Computational Social Choice Meets Databases

Benny Kimelfeld[1], **Phokion G. Kolaitis**[2,3], **Julia Stoyanovich**[4]
[1] Technion, Israel
[2] UC Santa Cruz, USA
[3] IBM Research-Almaden, USA
[4] Drexel University, USA
bennyk@cs.technion.ac.il, kolaitis@cs.ucsc.edu, stoyanovich@drexel.edu

Abstract

We develop a novel framework that aims to create bridges between the computational social choice and the database management communities. This framework enriches the tasks currently supported in computational social choice with relational database context, thus making it possible to formulate sophisticated queries about voting rules, candidates, voters, issues, and positions. At the conceptual level, we give rigorous semantics to queries in this framework by introducing the notions of necessary answers and possible answers to queries. At the technical level, we embark on an investigation of the computational complexity of the necessary answers. In particular, we establish a number of results about the complexity of the necessary answers of conjunctive queries involving the plurality rule that contrast sharply with earlier results about the complexity of the necessary winners under the plurality rule.

1 Introduction

Social choice theory is concerned with the aggregation of preferences expressed by the members of a society to arrive at a collective decision. The origins of social choice theory are often traced to the work of Jean-Charles de Borda and Marquis de Condorcet in the 18th Century, even though it is now known that Condorcet's voting rule had been already proposed by Ramon Llull in the 13th Century [Hägele and Pukelsheim, 2001]. During the past two decades, social choice theory has been examined under the algorithmic lens, and computational social choice (COMSOC) has emerged as an interdisciplinary research area that combines insights and methods from mathematics, economics, logic, and computer science. The COMSOC community has carried out an in-depth investigation of computational aspects of voting and preference aggregation in an election or a poll. Since preferences are often only partially expressed, the notions of *necessary winners* and *possible winners* were formulated by Konczak and Lang [Konczak and Lang, 2005], as the candidates who win in every (respectively, in at least one) completion of the given partial preferences. Subsequent investigations produced a classification of the computational complexity of the necessary and possible winners for a variety of voting rules [Baumeister and Rothe, 2012; Betzler and Dorn, 2010; Xia and Conitzer, 2011].

Here, we bring forth a novel framework that aims to create bridges between the COMSOC community and the data management community. We enrich the kinds of data analysis tasks that COMSOC methods currently support by incorporating context about candidates, voters, issues, and positions, thus going well beyond the mere determination of winners. To achieve this, we accommodate COMSOC primitives within a relational database framework, enabling the formulation and evaluation of sophisticated queries.

Motivating Example. A *preference database* [Jacob *et al.*, 2014] is depicted in Figure 1. The relations CAND and VOTER contain demographic information about political candidates and voters, while SUPPORTS and OPPOSES list positions of candidates on campaign issues, and BALLOT records results of an election or a poll. Observe that BALLOT specifies preferences of a voter in an election with pairwise comparisons: the meaning of the tuple $(\texttt{Oct-5}, \texttt{Ann}; \texttt{Clinton}, \texttt{Trump})$ is that voter Ann prefers Clinton to Trump when polled on October 5.

Preferences of voters may be incomplete. In particular, Ann states that she prefers both Clinton and Johnson to Trump, but does not specify a relative preference between Clinton and Johnson. The incomplete preference relation BALLOT gives rise to four completions, B_1 through B_4, in which each session is associated with a complete ranking (a total order) over the candidates that is consistent with the partial preference in BALLOT.

A data analyst may want to aggregate the votes of Ann and Bob to determine the *winner* of the `Oct-5` election—the candidate deemed most desirable by the voters—using a *voting rule*. In this paper, we focus on *positional scoring rules*—voting rules that assign a score to each candidate based on the candidate's position in a ranking and then sum the scores across all rankings.

Our example involves two voting rules: plurality, which assigns a score of 1 to the top candidate in each ranking and 0 to all other candidates, and Borda, which assigns a score of $m - r$ to the candidate at position r out of m. The sum of scores of each candidate in each completion is shown in the bottom left table in Figure 1.

We are concerned with answering the following kind of question: *Is there a winner according to the plurality rule in the October 5 election who is pro-choice?* This question is phrased in logic-rule style below as:

$$q_1() :- \textsc{Winner}(\texttt{plurality}, \texttt{Oct-5}, c), \textsc{Supports}(c, \texttt{pro-choice})$$

What is the meaning of such a query posed on a database with partial preferences? In Section 3, we propose formal semantics that generalizes the concepts of *necessary* and *possible winners* from computational social choice to those of *necessary* and *possible answers* in a partial preference database.

