The Expressivity of Constraint Query Languages with Boolean Algebra Linear Cardinality Constraints[⋆]

Peter Z. Revesz

Department of Computer Science and Engineering
University of Nebraska-Lincoln, Lincoln, NE 68588, USA
{revesz}@cse.unl.edu

Abstract. Constraint query languages with Boolean algebra linear cardinality constraints were introduced recently and shown to be evaluable using a quantifier elimination method in [22]. However, the expressive power of constraint query languages with linear cardinality constraints is still poorly understood in comparison with other cases of constraint query languages. This paper makes several contributions to the analysis of their expressive power. Several problems that were previously provably impossible to express even in $FO + POLY$ are shown to be expressible using first-order query languages with linear cardinality constraints $FO + BALC$. We also show that all monadic Datalog queries are expressible in $FO + BALC$. Finally, we also show a new results for $FO + LINEAR$ by expressing in it the problem of finding the time when two linearly moving point objects are closest to each other.

1 Introduction

An important question for *constraint databases* [13] is their expressive power, that is, to know what problems they can or cannot express [17, 20]. Since constraint databases generalize relational databases with the extension of a tuple to *constraint tuples*, which are conjunctions of constraints, it seems intuitive that constraint databases can express more types of problems. However, the fact is that most results regarding the expressive power of constraint query languages are negative. Consider the following problems:

Definition 1. `[MAJORITY]` *The input has two unary relations* R_1 *and* R_2*. The output is true if and only if* $R_1 \subseteq R_2$ *and* $|R_2| \leq 2|R_1|$.

Definition 2. `[TRANSITIVE CLOSURE]` *The input is a binary relation* R*. The output is a binary relation that is the transitive closure of* R*, that is, all pairs* (a_0, a_n) *such that there are elements* (a_0, a_1),…,(a_{n-1}, a_n) *in* R.

[⋆] This work was supported in part by USA National Science Foundation grant EIA-0091530 and a NASA Space and EPSCoR grant.

One very powerful-looking first-order query language is Relational Calculus with polynomial constraints over the real numbers. We call this language $FO + POLY$. The following is a surprising theorem:

Theorem 1 (Benedikt et al. [3]). `MAJORITY` *and* `TRANSITIVE CLOSURE` *are not expressible in* $FO + POLY$.

Recently, Revesz [22] presented a first-order language with Boolean algebras and linear cardinality constraints. We call this language $FO + BALC$. In this paper we show the following:

Theorem 2. `MAJORITY` *and* `TRANSITIVE CLOSURE` *are expressible in* $FO{+}BALC$.

There are some known expressibility results also for first-order queries with linear constraints over the rational numbers, which we denote as $FO{+}LINEAR$. Afrati et al. [1] show that $FO{+}LINEAR$ can express a query that returns *true* if and only if the database consists of exactly c parallel lines where c is a constant. Similarly, they show that $FO{+}LINEAR$ can also express the query that returns *true* if and only if the database consists of two lines intersecting at a point.

In this paper we move beyond just static spatial objects and consider moving point objects. There is a growing interest in representing moving objects. For example, Cai et al. [4, 23], Chomicki et al. [6–8], Güting et al. [11], Kollios et al. [14], Saltenis et al. [24], and Wolfson et al. [26] describe moving object data models and techniques to query moving objects. Constraint databases are a natural representation of moving objects. A natural query on moving objects is the following.

Definition 3. `[TIME CLOSEST]` *Given two moving points that move along two different lines with uniform speed, find the time when they are closest to each other.*

In this paper we show the following:

Theorem 3. `TIME CLOSEST` *is expressible in* $FO + LINEAR$.

While in this paper we focus on first-order query languages, there are also some interesting expressibility results for recursive query languages. For example, Kuijpers and Smits [15] show that if the constraint database input is a binary relation $R(x, y)$ that describes a polynomial spatial relation, i.e., relations expressible using quantifier-free real polynomial constraints, then there is no Datalog query with linear constraints that returns *true* if and only if R is topologically connected.

We only consider in this paper the class of *monadic Datalog* queries, i.e., those queries in which each defined relation (in the head of the rules) is a unary relation. We show the following theorem for monadic Datalog queries:

Theorem 4. *Any monadic Datalog query is expressible in* $FO + BALC$.

The rest of the paper is organized as follows. Section 2 is a brief review of basic concepts. Section 3 proves Theorem 2. This section also shows that several other graph problems, such as `SAME COLOR` and `MAXIMAL CLIQUE` as well as the `N-QUEENS` problem are also expressible in $FO + BALC$. Section 4 proves Theorem 4. Section 5 proves Theorem 3. Section 6 discusses related work. Finally, Section 7 gives some conclusions.

