathbf{W}(r, \mathbf{T})$. By Lemma 4.3, this test can be carried out in polynomial time. □

We remark that the proof of Theorem 4.4 can be extended to the generalization of UCQs in which negated atoms are allowed. For example, if r is the plurality rule or the veto rule, then Possibility(q, r) is solvable in polynomial time, where q is the query that tests whether there are a winner and a loser both of whom are funded by the same PAC, i.e.,

$$\begin{aligned} q_4() \ \text{:-}\ & Donate(z, \mathsf{PAC}, x)\,,\ Donate(z, \mathsf{PAC}, y) \\ & \textsc{Winner}(x)\,,\ \neg\textsc{Winner}(y) \end{aligned}$$

4.2 Hardness

Let r be a voting rule and let q be a UCQ. To determine whether Possibility(q, r) is true, we need to, at least, be able to determine whether a candidate is a possible winner. Indeed, the possible-winner problem (i.e., determine whether a given candidate c is a possible winner in a given partial profile) is simply Possibility(q_c, r), where q_c is the query defined in (1). Theorem 2.3 implies that Possibility(q_c, r) is NP-hard, for every pure positional scoring rule other than plurality and veto (and regardless of the relational schema $\mathcal{S}$). Clearly, Possibility(q_c, r) reduces to the problem Possibility(q, r) for the query $q()$:- Winner$(x), R(x)$, which we refer to as q^1_{wr} in Equation (2). In fact, it is an easy observation that Possibility(q_c, r) reduces to Possibility(q^k_{wr}, r) for every $k > 0$. Thus, Possibility(q^k_{wr}, r) is NP-hard, for every $k > 0$.

5 QUERIES WITH WINNER ATOMS: NECESSARY ANSWERS

In this section, we analyze the complexity of the necessity problem for CQs and UCQs that involve Winner atoms (but no UWinner atoms). Thus, we continue to make the blanket assumption that the term UCQ refers to a union of conjunctive queries whose atoms are Winner atoms or atoms with relation symbols from the schema $\mathcal{S}$.

Table 1: The complexity of $\mathsf{Necessity}(q^k_{\mathrm{wr}}, r)$**, where** q^k_{wr} **is defined in Equation** (2) **(Example 3.5).**

k	*Complexity*	*Reference*
1	P for veto, plurality; coNP-c. for all other pure pos. rules	Theorem 5.3, Proposition 5.4
>1	coNP-c. for all pure pos. rules	Theorem 5.6
>2	coNP-c. for all pos. rules	Theorem 5.7

Let c be a candidate and let q_c be the CQ

$$q_c() \;:-\; \textsc{Winner}(c).$$

Theorem 2.3 implies that $\mathsf{Necessity}(q_c, r)$ is solvable in polynomial time for every pure positional scoring rule r, Indeed, $\mathsf{Necessity}(q_c, r)$ asks whether c is a necessary winner. In this section, we explore the boundaries of the combinations of r and q that have a tractable necessity testing. For that, we will refer to the CQs q^k_{wr} defined in Equation (2). Our results for these queries are summarized in Table 1.

5.1 Tractability

We begin with tractability results. Towards the main result of this part, we first consider the conjunctive query q^1_{wr} in Equation (2). Kimelfeld, Kolaitis, and Stoyanovich [20] showed that if r is the plurality rule, then $\mathsf{Necessity}(q^1_{\mathrm{wr}}, r)$ is solvable in polynomial time. The next lemma shows that this tractability result also holds true when r is the veto rule.

Lemma 5.1. *If r is the veto rule, then $\mathsf{Necessity}(q^1_{\mathrm{wr}}, r)$ is solvable in polynomial time.*

Proof. Consider an input $(D, \mathbf{P})$ to $\mathsf{Necessity}(q^1_{\mathrm{wr}}, r)$. We solve $\mathsf{Necessity}(q^1_{\mathrm{wr}}, r)$ by searching for a counterexample, that is, a completion where a candidate outside R^D beats every candidate inside R^D. Recall that, in the veto rule, the score that a candidate c gains is $n - k$, where n is the number of voters and k is the number of voters who position c last. So, we consider every $c \in C \setminus R^D$ and $k \in \{0, \dots, n-1\}$, and search for a completion of $\mathbf{P}$ in which c gains a score of at least $n-k$ and each $c' \in R^D$ gains a score of at most $n-k-1$; in other words, at most k voters position c last, and at least $k+1$ voters position c' last, for every $c' \in R^D$.

So, we fix c and k. Recall that, for a voter v, we denote by $\min(v)$ the set of candidates c that v can position last (i.e., v does not prefer c to any other candidate). We reduce the problem to Polygamous Matching (Definition 4.1), which is solvable in polynomial time (Proposition 4.2). In our construction, V is the set of voters, U is the set of candidates, and $E = \{(v, d) \mid d \in min(v)\}$. We set the bounds as follows.