Returning to the example in Figure 1, if we consider Borda's rule, Clinton is the only necessary winner: she is the sole winner in B_1, B_2 and B_3, and is among the winners in B_4. If we consider the plurality rule, we find the following winners in each completion: B_1: Clinton, B_2: Clinton and Trump, B_3: Clinton and Johnson, B_4: Johnson and Trump. Consequently, the set of necessary winners under plurality is empty. Observe that, although no candidate is a necessary winner under plurality, it is the case that *at least one* of the winners in each completion is pro-choice. Thus, under the plurality rule, the query q_1 is *necessary*, i.e., *true* is a necessary answer of q_1.

The preceding example illustrates the difference that context makes and points to the richness brought by combining social choice and data management.

Contributions. At the conceptual level, we develop a framework that combines social choice with database management, thus making it possible to study social choice problems in the context of additional information about voters, candidates, and issues. In particular, we give rigorous semantics to queries in this framework by introducing the notions of the *necessary answers* and the *possible answers*. At the technical level, we embark on an investigation of the computational complexity of query evaluation in this framework. In particular, we establish a number of results about the necessary answers of queries involving the plurality rule that stand in sharp contrast to results about the necessary winners under the plurality rule. Specifically, it is well known that there is a polynomial-time algorithm for computing the necessary winners under the plurality rule (in fact, such an algorithm exists for every pure positional scoring rule) [Konczak and Lang, 2005; Xia and Conitzer, 2011]. In contrast, we show that there are natural conjunctive queries involving database relations and winners under the plurality rule such that computing their necessary answers is a coNP-complete problem. Furthermore, we show that this hardness result extends to a large class of positional scoring rules. On the side of tractability, we give a polynomial-time algorithm for computing the necessary answers of conjunctive queries that involve the plurality rule and have the property that the winner atoms belong to different connected components of the query.

2 Preliminaries

Relational databases and conjunctive queries. A *schema* is a collection of *relation symbols*, each having an associated *signature*, which is a sequence of attribute names. A *database* instantiates each relation symbol with a corresponding relation (table). We will use the database in Figure 1 as our running example.

A *query* is a function that maps every database into a relation. More formally, a query has an associated input schema and an output signature, and it maps every database over the input schema into a relation over the output signature. If D is a database and q is a query, then $q(D)$ denotes the relation resulting by evaluating q on D; each tuple in $q(D)$ is referred to as an *answer* to q on D. In this paper, we study *conjunctive queries*, which correspond to the fragment of first-order logic obtained from atomic formulas using conjunction and existential quantification. Conjunctive queries are also known as select-project-join (SPJ) queries, and are among the most frequently asked database queries.

We will write queries as logic rules with a body and a head. For example, consider the following query:

$$q(c) :- \textsc{Voter}(\texttt{Ann}, s, e), \textsc{Cand}(c, p, s, a), a > \texttt{65}$$

This query computes candidates who are older than 65 and whose sex is the same as that of voter Ann, and will return a single tuple, `Clinton`, when evaluated over the database in Figure 1. Note that `Ann` and `65` are constants in q, while a, c, e, p, and s are variables. The variables e, p, s that occur in the body, but not in the head, of the query are existentially quantified.

A *Boolean* query is a query that has no free variables, hence it stands for a yes/no (true/false) question about the database. For example, the Boolean query

$$q'() :- \textsc{Voter}(\texttt{Ann}, s, e), \textsc{Cand}(c, p, s, a), a > \texttt{65}$$

asks whether or not there is a candidate who is older than 65 and whose sex is the same as that of voter `Ann`.

Conjunctive query evaluation has been a central topic of research in the database management community (see [Abiteboul *et al.*, 1995]). In particular, it is well known that, for every fixed conjunctive query q, there is a polynomial-time algorithm that, given a database D, computes $q(D)$.

Incomplete databases and possible worlds. Various notions of database incompleteness have been studied in depth for several decades. Common to these is the notion of *possible worlds*: these include the completions of incomplete databases and the solutions in data exchange and data integration [Fagin *et al.*, 2005; Imielinski and Jr., 1984; Lenzerini, 2002]. Query answering is a central challenge studied in these frameworks, where the goal is to find the *certain* answers, i.e., the answers obtained on every completion or on every solution. Additionally, a *possible answer* is an answer that is obtained on at least one possible world. More formally, if $\mathbf{W}$ denotes the set of possible worlds of the database representation at hand, then the set of certain answers to the query q is the intersection $\bigcap_{D\in\mathbf{W}} q(D)$, while the set of possible answers to q is the union $\bigcup_{D\in\mathbf{W}} q(D)$.