2 Constraint Databases

Constraint databases [13] and constraint logic programming [12] both represent input information as a set of constraint tuples. For example, to describe a graph with vertices $V = \{1, 2, 3, 4\}$ and edges $E = \{(1, 2), (2, 3), (1, 4)\}$, a constraint database over the Boolean algebras of sets of subsets of the integers could be the following, where comma means "and":

Edge

X	Y	
X	Y	$X = \{1\},\ Y = \{2\}$
X	Y	$X = \{2\},\ Y = \{3\}$
X	Y	$X = \{1\},\ Y = \{4\}$

We will use this type of representation for several graph problems in Section 3. The intended meaning of a constraint tuple is that any instantiation of the variables that satisfies the constraint belongs to the relation. In the above example the satisfying instantiations are obvious, but they are less obvious when the constraints are more complex. In particular, we allow besides the equality constraints above any *linear cardinality constraint* [22] of the form:

$$c_1|t_1| + \ldots + c_k|t_k|\ \theta\ b$$

where each t_i for $1 \leq i \leq k$ is a Boolean term –composed of set constants or variables, and the intersection, union, and set complement with respect to the whole set of integers–, each c_i for $1 \leq i \leq k$ and b are integer constants and θ is:
$=$ for the equality relation,
$\geq$ for the greater than or equal comparison operator,
$\leq$ for the less than or equal comparison operator, or
$\equiv_n$ for the congruence relation modulus some positive integer constant n.

Note: Boolean cardinality constraints can express other common constraints over sets. For example, the constraint that t_1 is a subset of t_2, denoted

$$t_1 \subseteq t_2 \ \text{ is equivalent to } \ |t_1 \wedge \overline{t_2}| = 0$$

where t_1 and t_2 are Boolean terms and $\overline{t_2}$ is the complement of t_2. For the sake of greater readability, in the following we will use $\subseteq$ constraints, because readers are more familiar with it.

In this paper we consider first-order languages FO with existential $\exists$ and universal $\forall$ quantifiers, and the connectives logical *and* $\wedge$, *or* $\vee$, and *not* $\neg$, and variables and constants with the usual composition.

We also consider Datalog, which is a rule-based language that is related to Prolog. Each Datalog query contains a Datalog program and an input database. We divide the set of relation names $\mathcal{R}$ into defined relation names and input relation names. Each Datalog query consists of a finite set of rules of the form:

$$R_0(x_1, \ldots, x_k) :\!\!-\, R_1(x_{1,1}, \ldots, x_{1,k_1}), \ldots R_n(x_{n,1}, \ldots, x_{n,k_n}), C_1, \ldots, C_m.$$

where each R_i is either an input relation name or a defined relation name, and the xs are either variables or constants, and each C_i is a constraint. The relation names $R_0, \ldots, R_n$ are not necessarily distinct. For a good introduction of Datalog queries and examples see [17, 20].

3 Problems Expressible in $FO + BALC$

In this section, we study the expressive power of $FO + BALC$. This language seems to be a natural language to express a variety of problems. including `MAJORITY`, several graph problems, and the `N-QUEENS` problem, which is a familiar search problem in AI.

3.1 The `MAJORITY` problem

To express the `MAJORITY` query in $FO + BALC$ we assume that the two input relations $R_1(X)$ and $R_2(Y)$ each contain one equality constraint that sets the value of X and Y equal to a set of numbers. Then the $FO + BALC$ query:

$$\exists X, Y \;\; R_1(X) \;\wedge\; R_2(Y) \;\wedge\; X \subseteq Y \;\wedge\; 2|X| - |Y| \geq 0.$$

correctly expresses `MAJORITY`. The simplicity of the above formula suggests that $FO + BALC$ is a natural language to express this and similar queries.

3.2 The `TRANSITIVE CLOSURE` problem

To express transitive closure, we at first introduce the following definition.

Definition 4. *Let S and X be any two set variables. Then,*

$$S[X] \;\;=_{def}\;\; |X| = 1 \;\wedge\; X \subseteq S.$$

The above definition says that $S[X]$ is true if and only if X is a singleton set, which is a subset of S. Using this definition, it becomes easier to express transitive closure. We express it as follows:

$$\phi_{TC}(Z_1, Z_2) = \forall S \; (S[Z_1] \wedge \forall X, Y \; S[X] \wedge R(X, Y) \rightarrow S[Y]) \rightarrow S[Z_2].$$

The $\phi_{TC}(Z_1, Z_2)$ is a formula with two free variables, namely Z_1 and Z_2. Let us consider any substitution for these two variables. Suppose that Z_1 is substituted by a_0 and Z_2 is substituted by a_n, such that, there are elements (a_0, a_1),...,(a_{n-1}, a_n) in R.

The substituted formula $\phi_{TC}(a_0, a_n)$ says that for all S if ($a_0 \in S$, and if every time the first argument of R is in S, then the second argument of R is also in S), then $a_n \in S$. Since we assumed that there is a sequence of elements (a_0, a_1),...,(a_{n-1}, a_n) in R, the condition of the main implication within ϕ is true if and only if $a_0, \ldots, a_n \in S$. Then clearly $a_n \in S$, hence the then clause is also true. Therefore, the main implication of ϕ is true, and (a_0, a_n) is a substitution into ϕ_{TC} that makes it true.

Conversely, if there is no sequence of elements of the form (a_0, a_1), ..., (a_{n-1}, a_n) in R, then there must exist an S which has in it a_0 and only those which are "reachable" by a sequence of elements from a_0. Then the condition of the main implication in ϕ_{TC} is true, but since a_n is not "reachable" from a_0 the then clause is false. That makes the main implication false, showing that (a_0, a_n) is not a satisfying substitution in this case. This shows that Theorem 2 holds.