- $\alpha_c = 0$ and $\beta_c = k$;
- $\alpha_{c'} = k + 1$ and $\beta_{c'} = n$, for all $c' \in R^D$;

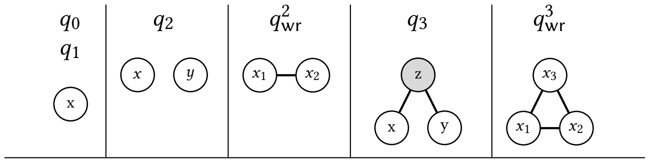


Figure 2: The Gaifman graphs of the queries from Examples 3.4 and 3.5. All the variables are Winner variables, except for the shaded z**.**

- $\alpha_{c''} = 0$ and $\beta_{c''} = n$, for all other candidates c''.

In the construction, selecting an edge (v, d) means that v positions d last. Hence, we get a "yes" instance if and only if there is a completion $\mathbf{T}$ of $\mathbf{P}$ in which the score of c is at least $n - k$ and the score of every $c' \in R^D$ is at most $n - k - 1$. □

The main tractability result of this section is Theorem 5.3 below, which generalizes both Lemma 5.1 and the tractability results in [20] for the plurality rule and for UCQs in which every CQ has *pairwise disconnected* Winner atoms in its *Gaifman graph.* As is well known, the notion of the Gaifman graph plays an important role in finite model theory (see [22]). In this graph, the nodes are the variables of the CQ and the edges consist of pairs of variables occurring in the same atom. As an example, Figure 2 depicts the Gaifman graphs of the specific CQs that we mentioned so far (Examples 3.4 and 3.5). A variable x that occurs in a Winner atom $\textsc{Winner}(x)$ is called a Winner *variable.* We say that the Winner atoms are *pairwise disconnected* if every two distinct Winner variables belong to different connected components of the Gaifman graph.

Example 5.2. As can be seen in Figure 2, the Winner atoms of the queries q_0, q_1 and q_2 from Example 3.4 are (pairwise) disconnected, but those of q_3 are not. In particular, this property is shared by every CQ with a single Winner atom, such as q^1_{wr} of Equation 2. In contrast, the property does not hold for the queries q^2_{wr} and q^3_{wr} in the figure. □

Theorem 5.3 concerns the class of UCQs such that each of their CQs has pairwise disconnected Winner atoms. An example of such a UCQ is the following query that asks whether there exist two winners who belong to different major parties and who disagree on the Trans-Pacific Partnership issue.

$$\begin{aligned}&\big(\textsc{Winner}(x)\,,\ \mathit{Cand}(x, \mathsf{R}, \_)\,,\ \mathit{Position}(x, \mathsf{TPP}, \mathsf{yes})\\ &\quad \textsc{Winner}(y)\,,\ \mathit{Cand}(y, \mathsf{D}, \_)\,,\ \mathit{Position}(y, \mathsf{TPP}, \mathsf{no})\big) \bigcup\\ &\big(\textsc{Winner}(x)\,,\ \mathit{Cand}(x, \mathsf{R}, \_)\,,\ \mathit{Position}(x, \mathsf{TPP}, \mathsf{no})\\ &\quad \textsc{Winner}(y)\,,\ \mathit{Cand}(y, \mathsf{D}, \_)\,,\ \mathit{Position}(y, \mathsf{TPP}, \mathsf{yes})\big)\end{aligned}$$

Theorem 5.3. *Let q be a UCQ such that the* Winner *atoms of each of its CQs are pairwise disconnected. If r is plurality or veto, then* Necessity(q, r) *is solvable in polynomial time.*

Proof. Let q be $q_1 \cup \cdots \cup q_\ell$, and consider an input $(D, \mathbf{P})$ to Necessity(q, r). We first make and justify some structural assumptions about q. For $i = 1, \ldots, \ell$, if q_i does not involve *any* Winner atom, then either $D \models q_i$ and then q is necessarily true on $(D, \mathbf{P})$, or $D \not\models q_i$ and then we can remove q_i from q without affecting the truth of q over the expansions. Therefore, we will assume that each q_i includes at least one Winner atom. We also assume that no q_i contains two identical Winner atoms (with the same Winner variable or constant), since we can clearly remove copies of atoms from a CQ while preserving equivalence.

With the above assumptions, we will assume that each q_i has the form

$$q_i() \ :- \ \varphi_i^1 \wedge \cdots \wedge \varphi_i^{k_i},$$

where each φ_i^j is a conjunction of atoms that includes precisely one Winner atom, and no variable is shared among two φ_i^j. We can make this assumption since the Winner atoms of q_i are pairwise disconnected. Hence, an expansion $D \sqcup \mathbf{T}$ satisfies q_i if and only if $D \sqcup \mathbf{T}$ satisfies each φ_i^j independently.