Voting profiles and voting rules. Let $C = \{c_1, \ldots, c_m\}$ be a set of *candidates* (or *alternatives*) and let $V = \{v_1, \ldots, v_n\}$ be a set of voters. A *complete voting profile*

CAND

cand	party	sex	edu	age
Clinton	D	F	JD	70
Johnson	L	M	BS	64
Trump	R	M	BS	71

SUPPORTS

cand	issue
Clinton	gun-control
Clinton	pro-choice
Johnson	pro-choice

OPPOSES

cand	issue
Johnson	gun-control
Trump	gun-control
Trump	pro-choice

VOTER

voter	sex	edu
Ann	F	MS
Bob	M	BS

BALLOT

election	voter	lcand	rcand
Oct-5	Ann	Clinton	Trump
Oct-5	Ann	Johnson	Trump
Oct-5	Bob	Clinton	Johnson
Oct-5	Bob	Trump	Johnson

A possible completion B_1 of BALLOT

election	voter	ranking
Oct-5	Ann	Clinton $\succ$ Johnson $\succ$ Trump
Oct-5	Bob	Clinton $\succ$ Trump $\succ$ Johnson

A possible completion B_2 of BALLOT

election	voter	ranking
Oct-5	Ann	Clinton $\succ$ Johnson $\succ$ Trump
Oct-5	Bob	Trump $\succ$ Clinton $\succ$ Johnson

A possible completion B_3 of BALLOT

election	voter	ranking
Oct-5	Ann	Johnson $\succ$ Clinton $\succ$ Trump
Oct-5	Bob	Clinton $\succ$ Trump $\succ$ Johnson

A possible completion B_4 of BALLOT

election	voter	ranking
Oct-5	Ann	Johnson $\succ$ Clinton $\succ$ Trump
Oct-5	Bob	Trump $\succ$ Clinton $\succ$ Johnson

	B_1		B_2		B_3		B_4	
cand	pl	Brd	pl	Brd	pl	Brd	pl	Brd
Clinton	2	4	1	3	1	3	0	2
Johnson	0	1	0	1	1	2	1	2
Trump	0	1	1	2	0	1	1	2

Candidate scores in the possible completions under the plurality (pl) and Borda (Brd) rules

Figure 1: An example of a preference database.

is a tuple $\mathbf{T} = (T_1, \ldots, T_n)$, where each T_i is a total order of the set C of candidates representing the ranking (preference) of voter v_j on the candidates in C.

Positional scoring rules constitute a large and extensively studied class of voting rules. Each positional scoring rule on a set of m candidates is specified by a scoring vector $\mathbf{a} = (a_1, \ldots, a_m)$ of non-negative integers, called the *score values*, such that $a_1 \geq a_2 \geq \ldots \geq a_m$. To avoid trivialities, we assume that there are at least two different score values. Suppose that $\mathbf{T} = (T_1, \ldots, T_n)$ is a total voting profile. The score $s(T_i, c)$ of a candidate c on T_i is the value a_s where s is the position of candidate c in T_i. When the positional scoring rule r is applied to $\mathbf{T} = (T_1, \ldots, T_n)$, it assigns to each candidate c the sum $\sum_{i=1}^{n} s(T_i, c)$ as the *score* of c. The set $\mathsf{W}(r, \mathbf{T})$ of the winners consists of the candidates who achieved maximum score.

From now on, we focus on positional scoring rules that are defined for every number m of candidates. Thus, a *positional scoring rule* is an infinite sequence $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m, \ldots$ of scoring vectors such that each $\mathbf{a}_m$ is a scoring vector of length m. Alternatively, a positional scoring rule is a function r that takes as argument a pair (m, s) of positive integers with $s \leq m$ and returns as value a non-negative integer $r(m, s)$ such that $r(m, 1) \geq r(m, 2) \ldots \geq r(m, m)$. We will also assume that the function r is computable in polynomial time. This implies that the winners can be computed in polynomial time. As examples, the *plurality* rule is given by the infinite sequence of scoring vectors of the form $(1, 0, \ldots, 0)$, the 2-*approval* rule is given by the infinite sequence of scoring vectors of the form $(1, 1, 0, \ldots, 0)$, and the *Borda* rule is given by the infinite sequence of scoring vectors of the form $(m-1, m-2, \ldots, 0)$.

Much on the literature makes the assumption that the rules are also *pure*, which means that the scoring vector $\mathbf{a}_{m+1}$ of length $(m+1)$ is obtained from the scoring vector $\mathbf{a}_m$ of length m by inserting a score in some position of the scoring vector $\mathbf{a}_m$, provided that the decreasing order of score values is maintained. The plurality, 2-approval and Borda rules are all pure.