3.3 The `SAME COLOR` problem

Given an undirected graph that is 2-colorable, we would like to know which pairs of vertices can be colored the same color. Suppose that the colors we consider are *blue* and *red*. The following expresses that Z_1 and Z_2 can be both colored blue.

$$\phi_B \;=\; \exists B\; B[Z_1] \;\wedge\; B[Z_2]$$

where each vertex which is in set B is assumed to be colored blue, and each vertex not in B is assumed to be colored red. The following formula expresses that the vertices that are connected to a blue vertex are red.

$$\phi_{B-Neighbor} \;=\; \forall X, Y\; B[X] \;\wedge\; Edge(X, Y) \;\rightarrow\; \neg B[Y].$$

Similarly, the following asserts that the vertices connected to a red vertex are blue.

$$\phi_{R-Neighbor} \;=\; \forall X, Y\; \neg B[X] \;\wedge\; Edge(X, Y) \;\rightarrow\; B[Y].$$

Then the formula:

$$\phi_{SC} \;=\; \phi_B \;\wedge\; \phi_{B-Neighbor} \;\wedge\; \phi_{R-Neighbor}$$

expresses the `SAME COLOR` problem.

3.4 The `MAXIMAL CLIQUE` problem

In an undirected graph, a clique is a subgraph in which every vertex is connected with every other vertex. The size of a clique is the number of vertices it contains. We represent an undirected graph by a binary relation *Edge* where *Edge* is symmetric, that is, it represents an undirected edge between X and Y by containing both (X, Y) and (Y, X) as two elements.

Given an undirected graph and an integer constant k, the `MAXIMAL CLIQUE` problem asks whether the size of the maximum clique in the graph is k. To express `MAXIMAL CLIQUE`, at first we express that a graph has a clique with k vertices as follows:

$$\phi_k \;=\; \exists S \; |S| = k \wedge \; \forall X, Y \; (S[X] \wedge S[Y] \wedge X \neq Y) \;\rightarrow\; Edge(X, Y).$$

In the above, S contains the vertices that belong to a clique. Clearly, if the maximal clique has size k, then the graph has a clique of size k but does not have a clique of size $k + 1$. That is,

$$\phi_{MC} = \phi_k \;\wedge\; \neg\phi_{k+1}.$$

3.5 The `N-QUEENS` problem

Given a chess-board of size $n \times n$ for some integer n, the `N-QUEENS` problem asks to place n queens on the chess-board so that no two queens are in the same row, column, or diagonal. The `N-QUEENS` problem is a quite challenging search problem that is not easy to implement in a procedural language like C++. We give a high-level declarative solution to this problem.

While the following solution can be generalized to any n, let us assume for the sake of simplicity that $n = 5$. Then let's count the squares on the chess-board in the usual way, that is, from left to right in each row, and from the top row to the bottom row, as shown below.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Then let variables R_i for $1 \leq i \leq 5$ be those locations in the ith row that contain a queen. We know the following:

$$\phi_R \;=\; R_1 \subseteq \{1, 2, 3, 4, 5\} \;\wedge \ldots \wedge\; R_5 \subseteq \{21, 22, 23, 24, 25\}.$$

Similarly, let variables C_i for $1 \leq i \leq 5$ be those locations in the ith column that contain a queen. We know that:

$$\phi_C \;=\; C_1 \subseteq \{1, 6, 11, 16, 21\} \;\wedge \ldots \wedge\; C_5 \subseteq \{5, 10, 15, 20, 25\}.$$

Further, let variables D_i for $1 \leq i \leq 7$ be the set of locations of queens on the diagonals that run downwards from left to right and have at least two squares. We know that:

$$\phi_D \;=\; D_1 \subseteq \{4, 10\} \;\wedge \ldots \wedge\; D_7 \subseteq \{16, 22\}.$$

The symmetric case is the variables L_i for $1 \leq i \leq 7$ that contain the set of locations of queens on the diagonals that run downwards from right to left and have at least two squares. For those we have:

$$\phi_L \;=\; L_1 \subseteq \{2, 6\} \;\wedge \ldots \wedge\; L_7 \subseteq \{20, 24\}.$$

Since no row or column can have more than one queen, there must be exactly one queen in each row and in each column. Further, on each diagonal there may be at most one queen. Therefore, the formula:

$$\phi_R \wedge \phi_C \wedge \phi_D \wedge \phi_L \wedge (\bigwedge_i |R_i| = 1) \wedge (\bigwedge_i |C_i| = 1) \wedge (\bigwedge_i |D_i| \leq 1) \wedge (\bigwedge_i |L_i| \leq 1)$$

correctly expresses the `N-QUEENS` problem.

4 Monadic Datalog is Expressible in $FO + BALC$

The problems expressed in Section 3 give interesting examples that can be expressed in $FO + BALC$. In this section, instead of giving just examples, we show that an entire class of Datalog programs, namely the class of *monadic Datalog* programs, is expressible in $FO + BALC$.

In a monadic Datalog program P the defined relations are monadic, that is, have arity one. Without loss of generality we assume that the variables range over the integers and the defined (or intensional) relations are $S_1, \ldots, S_m$ and the input (or extensional) relations are $R_1, \ldots, R_n$.

