Variable Independence in Constraint Databases

Jan Chomicki, Dina Goldin, Gabriel Kuper, and David Toman

Abstract—In this paper, we study constraint databases with variable independence conditions (vics). Such databases occur naturally in the context of temporal and spatiotemporal database applications. Using computational geometry techniques, we show that variable independence is decidable for linear constraint databases. We also present a set of rules for inferring vics in relational algebra expressions. Using vics, we define a subset of relational algebra that is closed under restricted aggregation.

Index Terms—Constraint databases, aggregation, closure, integrity constraints, spatiotemporal databases.

1 Introduction

Constraint databases [19] generalize in a natural way the relational model of data by allowing infinite relations that are finitely representable using constraints. Constraint databases find numerous applications in spatial [4], [5], [6], [27], [36], temporal [7], [32], and spatiotemporal databases [14], [16]. A variety of constraint languages have been proposed, including dense order constraints, linear arithmetic constraints, and the most general—polynomial constraints [13], [1].

The fundamental relational query languages, relational algebra and calculus, and Datalog do not require that the relations be finite and are thus applicable to constraint databases. Their evaluation mechanisms, however, are different than in relational databases because, instead of finite relations, they deal with *finite representations* of infinite relations. Recent developments in constraint databases, in particular the research on aggregation and spatiotemporal applications, suggest a need for *middle-ground* formalisms that preserve some of the expressive power of constraint databases and constraint query languages, while at the same time generalizing in a natural way the basic assumptions underlying the classical relational model of data. In this paper, we propose an approach that fits in this category. We study constraint databases with *variable independence conditions (vics)*. It turns out that such conditions provide solutions to a number of practically and theoretically motivated problems in constraint databases, including the following:

- ensuring *closure* under aggregation [21],
- enhancing the expressive power of practical spatiotemporal query languages through *data interoperability* [11],
- simplifying spatiotemporal *query evaluation* [14], [16], and
- enabling compilation to SQL for temporal query languages with multiple temporal dimensions [32].

The first problem, which provided the original motivation for studying vics, is introduced below; the remaining problems are described later in the paper. As shown in [21], relational algebra over linear constraint databases is not closed under aggregation using **area**; that is, the output cannot always be represented with linear constraints. The typical example is where we have a region whose boundaries vary linearly with time, and we want to know how the area of the region varies with time.

Example 1.1. Consider a (linear constraint) relation

$$R = \{(x, y, z) : x \geq y \wedge y \geq 0 \wedge x \leq z\}.$$

For any given z, the corresponding plane fragment is shown in Fig. 1. Applying **area** to the first and second attributes of R results in the relation

$$S = \{(a, z) : a = z^2/2\}$$

which is not representable using linear arithmetic constraints.

While there are applications where one might conceivably need to be able to use the **area** operator in this fashion (and for which the *closure* problem would be unavoidable), this is not the case for many applications. Consider a geographical database with *cadastral* information, i.e., information on land ownership and land boundaries. Land ownership does not vary continuously—pieces of land are acquired by individuals at single, discrete points of time, allowing a grouping of the relation's attributes into "independent" subsets. As a result, we avoid the closure problem when computing the evolution of area ownership.

Example 1.2. Cadastral (real estate) records can be represented using a constraint relation with tuples of the form

$$n = N \wedge t_i \leq t \leq t'_i \wedge C_i(x, y),$$

- J. Chomicki is with the Department of Computer Science and Engineering, University at Buffalo, 201 Bell Hall, Box 602000, Buffalo, NY 14260-2000. E-mail: chomicki@cse.buffalo.edu.
- D.Q. Goldin is with the Department of Computer Science and Engineering, University of Connecticut, 191 Auditorium Road, U-155 Storrs, CT 06269-3155. E-mail: dqg@cse.uconn.edu.
- G. Kuper is with the Dipartimento Informatica e Telecomunicazioni, Facolta di Scienze, Universita di Trento, 38050 Povo, Trento, Italy. E-mail: kuper@acm.org.
- D. Toman is with the School of Computer Science, University of Waterloo, 200 University Avenue West, Waterloo, Ontario, Canada N2L 3G1. E-mail: david@uwaterloo.ca.

Manuscript received 15 Sept. 2000; revised 17 Dec. 2001; accepted 12 Aug. 2002.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 112869.