Partial orders. A *preference* over a collection of items is a linear order that ranks the items from the most to the least preferred. Often, our knowledge about the preference is only partial. Missing information in preferences is commonly modeled using a *partial order*, that is, a relation that is reflexive, transitive, and antisymmetric, but not necessarily total. A *completion* of a partial order is a total order that extends that partial order. A partial order may have exponentially many completions.

Partial voting profiles, necessary and possible winners. A *partial voting profile* is a tuple $\mathbf{P} = (P_1, \ldots, P_n)$, 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 $\mathbf{P} = (P_1, \ldots, P_n)$ is a complete voting profile $\mathbf{T} = (T_1, \ldots, T_n)$ such that each T_i is a completion of the partial order P_i. The notions of *necessary* and *possible* winners were introduced by Konczak and Lang [Konczak and Lang, 2005].

Let r be a voting rule and $\mathbf{P}$ a partial voting profile. The set $\mathsf{NW}(r, \mathbf{P})$ of the *necessary winners* with respect to r and $\mathbf{P}$ is the intersection of the sets $\mathsf{W}(r, \mathbf{T})$, where $\mathbf{T}$ varies over all completions of $\mathbf{P}$. In other words, a candidate c is a *necessary winner* with respect to r and P, if c is a winner in $\mathsf{W}(r, \mathbf{T})$ for every completion $\mathbf{T}$ of $\mathbf{P}$.

The set $\mathsf{PW}(r, \mathbf{P})$ of the *possible winners* with respect to r and $\mathbf{P}$ is the union of the sets $\mathsf{W}(r, \mathbf{T})$, where $\mathbf{T}$ varies over all completions of $\mathbf{P}$. In other words, a candidate c is a *possible winner* with respect to r and $\mathbf{P}$, if c is a winner in $\mathsf{W}(r, \mathbf{T})$ for at least one completion $\mathbf{T}$ of $\mathbf{P}$.

On the face of the definitions, computing necessary and possible winners requires exponential time, since, in general, 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 of the necessary and the possible winners for *all* pure positional scoring rules was obtained through the work of Konczak and Lang [Konczak and Lang, 2005], Xia and Conitzer [Xia and Conitzer, 2011], Betzler and Dorn [Betzler and Dorn, 2010], and Baumeister and Rothe [Baumeister and Rothe, 2012].

Theorem 1. [Classification Theorem] *The following hold.*

- *For every pure positional scoring rule r, there is a polynomial-time algorithm for computing the set $NW(r, \mathbf{P})$ of necessary winners, given a partial voting profile $\mathbf{P}$.*
- *If r is the plurality rule or r is the veto rule, then there is a polynomial-time algorithm for computing the set $PW(r, \mathbf{P})$ of possible winners, given a partial voting profile $\mathbf{P}$. For all other pure positional scoring rules, the following problem is NP-complete: given a partial voting profile $\mathbf{P}$ and a candidate c, is c a possible winner with respect to r and $\mathbf{P}$?*

3 A Relational Framework for COMSOC

Preference schemas and sessions. We adopt and refine the formalism for preference databases proposed in [Jacob *et al.*, 2014] and explored further in [Kenig *et al.*, 2017]. A *preference* schema consists of *preference* relation symbols and *ordinary* relation symbols. The attribute list of each preference symbol P is of the form $(\beta; A_l, A_r)$, where β is a list of attributes called *session signature*, and A_l, A_r are attributes with candidates as values. The intent is that if $(\mathbf{b}; c, d)$ is a tuple in an instance of the preference symbol P, then candidate c is preferred to candidate d in the session determined by $\mathbf{b}$. In what follows, we will assume that each session signature β consists of the attributes election and voter. The intuition is that values for election stand for particular events, such as an election (e.g., the election for city council members) or a poll (e.g., a poll on presidential candidates taken on October 5), while the values for voter range over the possible voters.

Incorporating voting rules and winners. We augment the preference schema at hand with a new ternary relation symbol WINNER. The signature of WINNER is the triple (rule, election, candidate). The intent is that rule has voting rules as values, while election and candidate have elections (or polls) and candidates as values, respectively. The new relation symbol WINNER can now be used in standard relational database queries (e.g., SQL queries). To illustrate this, we return to our running example (the preference database in Fig. 1) and give examples of Boolean queries over this database.

- Query q_1: Is there a winner according to the plurality rule in the October 5 election who is pro-choice?

$$\begin{aligned} q_1() :- &\ \textsc{Winner}(\texttt{plurality}, \texttt{Oct-5}, c), \\ &\ \textsc{Supports}(c, \texttt{pro-choice}) \end{aligned}$$

- Query q_2: Are there a winner according to the Borda rule and a winner according to the 2-approval rule such that the former supports gun control and the latter opposes it?

$$\begin{aligned} q_2() :- &\ \textsc{Winner}(\texttt{Borda}, \texttt{Oct-5}, c), \\ &\ \textsc{Winner}(\texttt{2-approval}, \texttt{Oct-5}, d), \\ &\ \textsc{Supports}(c, \texttt{gun-control}), \\ &\ \textsc{Opposes}(d, \texttt{gun-control}) \end{aligned}$$

Necessary and possible answers. What is the semantics of queries, such as the three preceding ones, in a framework that allows for partial preferences, voting rules, and winners? We propose two different semantics of queries over partial preference databases, namely, the *necessary answers* and the *possible answers*.