It thus follows that a counterexample to Necessity(q, r) is a completion $\mathbf{T}$ of $\mathbf{P}$ such that the expansion $D \sqcup \mathbf{T}$ violates at least one $\varphi_i^{j_i}$, for every $i = 1, \ldots, \ell$. So, we iterate through every combination $\varphi_1^{j_1}, \ldots, \varphi_\ell^{j_\ell}$ and test whether there exists a corresponding counterexample, that is, a completion $\mathbf{T}$ such that $D \sqcup \mathbf{T}$ violates *every* $\varphi_i^{j_i}$. If no such $\mathbf{T}$ is found, then q is necessarily true. So, fix $\varphi_1^{j_1}, \ldots, \varphi_\ell^{j_\ell}$. Recall that every $\varphi_i^{j_i}$ contains precisely one Winner atom. For $i = 1, \ldots, \ell$, let W_i be the set of candidates that satisfy $\varphi_i^{j_i}$, that is,

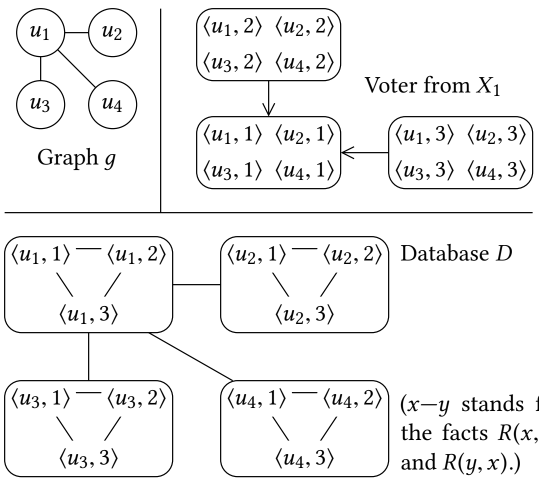
$$W_i \overset{\text{def}}{=\!=} \{c \in C \mid D \cup \{\textsc{Winner}(c)\} \models \varphi_i^{j_i}\}.$$

Let $W = W_1 \cup \cdots \cup W_\ell$. The question at hand is whether there exists a completion $\mathbf{T}$ such that $\mathbf{W}(r, \mathbf{T})$ is *disjoint* from W. This is exactly the complement of the problem Necessity(q^1_{wr}, r) when the R relation is equal to W. Hence, the theorem follows from the tractability of Necessity(q^1_{wr}, r) when r is the plurality rule [20] and when r is the veto rule (Lemma 5.1). □

In the next section, we discuss hardness and, in particular, show that Theorem 5.3 covers *all* tractable cases of pure positional scoring rules.

5.2 Hardness

The next result asserts that if r is a pure positional scoring rule other than plurality and veto, the necessity problem is already hard for the query q^1_{wr}.



u_1
u_2
u_3
u_4
Graph g
$\langle u_1,2\rangle$ $\langle u_2,2\rangle$ $\langle u_3,2\rangle$ $\langle u_4,2\rangle$
Voter from X_1
$\langle u_1,1\rangle$ $\langle u_2,1\rangle$ $\langle u_3,1\rangle$ $\langle u_4,1\rangle$
$\langle u_1,3\rangle$ $\langle u_2,3\rangle$ $\langle u_3,3\rangle$ $\langle u_4,3\rangle$
Database D
(x—y stands for the facts $R(x, y)$ and $R(y, x)$.)

Figure 3: An illustration of the reduction in the proof of Lemma 5.5 for $k = 2$.

Proposition 5.4. Necessity(q^1_{wr}, r) *is coNP-complete for every pure positional scoring rule r other than plurality and veto.*

Proof. We exhibit a reduction from the complement of the *possible unique winner* problem: given a partial profile $\mathbf{P}$ and a candidate c, determine whether c is a possible unique winner. By Theorem 2.3, this problem is NP-complete for every pure positional scoring rule r other than plurality and veto. Given $\mathbf{P}$ and c, we construct the election database $(D, \mathbf{P})$, where D consists of a single unary relation $R^D = C \setminus \{c\}$. For a completion $\mathbf{T}$ of $\mathbf{P}$, we have that $D \sqcup \mathbf{T} \models q^1_{\mathrm{wr}}$ if and only if $\mathbf{W}(r, \mathbf{T})$ includes a candidate other than c. Consequently, q^1_{wr} is necessary for $(D, \mathbf{P})$ if and only if c is *not* a possible unique winner in $\mathbf{P}$. □

By combining Theorem 5.3 and Proposition 5.4, we conclude that Necessity(q^1_{wr}, r) is tractable when r is plurality or veto, and intractable for every other pure positional scoring rule. Next, we show that the query q^2_{wr} is *harder* than q^1_{wr}, in the sense that q^2_{wr} is hard for *every* pure positional scoring rule, including plurality and veto. Hardness for plurality has been shown by Kimelfeld, Kolaitis, and Stoyanovich [20]. The next lemma completes the picture with the veto rule.

Lemma 5.5. *If r is the veto rule, then* Necessity(q^2_{wr}, r) *is coNP-complete.*

Proof. We exhibit a reduction from the complement of the *maximum independent set* problem. We are given as input a graph $G = (V, E)$ and a natural number k, and the goal is to determine whether G has an independent set of size k. We construct an instance $(D, \mathbf{P})$ of Necessity(q^2_{wr}, r), as illustrated in Figure 3. The partial voting profile $\mathbf{P}$ is defined as follows.