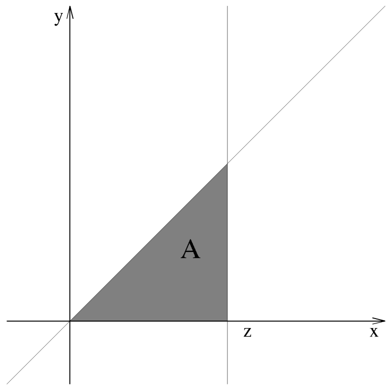


Fig. 1. $area(A) = z^2/2$.

where $C_i(x, y)$ are constraints that describe the region owned by N between (constant) times t_i and t'_i. If we want to find the area A of the land owned by N as a function of t, we can represent the result as a set of constraint tuples of the form

$$t_i < t < t'_i \wedge (n = N) \wedge (z = A)$$

which is clearly finitely representable.

The important property of the tuples in the above example is that the constraints on x and y—those on which the area computation is performed—are separate from the constraints on n and t. Many typical uses of area computation in GIS systems [37], [38] have this property, which we will define as *variable independence* of space and time. This is to be contrasted with the case where the boundaries of a region vary continuously with time and the area of the region may fail to be representable using the same class of constraints, as in Example 1.1.

To deal with vics, a number of fundamental issues need to be addressed. In particular, vics are in some sense analogous to integrity constraints, so the issues of their testing and inference arise naturally.

First, an appropriate *definition* of a vic has to be provided. After reviewing the basic notions of constraint databases and constraint databases in Section 2, we define vics in Section 3. We use a corrected version of the definition from [8]. We also introduce the notion of a *generalized vic (gvic)* that succinctly represents all vics holding in a relation. From now on, we will talk mostly about gvics.

Second, since gvics cannot be easily checked syntactically, it is essential to have efficient methods for *testing* whether a gvic holds in a constraint relation. This issue is addressed in Section 4. A new, geometric, PTIME approach to testing the satisfaction of gvics in constraint relations with linear arithmetic constraints is demonstrated.

Third, the applications of gvics require tools for *gvic inference* in query results. To obtain closure (in view of Example 1.1), a subclass of relational algebra with aggregation is defined in Section 6. The aggregation queries belonging to this subclass need to satisfy appropriate gvics. Also, data interoperability of restricted spatiotemporal relations (Section 7) requires the ability to infer gvics holding in the the result of a relational algebra operation. We present a suitable inference system in Section 5, in which, to obtain completeness, we generalize gvics to *disjunctions* of gvics. Using the same inference system, we show how to perform the inference of gvics in Constraint Datalog programs.

To complete the outline of the paper, we should mention that in Section 7, we present further application of variable independence in temporal and constraint databases. In Section 8, we discuss related work. We conclude in Section 9, outlining several directions for further research.

2 The Constraint Database Framework

2.1 Constraint Databases

In traditional database theory, a k-ary *relation* is a finite set of k-tuples (or points in a k-dimensional space) and a *database* is a finite set of relations. However, the relational calculus and algebra can be developed without the finiteness assumption for relations. We will use the term *unrestricted relation* for finite or infinite sets of points in a k-dimensional space. In order to be able to do something useful with such unrestricted relations, we need a finite representation that can be manipulated. This is exactly what is provided by *constraint* tuples and relations. Constraint tuples are quantifier-free formulas over a given signature, interpreted in a structure [19]. We use here the definitions from [35] that have been slightly modified. The first three definitions are standard in the model theory of classical first-order logic.

Definition 2.1. *A signature Ω consists of three sets: a set $\mathcal{F}$ of function symbols, a set $\mathcal{P}$ of predicate symbols, and a set $\mathcal{C}$ of constant symbols, together with an arity function that associates a natural number to each element of $\mathcal{F}$ and $\mathcal{P}$. We assume that a signature always contains the equality and inequality predicate symbols.*

In the following definitions, Ω will denote a fixed signature.

Definition 2.2. *Given a set $\mathcal{U}$, an Ω-*structure *$\mathcal{M}$ on $\mathcal{U}$ is defined by assigning to each $f \in \mathcal{F}$ of arity n a function $f^{\mathcal{M}} : \mathcal{U}^n \to \mathcal{U}$, to each $P \in \mathcal{P}$ an n-ary relation on $\mathcal{U}$ (a set $P^{\mathcal{M}} \subseteq \mathcal{U}^n$), and to each $c \in \mathcal{C}$ an element $c^{\mathcal{M}} \in \mathcal{U}$. The set $\mathcal{U}$ is called the* universe *of the structure.*

Definition 2.3. *An Ω-*constraint *is an atomic first-order formula over Ω or its negation.*

Example 2.4. The following classes of constraints are of special interest in this paper:

- *polynomial inequality constraints*: the constraints over

  $$\Omega = (+, *, <, 0, 1),$$

  where $\mathcal{F} = \{+, *\}$, $\mathcal{P} = \{<\}$, and $\mathcal{C} = \{0, 1\}$,
- *linear inequality constraints*: the constraints over

  $$\Omega = (+, <, 0, 1),$$

- *order constraints*: the constraints over

  $$\Omega = (<, \{c\}_{c \in C}),$$

  where C is a set of constant symbols corresponding to the elements of some fixed universe, and
- *equality constraints*: the constraints over

  $$\Omega = (\{c\}_{c \in C}).$$

We will interpret the polynomial and linear constraints over the reals or the rationals, the order constraints over the rationals or the integers, and the inequality constraints over arbitrary infinite sets.

Definition 2.5.

1. *A* constraint k-tuple, *in variables $x_1, \ldots, x_k$, over Ω is a finite conjunction $\alpha_1 \wedge \cdots \wedge \alpha_N$, where each α_i, $1 \leq i \leq N$, is an* Ω-constraint *with variables that are among $x_1, \ldots, x_k$.*
2. *A* constraint relation *of arity k over Ω is a finite set $S = \{\beta_1, \ldots, \beta_M\}$, where each β_i, $1 \leq i \leq M$, is a constraint k-tuple in the same variables $x_1, \ldots, x_k$ over Ω.*
3. *The* schema *of the constraint relation S is the pair consisting of the relation name S and the set of variables $\{x_1, \ldots, x_k\}$.*
4. *The* formula *ϕ_S corresponding to the constraint relation S is the disjunction $\beta_1 \vee \cdots \vee \beta_M$.*
5. *A* constraint database *is a finite collection of constraint relations.*

Definition 2.6. *Let $\mathcal{M}$ be an Ω-structure on a universe $\mathcal{U}$, S a constraint relation of arity k over Ω, and ϕ_S the formula corresponding to S. Then, S* represents *the* unrestricted relation

$$R = \{(a_1, \ldots, a_k) \in \mathcal{U}^k \mid \mathcal{M} \models \phi_S(a_1, \ldots, a_k)\}.$$

In the rest of the paper, we use S_i to denote constraint relations and R_i to denote the unrestricted relations represented by S_i. There is a clear correspondence between the attributes (variables) of a constraint relation and those of the corresponding unrestricted relation, so from now on we will not distinguish between them. In particular, we will assume that they have the same names.

2.2 Constraint Query Languages

In the following, we assume that we are given not only a signature Ω but also a constraint database schema Γ consisting of a finite number of constraint relation schemas.

Definition 2.7. *A relational calculus formula over Γ is a first-order formula over the signature Ω expanded with the relation names from Γ.*

The semantics of relational calculus formulas viewed as queries are defined on unrestricted relations, analogously to traditional database relations.

Definition 2.8. *Let $\mathcal{M}$ be an Ω-structure on $\mathcal{U}$, ϕ a relational calculus formula over Ω, and D an unrestricted database with schema Γ. Let $(\mathcal{M}, D)$ be $\mathcal{M}$ expanded with the interpretations in D for every relation name in Γ. The formula ϕ expresses the following query:*

$$Q(D) = \{(a_1, \ldots, a_k) \in \mathcal{U}^k \mid (\mathcal{M}, D) \models \phi(a_1, \ldots, a_k)\}.$$

Clearly, relational calculus formulas will be evaluated over constraint relations, not unrestricted ones. This can be done in one of two ways: 1) substituting the formula ϕ_S for every relation name S, eliminating the quantifiers from the result and transforming to DNF, or 2) translating to relational algebra over unrestricted relations and subsequently evaluating the resulting expression over the corresponding constraint relations. In both cases, the underlying Ω-structure $\mathcal{M}$ needs to admit effective quantifier elimination.

Definition 2.9. *An Ω-structure $\mathcal{M}$ is said to admit* quantifier elimination *if for every first-order formula $\phi(x_1, \ldots, x_k)$ over Ω there exists a quantifier-free formula $\psi(x_1, \ldots, x_k)$ over Ω such that the formula*

$$\forall x_1, \ldots, x_k(\phi \leftrightarrow \psi)$$

is valid in $\mathcal{M}$. If such a ψ can be effectively computed, then the quantifier elimination is said to be effective.

For example, it is well-known [19], [28] that all the structures in Example 2.4 admit effective quantifier elimination.[1]

We show below how relational algebra expressions over unrestricted relations can be evaluated on the corresponding constraint relations. See [13] for a longer discussion.