For each rule r_j of P with the form:

$$A_0 :\!\!-\; A_1, \ldots, A_k.$$

where each A_i for $1 \leq i \leq k$ is an atom (i.e., a relation name with variables from the set $X_1, \ldots, X_l$), we write the following expression:

$$\phi_j = \forall X_1, \ldots, X_l \;\; B_1 \wedge \ldots \wedge B_k \rightarrow B_0.$$

where B_i is A_i if A_i contains an input relation name, and B_i is $S_j[X]$ if A_i is $S_j(X)$ for some defined relation name S_j and X is either one of the variables $X_1, \ldots, X_l$ or a concrete set of integer constants. Note that ϕ_j may contain only $S_1, \ldots, S_m$ as free variables ranging over the subsets of the integers. Clearly, the essential difference between the monadic Datalog program and the conjunction:

$$\bigwedge_j \phi_j$$

is that for the monadic Datalog program the least model is returned while the conjunction can have many models. The intersection of all the models of the conjunction is the least model of the monadic Datalog program. To select the least model, we have to add an assertion that the model returned must be the minimal model. We can do that by writing the following expression:

$$\phi_{MoD} = \left(\bigwedge_j \phi_j\right) \ \wedge \ \left(\forall S_1^+ \ldots, S_k^+ \ \ \left(\bigwedge_j \phi_j^+\right) \ \rightarrow \ \left(\bigwedge_i \ S_i \subseteq S_i^+\right)\right)$$

where ϕ_j^+ is like ϕ_j with S_i replaced by S_i^+. Then ϕ_{MoD} expresses what we need. Each monadic Datalog program has a least model $S_1, \ldots, S_k$, which is the output database, i.e., the assignment to $S_1, \ldots, S_k$ returned by an evaluation of the program on an input database. Clearly, ϕ_{MoD} enforces that $S_1, \ldots, S_k$ is a minimal model by constraining all other models $S_1^+ \ldots, S_k^+$ to be bigger or equal to it. Let us see a concrete example.

Example 1. Consider the following monadic Datalog program P.

$$\begin{aligned} &S(X) :\!\!-\!\!- \mathit{Start}(X). \\ &S(Y) :\!\!-\!\!- S(X),\ \mathit{Edge}(X, Y). \end{aligned}$$

Here program P finds the vertices that are reachable from the start vertices contained in the input relation *Start*. The first rule can be expressed by:

$$\phi_1 \quad = \quad \forall X \quad \mathit{Start}(X) \rightarrow S[X].$$

The second rule can be expressed by:

$$\phi_2 \quad = \quad \forall X, Y \quad S[X] \wedge \mathit{Edge}(X, Y) \rightarrow S[Y].$$

Then P can be expressed in $FO + BALC$ as:

$$\phi_P \quad = \quad (\phi_1 \wedge \phi_2) \ \wedge \ \left(\forall S^+ \ (\phi_1^+ \wedge \phi_2^+) \ \rightarrow \ S \subseteq S^+\right).$$

Since S is a model of the rules of P, and for any other model S^+ of the rules of P we have $S \subseteq S^+$, it follows that S must be the least model.

5 The `TIME CLOSEST` problem in $FO + LINEAR$

Suppose that two cars, which both move linearly in the plane, want to radio-communicate with each other. What is the best time to attempt the radio communication? Intuitively, the best time would be when the two cars are closest to each other, hence that time instance needs to be found. We show that it can be found using only linear constraints, which is surprising, because at first glance the problem seems to require the Euclidean distance function, which is a quadratic polynomial constraint.

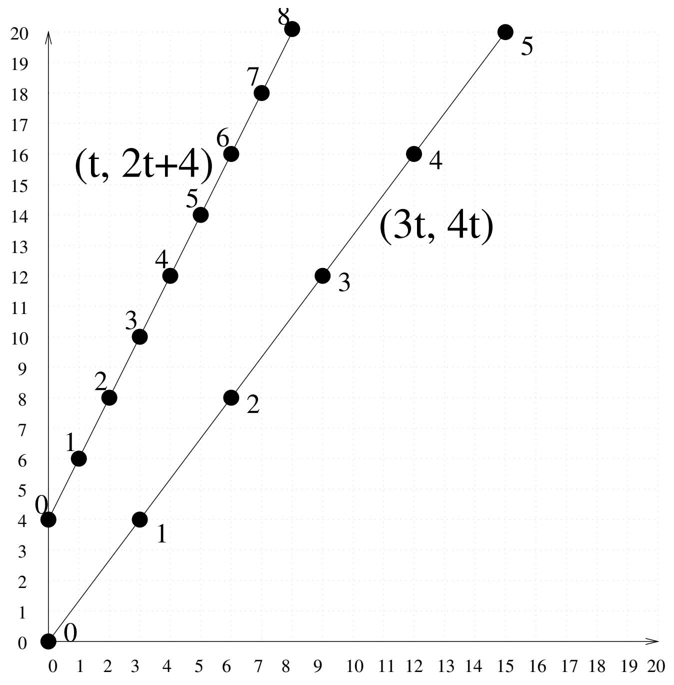

Fig. 1. Two cars moving in the plane.