Let D be a partial preference database. Then D gives rise to a partial voting profile $\mathbf{P}(D)$ consisting of the partial orders that correspond to the identifiers of the session signature. For example, the pair $(\texttt{Oct-5}, \texttt{Ann})$ gives rise to the partial order that expresses the (partial) preferences of Ann in the October 5 election (or poll).

Let $\widehat{D}$ be a total preference database. We say that $\widehat{D}$ is a *completion of* D if $\widehat{D}$ is obtained from D by completing all partial preferences into total ones. Thus, $\widehat{D}$ is a completion of D iff $\widehat{D}$ and D agree on the ordinary relation symbols and, for each preference relation symbol, the total voting profile $\mathbf{T}(\widehat{D})$ arising from $\widehat{D}$ is a completion of the partial voting profile $\mathbf{P}(D)$ arising from D.

Definition 1. Let q be a database query over the preference schema augmented with relation symbol WINNER.

- The *necessary answers* of q on D, denoted $\text{NA}(q, D)$, is the intersection $\bigcap q(\widehat{D})$ of the answers $q(\widehat{D})$ of q on $\widehat{D}$, as $\widehat{D}$ ranges over all completions of D.
- The *possible answers* of q on D, denoted $\text{PA}(q, D)$, is the union $\bigcup q(\widehat{D})$ of the answers $q(\widehat{D})$ of q on $\widehat{D}$, as $\widehat{D}$ ranges over all completions of D.

If q contains an atom of the form $\textsc{Winner}(r, e, c)$, then this atom is evaluated on $\widehat{D}$ by applying the voting rule r on the total voting profile $\mathbf{T}(\widehat{D})$. Thus, $\textsc{Winner}(r, e, c)$ evaluates to *true* on $\widehat{D}$ if and only if c belongs to the set $\mathsf{W}(r, \mathbf{T}(\widehat{D}))$ of the winners according to rule r and the total voting profile $\mathbf{T}(\widehat{D})$. In this evaluation, only the part of the voting profile $\mathbf{T}(\widehat{D})$ that is associated with the election e is needed.

Clearly, the preceding notions of necessary and possible answers coincide with those of certain and possible answers in the framework of incomplete databases, where the set $\mathcal{W}$ of possible worlds is the set of all completions $\widehat{D}$ of a given partial preference database D.

If q is a Boolean query, we say that q is *necessary on D* if $q(\widehat{D})$ is true for every completion $\widehat{D}$, and *possible on D* if $q(\widehat{D})$ is true for at least one completion $\widehat{D}$. We denote by $\mathsf{Necessity}(q)$ the decision problem: given D, is q necessary on D? Similarly, we denote by $\mathsf{Possibility}(q)$ the decision problem: given D, is q possible on D?

We conclude this section by pointing out that our framework is different from other approaches that have explored the interaction between databases and social choice. For example, Konczak [Konczak, 2006] investigated the computation of necessary and possible winners via logic programming (however, that work does not involve the concepts of necessary answers and possible answers of queries considered here). In a different direction, Lukasiewicz et al. [Lukasiewicz *et al.*, 2014] investigated top-k queries in databases using rankings whose computation involve the aggregation of partial preferences.

4 Complexity of Necessary Answers

How difficult is it to compute the necessary answers and the possible answers of a fixed conjunctive query q, given a partial preference database D? As regards upper bounds, we can consider every candidate tuple of values from the domain of D, and test whether this tuple is indeed a necessary or possible answer. If the voting rules occurring in q are such that their winners are computable in polynomial time, then, for every completion $\widehat{D}$ of D, we have that $q(\widehat{D})$ can be evaluated in polynomial time in the size of $\widehat{D}$. Consequently, deciding whether a tuple is a necessary answer of q is in coNP, while deciding whether it is possible is in NP.

Here, we will investigate the computational complexity of the necessary answers of conjunctive queries. We begin by considering several motivating examples.