Let Ω be a signature and $\mathcal{M}$ an Ω-structure. Let R_1 and R_2 be the unrestricted relations represented in the context of $\mathcal{M}$ (Definition 2.6) by the constraint relations S_1 and S_2, respectively. We show how the result of applying an arbitrary PJRUN-algebra operator to R_1 and R_2 can be represented as a constraint relation:

P. $\pi_X(R_1)$ is represented by the constraint relation $\{QE_X(t) \mid t \in S_1\}$, where $QE_X(t)$ is the result of applying a quantifier elimination procedure for $\mathcal{M}$ to t (QE_X eliminates variables *not in* X).

J. $R_1 \bowtie R_2$ is represented by $\{t_1 \wedge t_2 \mid t_1 \in S_1, t_2 \in S_2\}$.

R. $R_1\theta$ is represented by $\{t\theta \mid t \in S_1\}$, where θ is a variable renaming.

U. $R_1 \cup R_2$ is represented by $S_1 \cup S_2$.

N. $R_1 - R_2$ is represented by $\{t_1 \wedge t_2 \mid t_1 \in S_1, t_2 \in (S_2)^c\}$, where S^c is the set of disjuncts (viewed as constraint tuples) of a DNF formula equivalent to $\neg\phi_S$.

Note that we have omitted the selection operation $\sigma_\varphi(R)$. In the constraint setting, this operation can be replaced by a natural join with a unrestricted relation represented by the (singleton) constraint relation $\{\varphi\}$.

The constraint database framework requires that for each input, constraint queries must be evaluable in closed form:

Definition 2.10. *A query Q over unrestricted relations is* closed *over an Ω-structure $\mathcal{M}$ if, for every unrestricted database D that is finitely representable using a constraint relation over $\mathcal{M}$, $Q(D)$ (the result of applying Q to D) is also finitely representable using a constraint relation over $\mathcal{M}$ and the latter representation can be effectively obtained from the representation of D. A class of queries is closed if every query in this class is closed.*

Proposition 2.11. *If an Ω-structure $\mathcal{M}$ allows effective quantifier elimination, then the relational calculus and algebra as defined above are closed over $\mathcal{M}$.*

1. Note that we will not interpret $\Omega = (+, *, <, 0, 1)$ over the integers because, in this case, it is well-known that the structure does not admit quantifier elimination. Integers with order admit effective quantifier elimination if not only order but also gap-order constraints are allowed (see Section 7.3).

This proposition implies that the relational algebra and calculus over all the classes of constraints and structures listed in Example 2.4 are closed.

3 Independence of Variables

In this section, we define *variable independence* for unrestricted relations that can be finitely represented using constraint relations.

3.1 Variable Independence Condition

In the following definitions, we assume that R is a constraint relation over a finite set U of variable (attribute) identifiers, and X and Y are subsets of U. As is common in database theory, we write XY for $X \cup Y$ and omit curly brackets for singleton sets.

Definition 3.1. *Let t be a constraint tuple over U, T the unrestricted relation corresponding to $\{t\}$, and R an unrestricted relation over U. We say that t satisfies a* variable independence condition (vic) *$\mathcal{I}(X)$ if*

$$T = \pi_X(T) \bowtie \pi_{U-X}(T).$$

Furthermore, we say that R satisfies a vic $\mathcal{I}(X)$ if there is a constraint relation S that finitely represents R, in which every constraint tuple satisfies $\mathcal{I}(X)$.

While variable independence is a property of *unrestricted* relations, in the constraint database framework we can only manipulate *finite* constraint representations of such relations: the constraint relations. Therefore, the task of determining if a vic $\mathcal{I}(X)$ holds in an unrestricted relation R reduces to determining if, given a *particular* constraint relation S representing R, there is (a possibly different) constraint relation S' also representing R such that all constraint tuples in S' satisfy $\mathcal{I}(X)$. In this case, we also say that S satisfies $\mathcal{I}(X)$.

Example 3.2. We return to the constraint relation of Example 1.2, consisting of tuples of the form

$$t_1 < t < t_2 \wedge (n = N) \wedge C(x,y),$$

where $C(x,y)$ is a conjunction of linear arithmetic constraints describing a piece of land owned by person N over the time interval (t_1, t_2). The unrestricted relation represented by the above constraint relation satisfies the following vics: $\mathcal{I}(\{n\})$, $\mathcal{I}(\{t\})$, and $\mathcal{I}(\{x,y\})$. On the other hand, the relation R from Example 1.1 does not satisfy the vic $\mathcal{I}(\{x,y\})$.