The two cars can be represented by two constraint database relations $P_1(x, y, t)$ and $P_2(x, y, t)$. For example, an input database instance could be the following (see also Figure 1):

$$P_1(x, y, t) :\!\!- \; x = t, \;\; y = 2t + 4.$$
$$P_2(x, y, t) :\!\!- \; x = 3t, \;\; y = 4t.$$

Suppose that we would like to find the time instance t when the two cars are closest to each other. We can define first the difference between the two cars at any time t as follows:

$$\Delta P(x, y, t) = \exists x_1, x_2, y_1, y_2 \; P_1(x_1, y_1, t) \wedge P_2(x_2, y_2, t) \wedge x = x_2 - x_1 \wedge y = y_2 - y_1.$$

ΔP is also a moving point in the plane as shown in Figure 2. The difference between the two cars is exactly the difference between ΔP and the origin at any time t. Therefore, the two cars are closest to each other when ΔP is closest to the origin. Now the projection of ΔP onto the plane is a line, which is the path along which ΔP travels. We can find this by:

$$\Delta Pline(x, y) = \exists t \; \Delta P(x, y, t).$$

Let us now take the line which goes through the origin and is perpendicular to $\Delta Pline$. If (x_1, y_1) and (x_2, y_2) are two points on $\Delta Pline$, then the slope of

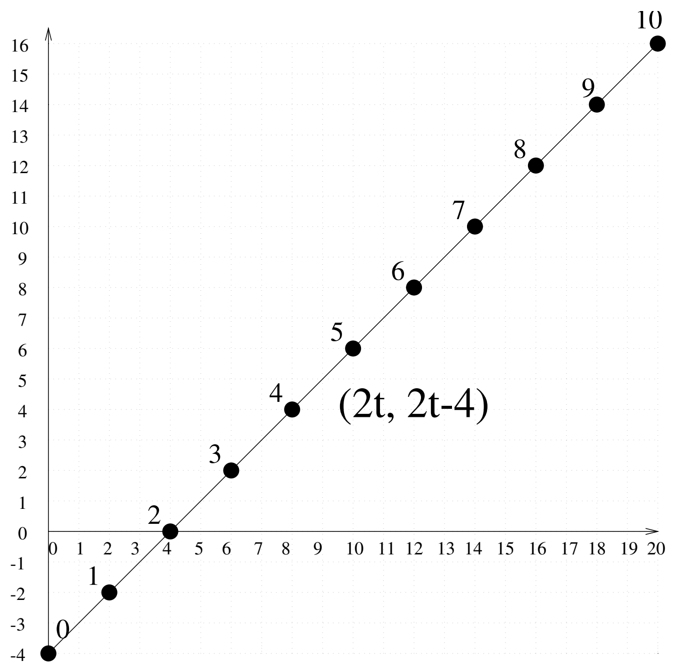

Fig. 2. The ΔP moving point.

$\Delta Pline$ is:

$$\frac{y_2 - y_1}{x_2 - x_1}.$$

The perpendicular line will have a negative reciprocal slope and will go through the origin. Hence its line equation is:

$$y = -\frac{x_2 - x_1}{y_2 - y_1} x \tag{1}$$

Now we can chose any two distinct points on the line $\Delta Pline$ for expressing the line equation. Let us choose (x_1, y_1) to be the intersection point of $\Delta Pline$ and the line perpendicular to it and going through the origin. Further, let us chose the second point (x_2, y_2) such that

$$y_2 = x_1 + y_1. \tag{2}$$

Clearly, this is always possible to do when the line is not vertical. Now what is the intersection point? It will satisfy Equations (1) and (2), that is:

$$\begin{aligned} y_1 &= -\frac{x_2 - x_1}{y_2 - y_1} x_1 \\ x_1 &= y_2 - y_1. \end{aligned} \tag{3}$$

The above can be simplified to:

$$\begin{aligned} y_1 &= x_1 - x_2 \\ x_1 &= y_2 - y_1. \end{aligned} \tag{4}$$

Therefore, if $\mathit{\Delta Pline}$ is not vertical, that is, $x_1 \neq x_2$, then the point of $\mathit{\Delta Pline}$ that is closest to the origin is exactly the intersection point, hence:

$$\begin{aligned} \mathit{Closest\_Point}(x_1, y_1) = {} & \exists x_2, y_2\ \mathit{\Delta Pline}(x_1, y_1)\ \wedge\ \mathit{\Delta Pline}(x_2, y_2)\ \wedge \\ & y_1 = x_1 - x_2\ \wedge\ x_1 = y_2 - y_1\ \wedge\ x_1 \neq x_2. \end{aligned}$$

Otherwise, if $\mathit{\Delta Pline}$ is vertical, that is, for any two different points $x_1 = x_2$, then the closest point is:

$$\begin{aligned} \mathit{Closest\_Point}(x_1, y_1) = {} & \exists x_2, y_2\ \mathit{\Delta Pline}(x_1, y_1)\ \wedge\ \mathit{\Delta Pline}(x_2, y_2)\ \wedge \\ & y_1 = 0\ \wedge\ x_1 = x_2\ \wedge\ y_2 \neq 0. \end{aligned}$$

Hence putting the above two cases together, we have:

$$\begin{aligned} \mathit{Closest\_Point}(x_1, y_1) = {} & \exists x_2, y_2\ \mathit{\Delta Pline}(x_1, y_1)\ \wedge\ \mathit{\Delta Pline}(x_2, y_2)\ \wedge \\ & ((y_1 = x_1 - x_2\ \wedge\ x_1 = y_2 - y_1\ \wedge\ x_1 \neq x_2)\ \vee \\ & (y_1 = 0\ \wedge\ x_1 = x_2\ \wedge\ y_2 \neq 0)). \end{aligned}$$

The time when the two cars are closest to each other is:

$$\mathit{Closest\_Time}(t) = \exists x, y\ \ \mathit{\Delta P}(x, y, t)\ \wedge\ \mathit{Closest\_Point}(x, y).$$

Clearly, the above formula is in $FO + LINEAR$, which shows Theorem 3.