Example 1. Assume that q is an atomic query of the form $\textsc{Winner}(r, e, c)$. Computing the necessary and the possible answers of q is the same as computing the necessary and the possible winners according to rule r. Thus, if r is a pure positional scoring rule, then this query is accounted for by the Classification Theorem 1 discussed earlier. ☐

Example 2. Let q be the query q_1 encountered earlier:

$$q_1() \;{:}-\; \textsc{Winner}(\texttt{plurality}, \texttt{Oct-5}, c),\\ \textsc{Supports}(c, \texttt{pro-choice})$$

A moment's reflection reveals that there is a difference between the problem of deciding whether q is possible and the problem of deciding whether q is necessary.

Indeed, to determine whether q_1 is possible on a partial preference database D, it is enough to compute the set $\mathrm{PW}(\texttt{plurality}, \mathbf{P}(D))$ of the possible winners with respect to the plurality rule and the partial voting profile $\mathbf{P}(D)$, and then intersect this set with the set of the pro-choice candidates. Since the possible winners with respect to the plurality rule are computable in polynomial time, it follows that $\mathsf{Possibility}(q_1)$ is decidable in polynomial time.

In other words, the possibility of q_1 can be rewritten to a query that involves the possible winners $\mathrm{PW}(\texttt{plurality}, \mathbf{T}(D))$. Concretely, $\mathsf{Possibility}(q_1)$ is equivalent to the query

$$q_1'() \;{:}-\; (c \texttt{ IN } \mathrm{PW}(\texttt{plurality}, \mathbf{T}(D))),\\ \textsc{Supports}(c, \texttt{pro-choice})\,.$$

In contrast, $\mathsf{Necessity}(q_1)$ cannot be rewritten (at least in a straightforward way) to a computation involving the necessary winners $\mathrm{NW}(\texttt{plurality}, \mathbf{T}(D))$. In particular, $\mathsf{Necessity}(q_1)$ is *not* equivalent to the query

$$q_1''() \;{:}-\; (c \texttt{ IN } \mathrm{NW}(\texttt{plurality}, \mathbf{T}(D))),\\ \textsc{Supports}(c, \texttt{pro-choice})\,.$$

Indeed, for every completion $\widehat{D}$ of D, there may exist a pro-choice winner (thus, q_1 is necessary on D), but there may be different winners for different completions and, as a result, there may exist no necessary winner who is also pro-choice (thus, the query q_1'' evaluates to *false* on D).

Our results, however, will imply that $\mathsf{Necessity}(q_1)$ is decidable in PTIME. ☐

Example 3. Assume that q is the query q_3: Are there two winners with different positions on at least one issue?

$$q_3() \;{:}-\; \textsc{Winner}(\texttt{plurality}, e, c), \textsc{Supports}(c, i),\\ \textsc{Winner}(\texttt{plurality}, e, d), \textsc{Opposes}(d, i)$$

Note that the query q_3 involves two $\textsc{Winner}$ atoms and that the candidates c and d occurring in these atoms are linked via the variable i in the two ordinary atoms of q_3.

Our results will imply that $\mathsf{Necessity}(q_3)$ is coNP-complete. Thus, Examples 2 and 3 demonstrate that, when bringing together preferences, voting rules, and relational data in a unifying framework, we are in a new state of affairs in which methods and results from computational social choice need not apply directly. ☐

4.1 Complexity Results for the Plurality Rule

We present several complexity results for Boolean conjunctive queries. In what follows in this subsection, we assume that all queries considered involve the plurality rule and some fixed election, which appears as a constant value in the queries. Hence, each $\textsc{Winner}$ atom has at most one variable, which stands for a winning candidate; we refer to such variable as a *winner* variable. We begin with a tractability result.

Theorem 2. *If q is a Boolean conjunctive query consisting of a single* $\textsc{Winner}$ *atom and ordinary atoms, then* $\mathsf{Necessity}(q)$ *is decidable in polynomial time.*

Hint of Proof: We reduce the problem to the following generalization of the necessary-winner problem: Given a partial voting profile and a subset X of candidates, determine whether X overlaps with the set of winners in every completion. In turn, we reduce this problem to maximum matching in a bipartite graph. ☐

As a direct corollary of Theorem 2, for the query q_1 in Example 2, we have that $\mathsf{Necessity}(q_1)$ is in polynomial time. Another corollary, discussed next, generalizes Theorem 2 through the notion of the *Gaifman graph* of a query, a notion that plays an important role in finite model theory (see [Libkin, 2004]); in this graph, the nodes are the variables of the query and the edges consist of pairs of variables occurring in the same ordinary atom.

The corollary applies to the case where every two distinct winner variables belong to different connected components of the Gaifman graph. In this case, we say that the WINNER atoms are *pairwise disconnected*.