We say that two disjoint subsets of U, X, and Y are *independent* in t if t satisfies $\mathcal{I}(Z)$, such that $X \subseteq Z$ and $Y \cap Z = \emptyset$, i.e., X and Y are *separated* in U by Z. We say that X and Y are *related* in t otherwise. Clearly, variable independence in a tuple is decidable for constraint theories admitting effective quantifier elimination. Similarly, we will say that X and Y are *independent* in R if R satisfies $\mathcal{I}(Z)$ for some set Z such that $X \subseteq Z$ and $Y \cap Z = \emptyset$, and *related* in R otherwise.

Definition 3.3. *Let R be an unrestricted relation, U the schema of R, and $X, Y \subseteq U$. Then, the following are axioms for inferring vics in R:*

1. $\mathcal{I}(\emptyset)$;
2. $\mathcal{I}(X) \Rightarrow \mathcal{I}(U - X)$;
3. $\mathcal{I}(X) \wedge \mathcal{I}(Y) \Rightarrow \mathcal{I}(XY)$.

Theorem 3.4. *The axioms in Definition 3.3 are sound and complete.*

Proof. Let $S = \{c_1, \ldots, c_k\}$ be a constraint relation that represents R, where c_i are constraints over U. An independence constraint $\mathcal{I}(X)$ is then equivalent to a MVD $\mathbf{id} \twoheadrightarrow X$ in an unrestricted relation $\{(i, a_1, \ldots, a_l) : c_i(a_1, \ldots, a_l), c_i \in S\}$ over the schema $\{\mathbf{id}\} \cup U$; $\mathbf{id}$ is a (hypothetical) tuple identifier of constraint tuples in S. The conclusion then follows from a (sound and complete) axiom system for MVDs:

1. $\mathcal{I}(\emptyset)$ follows immediately from the existence of the finite constraint relation S that represents R.
2. $\mathcal{I}(X) \Rightarrow \mathcal{I}(U - X)$ follows from the MVD inference $\mathbf{id} \twoheadrightarrow X \Rightarrow \mathbf{id} \twoheadrightarrow U - X$.
3. To show $\mathcal{I}(X) \wedge \mathcal{I}(Y) \Rightarrow \mathcal{I}(XY)$, we assume that R satisfies $\mathcal{I}(X)$ and $\mathcal{I}(Y)$. Then, there must be two constraint relations S (as above) and $S' = \{c'_1, \ldots, c'_{k'}\}$ in which every constraint tuple satisfies $\mathcal{I}(X)$ and $\mathcal{I}(Y)$, respectively. We construct an unrestricted relation $\{(i, i', a_1, \ldots, a_l) : c_i(a_1, \ldots, a_l) \wedge c_{i'}(a_1, \ldots, a_l), c_i \in S, c_{i'} \in S'\}$ over the schema $\{\mathbf{id}, \mathbf{id}'\} \cup U$. In this relation, $\mathbf{id} \twoheadrightarrow X$ and $\mathbf{id}' \twoheadrightarrow Y$. Using inference axioms for MVDs, we get $\mathbf{id}, \mathbf{id}' \twoheadrightarrow XY$. The (finite) set of (i, i') pairs then defines the elements of a constraint relation S'' that also represents R and in which every constraint tuple satisfies $\mathcal{I}(XY)$.

Completeness follows from the observation that, given a finite set of vic's $\mathcal{I}(X_1), \ldots, \mathcal{I}(X_n)$ holding in R, there must be finite representations $S_1, \ldots, S_n$ of R such that $\mathcal{I}(X_i)$ is satisfied by all constraint tuples in S_i $(0 < i \leq n)$. Then, however, we can construct an unrestricted relation

$$\{(i_1, \ldots, i_n, a_1, \ldots, a_l) : c^1_{i_1}(a_1, \ldots, a_l) \wedge \ldots \wedge c^n_{i_n}(a_1, \ldots, a_l), c^j_{i_j} \in S_j\}$$

from the given constraint relations S_j in which $i_j \twoheadrightarrow X_j$ for $1 \leq j \leq n$. Then, we simply use a complete inference system on the induced MVDs. □

The following corollary completely characterizes interactions of vics within a single relation.

Corollary 3.5. *The vics form a Boolean algebra with generators defined by the MVD basis for* **id**.

Example 3.6. In addition to the vics mentioned in Example 3.2, the unrestricted relation represented by the constraint relation of Example 1.2 satisfies all their Boolean combinations.

Note also that, for classical (finite) relations, where each tuple always represents exactly one point, it is *trivially true* that every relation (over U) satisfies vic $\mathcal{I}(X)$ for every $X \subseteq U$.