Example 2. Let us look at what will happen when we have the input database instance P_1 and P_2 as given above. In that case, we obtain:

$$\mathit{\Delta P}(x, y, t)\quad :\!\!-\ x = 2t,\ y = 2t - 4.$$

$$\mathit{\Delta Pline}(x, y)\ :\!\!-\ x = y + 4.$$

For $\mathit{Closest\_Point}$, we get after simplifications:

$$\begin{aligned} \mathit{Closest\_Point}(x_1, y_1)\ = {} & \ \exists x_2, y_2\ \ x_1 = y_1 + 4\ \wedge x_2 = y_2 + 4\ \wedge \\ & y_1 = x_1 - x_2\ \wedge\ x_1 = y_2 - y_1\ \wedge\ x_1 \neq x_2. \end{aligned}$$

Eliminating x_2 and y_2 we get:

$$\mathit{Closest\_Point}(x_1, y_1)\ =\ x_1 = 2\ \wedge\ y_1 = -2.$$

Finally, the closest time is calculated as:

$$\mathit{Closest\_Time}(t)\ =\ \exists x, y\ \ x = 2t\ \wedge\ y = 2t - 4\ \wedge\ x = 2\ \wedge\ y = -2.$$

Eliminating x and y we get:

$$\mathit{Closest\_Time}(1).$$

Therefore, the two cars are closest at time 1. It is at that time that the two cars should attempt to radio-communicate with each other.

6 Related Work

The present work extends the author's earlier work that presented a quantifier elimination for the first-order theory of atomic Boolean algebras of sets with linear cardinality constraints [21, 22] but did not examine its expressive power.

Feferman and Vaught (see Theorem 8.1 in [10]) proved the decidability of the first-order theory of atomic Boolean algebras of sets with *set-theoretical equivalence* which are also commonly called today *equicardinality* constraints. Let us denote this logic by $FO + EC$. An equicardinality constraint between sets A and B, denoted $A \sim B$, simply means that sets A and B have the same cardinality and can be easily expressed by the linear cardinality constraint $|A| - |B| = 0$. Hence obviously $FO + BALC$ includes $FO + EC$. Interestingly, however, the two logics have the same expressive power, because $FO + EC$ can express any linear cardinality constraint. For example, the $FO + BALC$ formula:

$$\exists X \quad 2|X| - |Y| = 0$$

can be expressed by the $FO + EC$ formula:

$$\exists X, Z \quad X \cap Z \sim \emptyset \quad \wedge \quad X \sim Z \quad \wedge \quad X \cup Z \sim Y$$

where $\emptyset$ is the symbol for the empty set. The formula says that there exist sets X and Z that do not intersect, have an equal cardinality, and whose union has an equal cardinality with Y.

Although equal in expressive power, $FO + EC$ has some limitations, because while we can eliminate the variable from the first formula and obtain:

$$|Y| \equiv_2 0$$

we cannot eliminate the variables from the second formula and get a quantifier-free formula with only equicardinality constraints. This shows that:

Theorem 5. *$FO + EC$ does not admit quantifier elimination.*

Now let's try to consider a multi-sorted logic, that is, one where each of the variables and quantifiers ranges either over the integers (this is not allowed in $FO + BALC$) or the subsets of the integers. This kind of multi-sorted logic was first considered by Zarba [27], who gave a quantifier elimination method for the fragment that contains only quantifiers ranging over the integers and conjectured the whole logic to be undecidable. Kuncak et al. [16] showed this logic, which they called Boolean algebra with Presburger arithmetic and can be denoted by $FO + BAPA$, to be decidable and admitting quantifier elimination.

Obviously $FO + BAPA$ includes $FO + BALC$. However, in this case too, it can be shown that $FO + BAPA$ and $FO + BALC$ have the same expressive power. The proof reduces any $FO + BAPA$ formula to a logically equivalent $FO + BALC$ formula as follows.

An integer variable can occur in a $FO + BAPA$ formula only within the Presburger arithmetic constraints of addition of the form $x + y = z$, comparison of the form $x \geq y$, and congruence of the form $x \equiv_n b$, where x, y and z are integer variables and b is an integer constant. For every integer variable x introduce a new set variable X. Then translate every addition constraint of the above form into:

$$|X| + |Y| - |Z| = 0 \tag{5}$$

every comparison constraint of the above form into:

$$|X| - |Y| \geq 0 \tag{6}$$

and every congruence constraint of the above form into:

$$|X| \equiv_n b. \tag{7}$$

Clearly, the Presburger addition (comparison and congruence) constraint is true for some assignment of integer constants c_1, c_2, and c_3 to x, y, and z if and only if Equation (5) (respectively, Equation (6) and Equation (7)) is true for any arbitrary assignment of set constants C_1, C_2, and C_3 for X, Y, and Z with the only restriction that $|C_1| = c_1$, $|C_2| = c_2$, and $|C_3| = c_3$.