- The candidate set C consists of all pairs $\langle u, i \rangle$, where u is a node of G and $1 \leq i \leq k$. We partition C into *parties* $C_1, \ldots, C_k$ such that $C_i = \{\langle u, i \rangle \mid u \in V\}$.
- There are $k \times (|V| - 1)$ voters. Let $X_1, \ldots, X_k$ be a partitioning of the voters such that every X_i is of size $|V| - 1$.
- For $1 \leq i \leq k$, the voters in X_i have the same partial preference: they prefer every candidate in $C \setminus C_i$ to every candidate in C_i; there are no preferences within X_i and no preferences within $C \setminus X_i$.

The database D consists of one binary relation R^D that contains all candidate pairs $(\langle u, i \rangle, \langle u', i' \rangle)$ such that one of the following holds:

- $u = u'$ and $i \neq i'$;
- $u \neq u'$, but u is a neighbor of u' in g.

This concludes the construction. We complete the proof by proving correctness, that is, Necessity(q^2_{wr}, r) is false for $(D, \mathbf{P})$ if and only if G has an independent set of size k.

We observe that, in every completion $\mathbf{T}$ of $\mathbf{P}$, there are at least k winners. Indeed, only voters from C_i can veto (i.e., position as last) a candidate from X_i, and $|X_i| = |C_i| - 1$, and hence at least one candidate from C_i has no vetoes at all in $\mathbf{T}$. In particular, that candidate gets the highest possible score (namely, $k \times (|V| - 1)$) and must be a winner. We conclude that at least one candidates in C_i is a winner in $\mathbf{T}$, which proves the observation.

If G has an independent set $U = \{u'_1, \ldots, u'_k\}$ of size k, then we construct a completion $\mathbf{T}$ of $\mathbf{P}$ as follows. For $1 \leq i \leq k$, every voter in X_i vetoes a unique candidate from $C_i \setminus \{\langle u'_i, i \rangle\}$. Then we have that, in $\mathbf{T}$, every candidate $\langle u'_i, i \rangle$ gains the maximum score $k \times (|V| - 1)$, since no candidate vetoes $\langle u'_i, i \rangle$. Every other candidate c gains the score of $k \times (|V| - 1) - 1$, since precisely one voter vetoed c. The winners in $\mathbf{T}$ are the $\langle u'_i, i \rangle$. Since no tuple in R^D contains two nodes among the $\langle u'_i, i \rangle$, we get that $D \sqcup \mathbf{T} \not\models q^2_{\text{wr}}$.

Next, suppose that $\mathbf{T}$ is a completion of $\mathbf{P}$ where $D \sqcup \mathbf{T} \not\models q^2_{\text{wr}}$. By the way we constructed R^D, the winners must correspond to different nodes in g, and no two nodes are neighbours. From the above observation, it follows that g has an independent set of size at least k. □

We then establish the following result.

Theorem 5.6. *If $k > 1$, then* Necessity(q^k_{wr}, r) *is coNP-complete, for every pure positional scoring rule r.*

Proof. We prove the theorem for $k = 2$. For $k > 2$, coNP-hardness is proved by reducing Necessity(q^2_{wr}, r) to the problem Necessity(q^k_{wr}, r). Kimelfeld, Kolaitis, and Stoyanovich [20] showed that Necessity(q^2_{wr}, r) is coNP-complete when r is the plurality rule. Lemma 5.5 shows that this is also the case when r is the veto rule. For the remaining positional rules, we reduce Necessity(q^1_{wr}, r) to Necessity(q^2_{wr}, r) and use Proposition 5.4. The reduction is simple: given an instance $(D, \mathbf{P})$ of Necessity(q^1_{wr}, r), construct an instance $(D', \mathbf{P})$ of Necessity(q^2_{wr}, r) by setting the relation $R^{D'}$ to be $\{(c, c) \mid c \in R^D\}$. □

If $k > 2$, then we can drop the assumption that the rule r is pure and establish a hardness result that is much stronger than that of Theorem 5.6. To the best of our knowledge, this is the first general hardness result that applies to *every* positional scoring rule, whether pure or not. The proof of this result will appear in the full version of this paper.

Theorem 5.7. *If $k > 2$, then* Necessity(q^k_{wr}, r) *is coNP-complete for every positional scoring rule.*

It remains an open problem to determine whether or not Theorem 5.7 holds true for $k = 2$, that is, whether Necessity(q^2_{wr}, r) is coNP-complete for every positional scoring rule, thus extending Theorem 5.6 to arbitrary positional scoring rules.

6 QUERIES WITH WINNER AND UNIQUE-WINNER ATOMS

In this section, we discuss the possibility and necessity problems for UCQs that involve UWinner atoms. Thus, in this section, the term UCQ refers to a union of conjunctive queries whose atoms are Winner atoms or UWinner atoms or atoms with relation symbols from the schema $\mathcal{S}$.