We view the set of variable independence conditions as a part of the schema of a constraint relation. All instances of such a schema have to satisfy the vics in this set. It may be

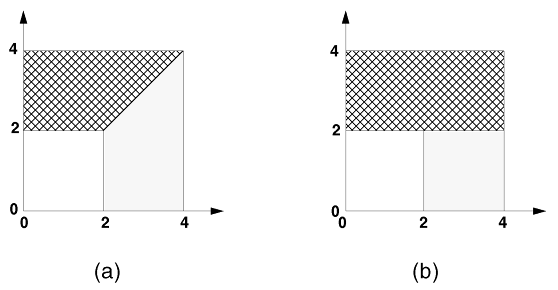


Fig. 2. Instance (a) is equivalent to instance (b).

the case that constraint relations are guaranteed to satisfy given vics by virtue of coming from a restricted data model. We will see an example of this in Section 7. In general, however, *decidability of variable independence* for constraint relations (as opposed to individual tuples) is far from obvious. As the next example shows, it is not sufficient to check the individual tuples in order to verify variable independence:

Example 3.7. The instance of $R(x,y)$ in Fig. 2a contains two tuples:

$$2 \le y \le 4 \wedge 0 \le x \le y;$$
$$2 \le x \le 4 \wedge 0 \le y \le x.$$

In each tuple, x and y are related. However, there is an an equivalent relation (Fig. 2b) where this is not the case:

$$2 \le y \le 4 \wedge 0 \le x \le 4;$$
$$2 \le x \le 4 \wedge 0 \le y \le 2.$$

In Section 4, we show how to test whether a constraint relation represents an unrestricted relation that satisfies a variable independence condition in its schema. This test, performed upon relation updates, ensures that we *maintain* the set of vics in the schemas of all relations in the constraint database.

3.2 Multiple Sets of Variables

The notion of a variable independence condition (Definition 3.1) refers to one set of attributes. We now generalize the notion of variable independence for tuples to an arbitrary number of such sets.

Definition 3.8. *Let $(X_1,\ldots,X_n)$ be a partition of U, t a generalized tuple over U, and T an unrestricted relation corresponding to $\{t\}$. We say that t satisfies a* generalized vic (gvic) $\mathcal{I}(X_1,\ldots,X_n)$ *if*

$$T = \pi_{X_1}(T) \bowtie \cdots \bowtie \pi_{X_n}(T).$$

The definition of variable independence for relations can be generalized in a similar way. Also, it is easy to see that a vic $\mathcal{I}(X)$ is equivalent to the gvic $\mathcal{I}(X, U-X)$ (and, thus, we can simply consider the notation for vics as a shorthand for the corresponding gvics).

Theorem 3.9. *Let $\{\mathcal{I}(X_1),\ldots,\mathcal{I}(X_k)\}$ be a finite set of vics. Then, there is a unique gvic $\mathcal{I}(Y_1,\ldots,Y_l)$ such that*

$$R\ satisfies\ \mathcal{I}(X_i)\ for\ all\ 0 < i \le k \Longleftrightarrow$$
$$R\ satisfies\ \mathcal{I}(Y_1,\ldots,Y_l).$$

Proof. Let $Y_1,\ldots,Y_l$ be the MVD basis of $\mathbf{id} \twoheadrightarrow X_i$ for $0 < i \le k$. The basis is unique and partitions the schema of R into the required disjoint sets. □

It is also easy to see that, if a gvic $\mathcal{I}(X_1,\ldots,X_k)$ holds in R, then the vics $\mathcal{I}(X_i)$ for $0 < i \le k$ hold in R. Therefore, the representation result extends to gvics:

Corollary 3.10. *Let F be a finite set of (g)vics. Then, F can be equivalently represented by a single gvic.*

Proof. Every gvic $\mathcal{I}(X_1,\ldots,X_k)$ is equivalent to the set of vics $\{\mathcal{I}(X_i) : 0 < i \le k\}$. The rest follows from Theorem 3.9. □

Example 3.11. In Example 3.2, the constraint relation satisfies the gvic $\mathcal{I}(\{t\},\{n\},\{x,y\})$.

Definition 3.12. *A gvic $\mathcal{I}(X_1,\ldots,X_k)$ is* finer *than a gvic $\mathcal{I}(Y_1,\ldots,Y_m)$ if for all $i = 1,\ldots,k$, there is a $j = 1,\ldots,m$ such that $X_i \subseteq Y_j$ and one of those containments is proper.*

4 Testing Variable Independence

Given a relation R whose schema restricts X to be independent, we need to make sure this restriction is not violated. One way of enforcing independence restrictions on R's schema is by stipulating that each tuple satisfy these restrictions. We give an example of this approach (Worboys relations) in Section 7. See also [15].