The above gives a reduction of $FO + BAPA$ formulas to $FO + BALC$ formulas. Further, it is obvious that from any solution of the $FO + BALC$ formula, it is easy to generate a solution of the $FO + BAPA$ formula by simply taking the cardinalities of those set variables that were introduced in the reduction from $FO + BAPA$ to $FO + BALC$. (Remember that the set variables introduced in the conversion into $FO + BALC$ are simply integer variables in $FO + BAPA$.) Hence we have:

Theorem 6. *$FO + BALC$ and $FO + BAPA$ and $FO + EC$ have the same expressive power.*

Besides expressive power another important consideration for the above related logics is their computational complexities which turns out to be reducible to cases of Presburger arithmetic. Recall that any formula can be easily put into a prenex normal form where all the quantifiers precede the rest of the formula.

When read left to right, the quantifiers at the beginning of the prenex formula show a certain pattern of alternations between sequences of existential and sequences of universal quantifiers. The number of alternations turns out to be complexity-wise important as shown by the following theorem.

Theorem 7 (Reddy and Loveland [18]). *The validity of a Presburger arithmetic sentence with n quantifiers, length $O(n)$, and m quantifier alternations can be decided in $2^{n^{O(m)}}$ space.*

Revesz [21, 22] noted the following.

Theorem 8 (Revesz [21, 22]). *Quantifier elimination of any $FO + BALC$ formula with n quantifiers and length $O(n)$ and m quantifier alternations reduces to a quantifier elimination of a Presburger arithmetic formula with 2^n quantifiers and length $2^{O(n)}$ and m or $m + 1$ quantifier alternations.*

The reason for the above is that the quantifier-elimination in [21, 22] is based on a reduction of a $FO + BALC$ formula into a Presburger arithmetic formula by introducing into the prenex part of the formula a single sequence of 2^n existentially quantified integer variables. Kuncak et al. [16] use a similar reduction together with Theorem 7, which allows them to show that:

Theorem 9 (Kuncak et al [16]). *The validity of a $FO + BAPA$ sentence with n quantifiers, length $O(n)$ and m quantifier alternations can be decided in $2^{n^{O(mn)}}$ space.*

Similarly, Theorems 7 and 8 can be combined to show that:

Theorem 10. *The validity of a $FO+BALC$ sentence with n quantifiers, length $O(n)$ and m quantifier alternations can be decided in $2^{n^{O(mn)}}$ space.*

In summary, $FO+BALC$, $FO+BAPA$ and $FO+EC$ are closely related logics that have the same expressive power and computational complexity. Therefore, the choice among these three logics is only a stylistic preference as far as decision problems are concerned. However, when considering constraint query languages, where the constraint query evaluation requires quantifier elimination, then only $FO + BALC$ and $FO + BAPA$ can be considered.

Although all three logics assume the domain of variables to be atomic Boolean algebras (which are isomorphic to Boolean algebras of subsets of the integers), some mention must be made of the case when the domain is an *atomless* Boolean algebra. For example, consider the atomless Boolean algebra where the variables denote areas in the real plane, and the operators are interpreted as intersection and union of areas, and area complement with respect to the real plane. In this logic it is possible to introduce polynomial constraints such as $|A|^2 > |B \cap C|^3$, which expresses that the square of the area A is greater than the cube of the area that is the intersection of B and C. Note that here $|A|$ is the measure of the area of an element of the atomless Boolean algebra and not the cardinality of an element of an atomic Boolean algebra. Formulas with polynomial constraints over areas, abbreviated $FO+POLYA$, can be reduced to $FO+POLY$ similarly to the reduction of $FO+BALC$ to Presburger arithmetic. In particular, a $FO+POLYA$ formula with n area variables will be reduced to a $FO+POLY$ formula with 2^n real number variables. Each real number variable will represent one of the 2^n areas that are obtained by considering the intersections of the n area variables or their complements.

For example, if we have only the three area variables A, B, and C, then we need to consider eight areas $A \cap B \cap C, \ldots, \overline{A} \cap \overline{B} \cap \overline{C}$. Note that any polynomial constraint over the measures of Boolean terms over A, B, and C is expressible as a polynomial constraint over the measures of the eight areas. For instance, $|A|^2 > |B \cap C|^3$ can be expressed by:

$$(|A \cap B \cap C| + |A \cap B \cap \overline{C}| + |A \cap \overline{B} \cap C| + |A \cap \overline{B} \cap \overline{C}|)^2 > (|A \cap B \cap C| + |\overline{A} \cap B \cap C|)^3.$$

The measures of the eight areas are independent of each other. Hence the validity of a $FO+POLYA$ formula can be tested by considering a $FO+POLY$

formula where each of the eight measures are replaced by a unique real number variable and the three Boolean variables A, B, and C are replaced by real number variables that are constrained to be equal to a linear combination of the eight independent variables. Therefore, we have that:

Theorem 11. *The validity of a $FO + POLYA$ sentence with n quantifiers, length $O(n)$, and m quantifier alternations can be reduced to deciding the validity of a $FO{+}POLY$ sentence of length $2^{O(n)}$ with m or $m{+}1$ quantifier alternations.*

Since $FO + POLY$ formulas admit quantifier elimination, we also can show:

Theorem 12. *$FO + POLYA$ admits quantifier elimination.*

Tarski [25] gave the first decision procedure and quantifier elimination algorithm for the real closed fields, but more efficient algorithms include [2, 5, 9, 19].