We begin with a simple observation. Consider a CQ q that contains one or more UWinner atom. For an expansion $D \sqcup \mathbf{T}$ to satisfy q, the relation UWinner$^{D \sqcup \mathbf{T}}$ should be nonempty. By the semantics of UWinner it must hold that UWinner$^{D \sqcup \mathbf{T}}$ = Winner$^{D \sqcup \mathbf{T}}$ $= \{c\}$ for some candidate c. Moreover, a homomorphism from q to $D \sqcup \mathbf{T}$ must assign all the terms (variables and constants) that occur in UWinner and Winner atoms the same value (the unique winner). Consequently, we can unify these terms in q. More formally, let Θ_q be the set of all the terms in the UWinner and Winner atoms of q.

- If Θ_q contains no constants, we select a variable $x \in \Theta_q$ and replace in q every $y \in \Theta_q$ with x.
- If Θ_q contains a single constant c, we replace in q every variable $y \in \Theta_q$ with c.
- If Θ_q contains two or more constants, q is not satisfiable (hence, if q is one of the CQs of a UCQ, then it can be removed by the UCQ while preserving equivalence.

Therefore, in the remainder of this section, we will assume that if a CQ q contains at least one UWinner atom, then it contains a single UWinner atom and no Winner atoms. We will use this assumption for analyzing the complexity of both the possibility and the necessity problems.

6.1 Possible Answers

The following theorem generalizes Theorem 4.4 to allowing UWinner atoms.

Theorem 6.1. *Let q be a Boolean UCQ (that may contain* Winner *and* UWinner *atoms). If r is the plurality rule or the veto rule, then* Possibility(q, r) *is solvable in polynomial time.*

Proof. Consider an input $(D, \mathbf{P})$ for Possibility(q, r). As in the proof of Theorem 4.4, we may assume that q is a CQ. In that proof, we have covered the case where q does not contain UWinner atom, so we assume that q contains UWinner atoms. In view of the discussion in the beginning of this section, we may assume that q contains a single UWinner atom and no Winner atoms.

For a candidate c, we denote by D_c the database over $\mathcal{S}^{\mathrm{e}}$ where $R^{D_c} = R^D$ for every relation $R \in \mathcal{S}$ and UWinner^{D_c} = Winner$^{D_c} = \{c\}$. We have that a candidate c is a *possible unique winner* if and only if there is a completion $\mathbf{T}$ of $\mathbf{P}$ such that $D \sqcup \mathbf{T} = D_c$. Then, Possibility$(q, r)$ is true on $(D, \mathbf{P})$ if and only if q evaluates to true on D_c for some possible unique winner c. From Theorem 2.3, it follows that computing the set of possible unique winners, PU, can be done in polynomial time if r is the plurality rule or the veto rule. □

For the remaining pure positional scoring rules, the hardness of Possibility(q_c, r) and Possibility(q^k_{wr}, r) that we discussed in Section 4.2 continues to hold if we replace the Winner relation symbol with UWinner; this is because determining whether a candidate is a possible *unique* winner is also NP-hard (Theorem 2.3). On the other hand, consider the following query, which asks whether a winner exists.

$$q_{\exists\mathrm{w}}() \;:\!-\; \textsc{Winner}(x)$$

The possibility (as well as necessity) of this query is trivial, as it is always true. Yet, as we show next, it becomes NP-hard for some voting rules if Winner is replaced with UWinner:

$$q_{\exists\mathrm{uw}}() \;:\!-\; \textsc{UWinner}(x)$$

The problem Possibility$(q_{\exists\mathrm{uw}}, r)$ asks whether there is a completion with a unique winner, which is now nontrivial. We can show NP-hardness for the case where r is the k-approval rule, for $k > 1$, by adapting the proof of Xia and Conitzer [27, Theorem 2] for the NP-hardness of the possible unique-winner problem.

Theorem 6.2. *For every $k > 1$, if r is the k-approval rule, then* Possibility$(q_{\exists\mathrm{uw}}, r)$ *is NP-complete.*

Proof. Xia and Conitzer [27] prove that the problem of determining whether a candidate is a possible unique winner is NP-complete for k-approval, for every $k > 1$. They do so by building a reduction from 3-satisfiability. In more detail, given a propositional formula φ, they construct an instance $(\mathbf{P}, c)$ where φ is satisfiable if and only if c is a possible unique winner. In this construction, c is a necessary winner in $\mathbf{P}$. Hence, if there is *any* unique winner in a completion $\mathbf{T}$, then this unique winner must be c. Therefore, their reduction is also a reduction to the problem of whether a possible unique winner exists, that is, Possibility$(q_{\exists\mathrm{uw}}, r)$. □

Recall that, by Theorem 6.1, we have a polynomial-time algorithm for Possibility$(q_{\exists\mathrm{uw}}, r)$ when r is the plurality rule, i.e., when r is k-approval with $k = 1$; thus, the case of k-approval is settled for every $k \geq 1$. Nevertheless, the complexity of Possibility$(q_{\exists\mathrm{uw}}, r)$ remains an open problem for every positional scoring rule other than k-approval and veto.