In general, to maintain a vic (or a gvic), we must be able to test its satisfaction in a relation. We can then reject the updates that violate it. (This is analogous to maintaining integrity constraints in traditional databases.) In this section, we provide such a *test* for linear constraint databases. Unlike the work of [22], where all tests are performed on constraint sets (i.e., tuples), this test is for disjunctions of constraint sets (i.e., relations).

4.1 Variable Independence for Semilinear Sets

Testing for variable independence is specific to each constraint class. In this section, we present a technique based on computational geometry [3], [18] that allows testing for variable independence of *semilinear* relations (constraint relations defined using linear arithmetic constraints).

Definition 4.1. *A semilinear relation is an unrestricted relation that can be represented by a constraint relation over $\Omega = \{+,<,0,1\}$ interpreted over the reals.*

At the end of the section, we generalize the result to relations that can be defined by more general polynomial (in)equality constraints.

Given a finite set of variables X, we denote by $\mathbb{R}^X$ the $|X|$-dimensional Euclidean space whose coordinate axes are labeled by the variable names in X.

Definition 4.2. *An* arrangement *$\mathcal{A}(H_1,\ldots,H_l)$ for a finite set $H_1,\ldots,H_l$ of hyperplanes in $\mathbb{R}^U$ is the coarsest common refinement of the partitions of $\mathbb{R}^U$ induced by each of the H_i.*

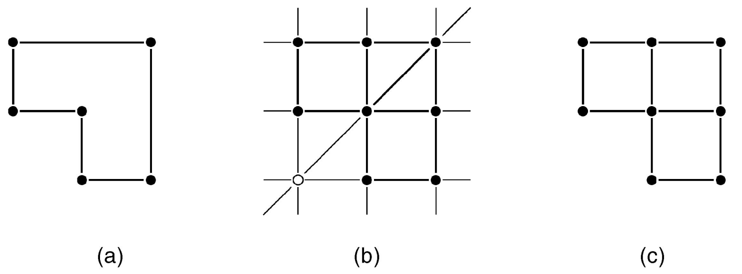


Fig. 3. Illustrating a dimensional decomposition and an arrangement.

The elements of $\mathcal{A}(H_1, \ldots, H_l)$ are open, disjoint, k-dimensional ($0 < k \leq |U|$) cells. There are $\mathcal{O}(l^{|U|})$ such cells in dimensions $0, \ldots, |U|$, and this bound is tight [18].

Lemma 4.3. *$R \subseteq \mathbb{R}^U$ is a semilinear set if and only if there are hyperplanes $H_1, \ldots, H_l$ in $\mathbb{R}^U$ such that $R = \bigcup_{q \in I} q$ for some $I \subseteq \mathcal{A}(H_1, \ldots, H_l)$.*

Proof. Each constraint relation S that represents an unrestricted semilinear relation R can be defined as a DNF of semilinear constraints. The corresponding linear constraints define the hyperplanes H_i. □

Given a constraint relation S that represents an unrestricted semilinear relation R over U, we define $\mathcal{A}_S$ to be an arrangement in $\mathbb{R}^U$ induced by the hyperplanes defined by the constraints in S (each constraint of the form $\mathbf{a} \cdot \mathbf{x} > k \in S$ is associated with the hyperplane $\mathbf{a} \cdot \mathbf{x} = k$). To test if R satisfies a vic $\mathcal{I}(X)$, we introduce the notion of X-rectangular decomposition:

Definition 4.4. *A pair of finite arrangements $\mathrm{part}(X)$ in $\mathbb{R}^X$ and $\mathrm{part}(U - X)$ in $\mathbb{R}^{U-X}$ is called a* X-rectangular decomposition *of R if for all*

$$c \in \{x \times y : x \in \mathrm{part}(X), y \in \mathrm{part}(U - X)\},$$

we have $c \subseteq R$ or $c \cap R = \emptyset$.

In other words, R is equivalent to a finite union of products of pairs of cells taken from $\mathrm{part}(X)$ and $\mathrm{part}(U - X)$, respectively. Fig. 3 shows (a) the dimensional decomposition, (b) the arrangement $\mathcal{A}_S$, and (c) the resulting semilinear set that satisfies the vic $\mathcal{I}(\{x\})$ for the constraint relation introduced in Fig. 2a and discussed in Example 3.7.