7 Conclusions

We gave several positive results about the expressivity of constraint queries. There are many interesting open questions regarding expressivity. For example, can we express some more graph problems using constraint queries, such as, the chromatic number of a graph? Some earlier negative results gave the impression that constraint queries are not too useful. However, it is still currently being discovered what are the best application areas for various constraint queries. For the problems that we considered, the solutions found are simpler than the procedural language solutions. This suggests that by providing high-level declarative query languages, constraint database systems could be beneficial for users in practice on problems related to the ones presented in this paper.

References

1. F. Afrati, T. Andronikos, and T. Kavalieros. On the expressiveness of query languages with linear constraints: Capturing desirable spatial properties. In *Proc. Workshop on Constraint Databases and Their Applications*, volume 1191 of *Lecture Notes in Computer Science*, pages 105–115. Springer-Verlag, 1997.
2. S. Basu. New results on quantifier elimination over real closed fields and applications to constraint databases. *Journal of the ACM*, 46(4):537–55, 1999.
3. M. Benedikt, G. Dong, L. Libkin, and L. Wong. Relational expressive power of constraint query languages. *Journal of the ACM*, 45(1):1–34, 1998.
4. M. Cai, D. Keshwani, and P. Z. Revesz. Parametric rectangles: A model for querying and animating spatiotemporal databases. In *Proc. 7th International Conference on Extending Database Technology*, volume 1777 of *Lecture Notes in Computer Science*, pages 430–44. Springer-Verlag, 2000.
5. B. F. Caviness and J. R. Johnson, editors. *Quantifier Elimination and Cylindrical Algebraic Decomposition.* Springer-Verlag, 1998.
6. J. Chomicki, S. Haesevoets, B. Kuijpers, and P. Z. Revesz. Classes of spatiotemporal objects and their closure properties. *Annals of Mathematics and Artificial Intelligence*, 39(4):431–461, 2003.

7. J. Chomicki and P. Z. Revesz. Constraint-based interoperability of spatiotemporal databases. *Geoinformatica*, 3(3):211–43, 1999.
8. J. Chomicki and P. Z. Revesz. A geometric framework for specifying spatiotemporal objects. In *Proc. International Workshop on Time Representation and Reasoning*, pages 41–6, 1999.
9. G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In H. Brakhage, editor, *Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pages 134–83. Springer, 1975.
10. S. Feferman and R. L. Vaught. The first-order properties of products of algebraic systems. *Fundamenta Mathematicae*, 47:57–103, 1959.
11. R.H. Güting, M.H. Böhlen, M. Erwig, C.C. Jenssen, N.A. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and querying moving objects. *ACM Transactions on Database Systems*, 25, 2000.
12. J. Jaffar and J. L. Lassez. Constraint logic programming. In *Proc. 14th ACM Symposium on Principles of Programming Languages*, pages 111–9, 1987.
13. P. C. Kanellakis, G. M. Kuper, and P. Z. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51(1):26–52, 1995.
14. G. Kollios, D. Gunopulos, and V. J. Tsotras. On indexing mobile objects. In *Proc. ACM Symposium on Principles of Database Systems*, pages 261–72, 1999.
15. B. Kuijpers and M. Smits. On expressing topological connectivity in spatial Datalog. In *Proc. Workshop on Constraint Databases and Their Applications*, volume 1191 of *Lecture Notes in Computer Science*, pages 116–33. Springer-Verlag, 1997.
16. V. Kuncak, H. H. Nguyen, and M. Rinard. An algorithm for deciding BAPA: Boolean algebra with Presburger arithmetic. In *Proc. 20th International Conference on Automated Deduction*, Lecture Notes in Computer Science. Springer-Verlag, 2005.
17. G. M. Kuper, L. Libkin, and J. Paredaens, editors. *Constraint Databases.* Springer-Verlag, 2000.
18. C.R. Reddy and D.W. Loveland. Presburger arithmetic with bounded quantifier alternation. In *Proc. ACM Symp. on Theory of Comp.*, pages 320–325, 1978.
19. J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. *Journal of Symbolic Computation*, 13(3):255–352, 1992.
20. P. Z. Revesz. *Introduction to Constraint Databases.* Springer-Verlag, New York, 2002.
21. P. Z. Revesz. Cardinality constraint databases. In *Manuscript submitted to 23rd ACM Symposium on Principles of Database Systems*, November 2003.
22. P. Z. Revesz. Quantifier-elimination for the first-order theory of Boolean algebras with linear cardinality constraints. In *Proc. 8th East European Conference on Advances in Databases and Information Systems*, volume 3255 of *Lecture Notes in Computer Science*, pages 1–21. Springer-Verlag, 2004.
23. P. Z. Revesz and M. Cai. Efficient querying of periodic spatio-temporal databases. *Annals of Mathematics and Artificial Intelligence*, 36(4):437–457, 2002.
24. S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez. Indexing the positions of continuously moving objects. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 331–42, 2000.
25. A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* University of California Press, Berkeley, 1951.
26. O. Wolfson, A. Sistla, B. Xu, J. Zhou, and S. Chamberlain. DOMINO: Databases for moving objects tracking. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 547–9, 1999.

27. C. G. Zarba. A quantifier elimination algorithm for a fragment of set theory involving the cardinality operator. In *18th Int. Workshop on Unification*, 2004.