6.2 Necessary Answers

We now consider the necessity problem for UCQs that involve the UWinner relation symbol. Our result here is the tractability for the plurality and veto rules, where we generalize Theorem 5.3 to allow for arbitrary disjuncts (CQs) with UWinner atoms.

So, suppose that r is the plurality rule or the veto rule. Let us start with a conjunctive query q that contains a UWinner atom. As mentioned in the beginning of the section, we may assume that q contains a single UWinner atom and no Winner atoms. Given an input $(D, \mathbf{P})$, let U be the set of all candidates c such that $D \sqcup \mathbf{T} \models q$, whenever c is a unique winner. Then Necessity(q, r) is true if and only if Possibility(q', r) is false for $(D', \mathbf{P})$ where q' is defined by

$$q'() \;:\!-\; \big(\textsc{Winner}(x)\,,\ \textsc{Winner}(y)\,,\ R_{\neq}(x, y)\big) \cup \big(\textsc{Winner}(z)\,,\ R_{\bar{U}}(z)\big)$$

and D' consists of the fact $R_{\neq}(c_1, c_2)$ for all candidates c_1 and c_2 such that $c_1 \neq c_2$, and the fact $R_{\bar{U}}(c)$ for all candidates c that are *not* in U. Note that q' states that either there are two distinct winners (hence, UWinner is empty) or there is a winner that violates q. Theorem 6.1 implies that Possibility(q', r) is solvable in polynomial time, thus Necessity(q, r) is solvable in polynomial time. In fact, the following generalization of Theorem 5.3 shows a stronger result, the proof will appear in the full version of this paper.

Theorem 6.3. *Let q be a UCQ where in each CQ q', either q' contains a* UWinner *atom or the* Winner *atoms are pairwise disconnected. If r is the plurality or the veto rule, then* Necessity(q, r) *is solvable in polynomial time.*

For other positional scoring rules, the complexity remains unresolved, even for the query $q_{\exists\mathrm{uw}}$, where the goal is to determine whether there is *always* a single winner (whose identity may differ from one completion to another).

7 FIXED PARAMETER TRACTABLE

Here, we investigate the possibility and necessity problems through the lens of parameterized complexity. We show that,

Table 2: Complexity of UCQs for positional scoring rules. Each row applies to *every* rule in the list. A lower bound for answers refers to the existence of a UCQ with the corresponding complexity. "U" stands for "Unique" and refers to UCQs with UWinner atoms and no Winner atoms, "D/U" stands for "Disconnected/Unique" and refers to the condition of Theorem 6.3, and "?" means that no complete classification is known.

Rule	*Possible Winners*	*Possible Answers*	*Necessary Winners*	*Necessary Answers*	*U Necessary Answers*	*D/U Necessary Answers*
plurality, veto	P	P (Thm 4.4, 6.1)	P	coNP-c. (Thm 5.6)	P (6.3)	P (Thm 5.3, 6.3)
pure \ {plurality, veto}	NP-c.	NP-c.	P	coNP-c. (Prop 5.4)	?	coNP-c. (Prop 5.4)
non-pure	?	?	P	coNP-c. (Thm. 5.7)	?	?

taking the number of candidates as a parameter, it is *Fixed Parameter Tractable* (FPT) to compute the possible and necessary answers to every fixed UCQ. Thus, we generalize the result by Betzler et al. [5] asserting that the possible-winner problem is FPT, where the parameter is the number of candidates. Towards this goal, we first prove the following generalization, which implies that, under fixed parameter tractability, we can actually produce all possible *winner sets*.

Theorem 7.1. *Let r be a positional scoring rule. The following decision problem is FPT with parameter the number $|C|$ of candidates : given a partial profile $\mathbf{P}$ over the set C of candidates, and a subset $S \subseteq C$, is there a completion $\mathbf{T}$ such that $\mathbf{W}(r, \mathbf{T}) = S$?*

Proof. We adopt the approach that has been used for designing FPT algorithms for computational social choice [5, 28], namely, an FPT reduction to the feasibility of an Integer Linear Program (ILP), where the parameter is the number of variables. This problem is known to be FPT via an algorithm designed by Lenstra [15] (and later improved by Kannan [16]).

We need some notation. We denote the partial profile $\mathbf{P}$ by $(P_1, \dots, P_n)$. For a partial order P over C, let V_P be the set $\{i \mid P_i = P\}$ (thus, V_P represents the set of voters that have the partial preference P) and let L_P be the set of all linear extensions of P (thus, L_P consists of total orders over C). Note that the V_P's form a partition of the set of voters. Recall that for a total order T over C and for a candidate c, we denote by $s(T, c)$ the score value that T assigns to c according to r; in other words, if c is the jth element of T, then $s(T, c) = r(m, j)$. We assume that C is nonempty (otherwise, the answer is trivially false), and we fix a candidate $c_0 \in C$.