Corollary 1. *If q is a Boolean conjunctive query with pairwise-disconnected* WINNER *atoms, then* $\mathsf{Necessity}(q)$ *is decidable in polynomial time.*

Hint of Proof: When the winner variables of a Boolean conjunctive query q are pairwise disconnected, the query "factors out" over the subqueries q' that correspond to the connected components of the Gaifman graph. Consequently, q is necessary on D if and only if each q' is necessary on D. □

Next, we show that, for a natural class of conjunctive queries, the necessity of queries exhibit a PTIME vs. coNP-complete dichotomy.

Definition 2. Let $\mathcal{C}_{2\mathsf{W}}$ be the class of all Boolean conjunctive queries q with the following properties:

(i) There are two distinct WINNER atoms (and both involve the plurality rule and the same fixed election).

(ii) All other atoms of q are ordinary atoms such that no ordinary relation symbol occurs twice (i.e., the ordinary atoms form a *self-join free* query).

Theorem 3. *Let q be a query in the class $\mathcal{C}_{2\mathsf{W}}$.*

- *If the* WINNER *atoms of q are pairwise disconnected, then* $\mathsf{Necessity}(q)$ *is decidable in polynomial time.*
- *Otherwise,* $\mathsf{Necessity}(q)$ *is coNP-complete.*

Hint of Proof: Tractability follows from Corollary 1. For coNP-hardness, we first handle the query q_h, where

$$\begin{aligned} q_h() \;{:}-\; & \textsc{Winner}(\texttt{plurality}, e, c), R(c, d), \\ & \textsc{Winner}(\texttt{plurality}, e, d) \end{aligned}$$

We then reduce q_h to each remaining query via the proof technique of a dichotomy by Kenig et al. [Kenig *et al.*, 2017]. □

As a direct application, $\mathsf{Necessity}(q_3)$ is coNP-complete, where q_3 is the query in Example 3. In contrast, $\mathsf{Necessity}(q_4)$ is solvable in polynomial time, where q_4 is the following query.

$$\begin{aligned} q_4() \;{:}-\; & \textsc{Winner}(\texttt{plurality}, e, c), \\ & \textsc{Winner}(\texttt{plurality}, e, d), \\ & \textsc{Supports}(c, \texttt{pro-choice}), \textsc{Cand}(d, p, \texttt{BS}, a). \end{aligned}$$

4.2 Hardness Beyond the Plurality Rule

Here, we show that for a large class of positional scoring rules, there are conjunctive queries q involving WINNER atoms and ordinary atoms such that $\mathsf{Necessity}(q)$ is coNP-complete.

Definition 3. *Let r be a pure positional scoring rule. We say that r is* eventually constant *if there is a positive integer k such that for every m and for every s with $k < s \leq m$, we have that $r(m, s) = 0$.*

The plurality rule, the k-approval rule, for $k \geq 2$, and the rule r with scoring vectors of the form $(2, 1, 0, \dots, 0)$ (this rule played an important role in the proof of the Classification Theorem 1 [Baumeister and Rothe, 2012; Betzler and Dorn, 2010]) are eventually constant positional scoring rules, while the Borda rule is not.

Theorem 4. *If r is an eventually constant positional scoring rule, then there is a Boolean conjunctive query q involving* WINNER *atoms and ordinary atoms such that* $\mathsf{Necessity}(q)$ *is coNP-complete.*

Hint of Proof: By reduction from the complement of POSITIVE a-IN-b SAT, for suitable values of a and b. This problem asks: given a CNF-formula consisting entirely of positive clauses of length b, is there a truth assignment such that exactly a variables each clause are true? If $b \geq 3$ and $1 \leq a < b$, then POSITIVE a-IN-b SAT is NP-complete by Schaefer's dichotomy [Schaefer, 1978]. □

Definition 4. *Let k be a positive integer. The k-*veto *rule (also known as the $(m - k)$-*approval *rule) is the positional scoring rule with scoring vectors of the form $(1, \dots, 1, 0, \dots, 0)$ with $(m - k)$ scoring values of 1 at the beginning and k scoring values of 0 at the end.*

Clearly, the 1-veto rule is the well known veto rule.

Theorem 5. *For every $k \geq 1$, there is a Boolean conjunctive query q involving* WINNER *atoms of the k-veto rule and ordinary atoms such that* $\mathsf{Necessity}(q)$ *is coNP-complete.*

Hint of Proof: Similar to the proof of Theorem 4. □

5 Concluding Remarks

We presented a framework that enriches social choice with relational database context. This framework supports the formulation of queries about winners in elections, alongside contextual information about candidates, voters, and positions on issues. In the presence of incomplete voter preferences, the semantics of queries are given via the notions of necessary and possible answers, which extend the notions of necessary and possible winners. Our technical results about the necessary answers of conjunctive queries reveal that the context makes a substantial difference, since the complexity of the necessary answers of queries may be higher than the complexity of the necessary winners.