Our ILP uses a variable $X_{P,T}$ for each partial order P and linear extension $T \in L_P$. An assignment to $X_{P,T}$ represents a (feasible) number of voters i in V_P that have their partial order (i.e., P_i, which is equal to P) completed into T. Our ILP specifies constraints that assert the feasibility of the $X_{P,T}$.

$$\text{For every } P \text{ and } T: \quad X_{P,T} \in \mathbb{N}$$
$$\text{For every } P: \quad \sum_{T \in L_P} X_{P,T} = |V_P|.$$

To simplify the presentation, we also use a variable Y_c, for every candidate c, that holds the total score gained by c. Hence, the Y_c are realized via the following set of constraints.

$$\text{For every } c \in C: \quad Y_c = \sum_P \sum_{T \in L_P} s(T, c) \cdot X_{P,T}.$$

In addition, the ILP contains constraints that assert that the score of c_0 is the same as that of every $c \in S$.

$$\text{For every } c \in S: \quad Y_{c_0} = Y_c.$$

Moreover, the ILP contains constraints that assert that the score of c_0 is strictly larger than that of every candidate who is not in S.

$$\text{For every } c \in C \setminus S: \quad Y_{c_0} > Y_c.$$

This completes the description of the ILP. To complete the proof, observe that the ILP has a solution if and only if there is a completion $\mathbf{T}$ of $\mathbf{P}$ such that $\mathbf{W}(r, \mathbf{T}) = S$. Also, observe that the number of variables $X_{P,T}$ and Y_c depends (in a single-exponential manner) only on the number of candidates. In fact, the same holds for the number of constraints. The only components of the program that depend on the number of voters are the constants $|V_P|$. □

From Theorem 7.1, we derive the following fixed parameter tractability result.

Corollary 7.2. *Let $\mathcal{S}$ be a schema and let q be a UCQ over $\mathcal{S}^e$. There is an FPT algorithm for computing the possible and necessary answers to q, where the parameter is the number of candidates.*

Proof. From Theorem 7.1 it follows that we can materialize all expansions $D \sqcup \mathbf{T}$ for the input $(D, \mathbf{P})$, by constructing every feasible relation $\text{Winner}^{D \sqcup \mathbf{T}}$ and its corresponding $\text{UWinner}^{D \sqcup \mathbf{T}}$. Then, by definition, the possible answers are the union of the $q(D \sqcup \mathbf{T})$, and the necessary answers are the intersection of the $q(D \sqcup \mathbf{T})$. □

A perusal of the proof of Theorem 7.1 reveals that Corollary 7.2 holds even for queries richer than UCQs. In fact, the corollary is true for every query q with a polynomial-time data complexity over ordinary databases (e.g., every safe query in relational calculus).

8 CONCLUDING REMARKS

We investigated the complexity of evaluating queries over election databases. We focused on the possibility and necessity problems for Boolean UCQs and positional scoring rules. The state of affairs is summarized in Table 2. While our study unveiled that these problems are often intractable, we have also established several different tractability results. In particular, the tractability side of the classification for the possible-winner problem [3, 4, 21, 27] generalizes to the possibility problem for UCQs under the plurality rule and the veto rule. In contrast, the tractability of the necessary-winner problem under every positional scoring rule [27] does not generalize to the necessity problem for Boolean UCQs; this reveals a fundamental difference between computational social choice and election databases. Finally, we showed that the possibility and necessity problems are fixed-parameter tractable when the parameter is the number of candidates; this is an appropriate parameter in such situations as political elections, where there are many voters but only a handful of candidates.

We have already listed several different open problems. We conclude the paper with additional open problems and directions for future investigation. First, as we mentioned in the body of the paper, some of our results go beyond the class of UCQs; nonetheless, very little is known at present about more expressive classes of queries. Furthermore, we have not considered voting rules that fall outside the class of positional scoring rules, such as Copeland's method, Simpson's rule, and Bucklin voting[2]. Future research also includes queries that involve multiple elections and multiple voting rules [20].

Another direction for future research is that of probabilistic voters, adopting statistical models of preferences, such as the uniform distribution over the linear extensions, Mallows [24] and the Repeated Insertion Model (RIM) [11]. The analog of computing necessary/possible winners is to compute the probability that a candidate wins [2, 13, 17, 23]. In the framework of election databases, the analog is *probabilistic query answering*, where the goal is to compute the marginal probability of query answers [10, 26]. The evaluation of CQs over databases with RIM preferences has been studied in past research [9, 19], but without the angle of elections and social choice.

[2]See, e.g., [27] for a description of these and other voting rules.

ACKNOWLEDGMENTS

This work was supported by NSF Grant 1814152 and BSF Grant 2017753. The work of Kimelfeld and Tibi was also supported by ISF Grant 1295/15.

REFERENCES

[1] Antoine Amarilli, Mouhamadou Lamine Ba, Daniel Deutch, and Pierre Senellart. 2017. Possible and Certain Answers for Queries over Order-Incomplete Data. In *TIME (LIPIcs)*, Vol. 90. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 4:1–4:19.
[2] Yoram Bachrach, Nadja Betzler, and Piotr Faliszewski. 2010. Probabilistic Possible Winner Determination. In *AAAI*. AAAI Press.
[3] Dorothea Baumeister and Jörg Rothe. 2012. Taking the final step to a full dichotomy of the possible winner problem in pure scoring rules. *Inf. Process. Lett.* 112, 5 (2012), 186–190.
[4] Nadja Betzler and Britta Dorn. 2010. Towards a dichotomy for the Possible Winner problem in elections based on scoring rules. *J. Comput. Syst. Sci.* 76, 8 (2010), 812–836.
[5] Nadja Betzler, Susanne Hemmann, and Rolf Niedermeier. 2009. A Multivariate Complexity Analysis of Determining Possible Winners Given Incomplete Votes. In *IJCAI*. 53–58.
[6] Craig Boutilier and Jeffrey S. Rosenschein. 2016. Incomplete Information and Communication in Voting. In *Handbook of Computational Social Choice*. 223–258.
[7] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia (Eds.). 2016. *Handbook of Computational Social Choice*. Cambridge University Press. https://doi.org/10.1017/CBO9781107446984
[8] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia. 2016. Introduction to Computational Social Choice. In *Handbook of Computational Social Choice*. 1–20.
[9] Uzi Cohen, Batya Kenig, Haoyue Ping, Benny Kimelfeld, and Julia Stoyanovich. 2018. A Query Engine for Probabilistic Preferences. In *SIGMOD Conference*. ACM, 1509–1524.
[10] Nilesh N. Dalvi and Dan Suciu. 2004. Efficient Query Evaluation on Probabilistic Databases. In *VLDB*. Morgan Kaufmann, 864–875.
[11] Jean-Paul Doignon, Aleksandar Pekec, and Michel Regenwetter. 2004. The repeated insertion model for rankings: Missing link between two subset choice models. *Psychometrika* 69, 1 (2004), 33–54.
[12] Jack Edmonds and Ellis L. Johnson. 2001. Matching: A Well-Solved Class of Integer Linear Programs. In *Combinatorial Optimization - Eureka, You Shrink!, Papers Dedicated to Jack Edmonds (Lecture Notes in Computer Science)*, Vol. 2570. Springer, 27–30.
[13] Noam Hazon, Yonatan Aumann, Sarit Kraus, and Michael Wooldridge. 2012. On the evaluation of election outcomes under uncertainty. *Artificial Intelligence* 189 (2012), 1 – 18.
[14] Marie Jacob, Benny Kimelfeld, and Julia Stoyanovich. 2014. A System for Management and Analysis of Preference Data. *PVLDB* 7, 12 (2014), 1255–1258.
[15] Hendrik W. Lenstra Jr. 1983. Integer Programming with a Fixed Number of Variables. *Math. Oper. Res.* 8, 4 (1983), 538–548.
[16] Ravi Kannan. 1987. Minkowski's Convex Body Theorem and Integer Programming. *Math. Oper. Res.* 12, 3 (1987), 415–440.
[17] Batya Kenig and Benny Kimelfeld. 2019. Approximate Inference of Outcomes in Probabilistic Elections. To appear in AAAI.
[18] Batya Kenig, Benny Kimelfeld, Haoyue Ping, and Julia Stoyanovich. 2017. Querying Probabilistic Preferences in Databases. In *PODS (PODS '17)*. ACM, New York, NY, USA, 21–36.

[19] Batya Kenig, Benny Kimelfeld, Haoyue Ping, and Julia Stoyanovich. 2017. Querying Probabilistic Preferences in Databases. In *PODS*. 21–36.

[20] Benny Kimelfeld, Phokion G. Kolaitis, and Julia Stoyanovich. 2018. Computational Social Choice Meets Databases. In *IJCAI*. ijcai.org, 317–323.

[21] Kathrin Konczak and Jérôme Lang. 2005. Voting procedures with incomplete preferences. In *Proc. IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling*, Vol. 20.

[22] Leonid Libkin. 2004. *Elements of Finite Model Theory*. Springer.

[23] Tyler Lu and Craig Boutilier. 2011. Vote Elicitation with Probabilistic Preference Models: Empirical Estimation and Cost Tradeoffs. In *ADT (Lecture Notes in Computer Science)*, Vol. 6992. Springer, 135–149.

[24] C. L. Mallows. 1957. Non-Null Ranking Models. I. *Biometrika* 44, 1-2 (1 June 1957), 114–130.

[25] Yossi Shiloach. 1981. Another Look at the Degree Constrained Subgraph Problem. *Inf. Process. Lett.* 12, 2 (1981), 89–92.

[26] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. 2011. *Probabilistic Databases*. Morgan & Claypool Publishers.

[27] Lirong Xia and Vincent Conitzer. 2011. Determining Possible and Necessary Winners Given Partial Orders. *J. Artif. Intell. Res.* 41 (2011), 25–67.

[28] Yongjie Yang. 2014. Election Attacks with Few Candidates. In *ECAI (Frontiers in Artificial Intelligence and Applications)*, Vol. 263. IOS Press, 1131–1132.