It remains an open problem to classify the complexity of the necessary answers for all positional scoring rules and conjunctive queries. It is also open to determine the complexity of the possible answers for the plurality rule and the veto rule (for all other positional scoring rules, even computing the possible winners is an intractable problem). It is interesting to go beyond conjunctive queries and, among others, consider queries that support aggregate operators, such as count and average.

The alternative modeling of preferences is another direction for future research. Probabilistic votes adopt statistical models of preferences, such as Mallows [Mallows, 1957] and the Repeated Insertion Model (RIM) [Doignon *et al.*, 2004]. The analog of computing necessary/possible winners is to compute the probability that a given candidate wins [Bachrach *et al.*, 2010; Lu and Boutilier, 2011].

In our framework, the analog is *probabilistic query answering*, where the goal is to compute the marginal probability of possible query answers [Dalvi and Suciu, 2004; Suciu *et al.*, 2011]. Conjunctive query evaluation over RIM databases has been studied in [Kenig *et al.*, 2017], but without the angle of computational social choice.

The modeling of voter preferences may also incorporate *constraints* (or *dependencies*) that restrict the possible completions to those satisfying some conditions that are known to hold. An example is that voters vote according to party affiliation—all candidates of one party are preferred to all candidates of another party (but we do not know upfront which party comes first).

References

[Abiteboul *et al.*, 1995] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[Bachrach *et al.*, 2010] Yoram Bachrach, Nadja Betzler, and Piotr Faliszewski. Probabilistic possible winner determination. In *AAAI*. AAAI Press, 2010.

[Baumeister and Rothe, 2012] Dorothea Baumeister and Jörg Rothe. Taking the final step to a full dichotomy of the possible winner problem in pure scoring rules. *Inf. Process. Lett.*, 112(5):186–190, 2012.

[Betzler and Dorn, 2010] Nadja Betzler and Britta Dorn. Towards a dichotomy for the possible winner problem in elections based on scoring rules. *J. Comput. Syst. Sci.*, 76(8):812–836, 2010.

[Dalvi and Suciu, 2004] Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, pages 864–875. Morgan Kaufmann, 2004.

[Doignon *et al.*, 2004] Jean-Paul Doignon, Aleksandar Pekeč, and Michel Regenwetter. The repeated insertion model for rankings: Missing link between two subset choice models. *Psychometrika*, 69(1):33–54, 2004.

[Fagin *et al.*, 2005] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.

[Hägele and Pukelsheim, 2001] Gunter Hägele and Friedrich Pukelsheim. Llull's writings on electoral systems. *Studia Lulliana*, 41(97):3–38, 2001.

[Imielinski and Jr., 1984] Tomasz Imielinski and Witold Lipski Jr. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.

[Jacob *et al.*, 2014] Marie Jacob, Benny Kimelfeld, and Julia Stoyanovich. A system for management and analysis of preference data. *PVLDB*, 7(12):1255–1258, 2014.

[Kenig *et al.*, 2017] Batya Kenig, Benny Kimelfeld, Haoyue Ping, and Julia Stoyanovich. Querying probabilistic preferences in databases. In *PODS*, pages 21–36, 2017.

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

[Konczak, 2006] Kathrin Konczak. Voting theory in answer set programming. In *WLP*, volume 1843-06-02 of *INFSYS Research Report*, pages 45–53. Technische Universität Wien, Austria, 2006.

[Lenzerini, 2002] Maurizio Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, New York, NY, USA, 2002. ACM.

[Libkin, 2004] Leonid Libkin. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

[Lu and Boutilier, 2011] Tyler Lu and Craig Boutilier. Vote elicitation with probabilistic preference models: Empirical estimation and cost tradeoffs. In *ADT*, volume 6992 of *Lecture Notes in Computer Science*, pages 135–149. Springer, 2011.

[Lukasiewicz *et al.*, 2014] Thomas Lukasiewicz, Maria Vanina Martinez, Gerardo I. Simari, and Oana Tifrea-Marciuska. Ontology-based query answering with group preferences. *ACM Trans. Internet Techn.*, 14(4):25:1–25:24, 2014.

[Mallows, 1957] C. L. Mallows. Non-null ranking models. i. *Biometrika*, 44(1-2):114–130, June 1957.

[Schaefer, 1978] Thomas J. Schaefer. The complexity of satisfiability problems. In *STOC*, pages 216–226, New York, NY, USA, 1978. ACM.

[Suciu *et al.*, 2011] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[Xia and Conitzer, 2011] Lirong Xia and Vincent Conitzer. Determining possible and necessary winners given partial orders. *J. Artif. Intell. Res.*, 41:25–67, 2011.