Lemma 4.5. *Let R be a semilinear set and x and y two points in $\mathbb{R}^U$ such that 1) $x \in R$ and $y \notin R$, and 2) $\pi_{U-X}(x) = \pi_{U-X}(y)$. Then, $\pi_X(x)$ and $\pi_X(y)$ belong to different cells in $\mathrm{part}(X)$ of every X-rectangular decomposition of R.*

Proof. Immediate from the observation that $x \in R$ and $y \notin R$. Thus, they must belong to different rectangular cells. However, as they share their $U - X$ coordinates, the difference must be in $\mathrm{part}(X)$. □

Now, we use the arrangement $\mathcal{A}_S$ to define a pair of arrangements in $\mathbb{R}^X$ and $\mathbb{R}^{U-X}$ as follows:

Definition 4.6. *Let R be a semilinear relation in $\mathbb{R}^U$ represented by a constraint relation S, X a subset of U, and $\mathcal{A}_S$ a partition of $\mathbb{R}^U$ derived from semilinear constraints in S. We define $\mathrm{part}_S(X)$ to be the arrangement in $\mathbb{R}^X$ induced by projecting elements of $\mathcal{A}_S$ to $\mathbb{R}^X$ one by one.*

We claim that, if S represents an unrestricted relation R that satisfies $\mathcal{I}(X)$, then the pair of arrangements $\mathrm{part}_S(X)$ and $\mathrm{part}_S(U - X)$ in the respective subspaces $\mathbb{R}^X$ and $\mathbb{R}^{U-X}$ represent a *canonical* X-rectangular decomposition of R. The crucial observation is that, while an unrestricted semilinear relation R may be represented by many (different) constraint relations S', all the arrangements $\mathcal{A}_{S'}$ generated from S' share a common property:

Lemma 4.7. *Let $R \subseteq \mathbb{R}^U$ be a semilinear relation and S an arbitrary constraint relation that represents R. Then, every element of the (dimensional) decomposition[2] of R and $\mathbb{R}^U - R$ (R's complement) of dimension $k \leq |U|$ is contained in a finite union of elements of $\mathcal{A}_S$ of dimension at most k.*

Thus, every arrangement $\mathcal{A}_S$ contains a representation of every face of R; the lemma follows from results on real semialgebraic sets [30], [3] and is similar to techniques used in [12].

Theorem 4.8. *Let R be a semilinear relation in $\mathbb{R}^U$ represented by a constraint relation S. Then, R satisfies a variable independence constraint $\mathcal{I}(X)$ if and only if $\mathrm{part}_S(X), \mathrm{part}_S(U - X)$ is an X-rectangular decomposition of R.*

Proof. The "if" direction of the proof is immediate: R is a finite union of rectangular cells of the form $x \times y$, where $x \in \mathrm{part}_S(X)$ and $y \in \mathrm{part}_S(U - X)$ and therefore can be represented by a finite set of constraint tuples, each of which satisfies $\mathcal{I}(X)$.

To show "only-if," assume that R is a semilinear set that satisfies the independence constraint $\mathcal{I}(X)$ but for which $\mathrm{part}_S(X)$ and $\mathrm{part}_S(U - X)$ do not form a X-rectangular decomposition. Therefore, there must be a $x \in \mathrm{part}_S(X)$ and $y \in \mathrm{part}_S(U - X)$ that only partially intersects R, i.e., $x \times y \not\subseteq R$ and $(x \times y) \cap R \neq \emptyset$.

However, since R satisfies the vic $\mathcal{I}(X)$, there must be an X-rectangular decomposition $\mathrm{part}(X), \mathrm{part}(U - X)$ of R different from $\mathrm{part}_S(X), \mathrm{part}_S(U - X)$. Without loss of generality, we assume that $\mathrm{part}(X), \mathrm{part}(U - X)$ is a refinement of $\mathrm{part}_S(X), \mathrm{part}_S(U - X)$ since otherwise we simply take the coarsest common refinement of

2. A *decomposition of a semialgebraic set* is a finite union of semialgebraic sets, each of them maximal with regard to set inclusion and homeomorphic to the open unit cube in $\mathbb{R}^k$ for $0 < k \leq n$. Such a decomposition always exists and is unique [3], Theorem 2.3.6.

$\mathrm{part}(X), \mathrm{part}(U-X)$ and $\mathrm{part}_S(X), \mathrm{part}_S(U-X)$ which must also be a rectangular decomposition of R.

Also, as $\mathrm{part}_S(X), \mathrm{part}_S(U-X)$ is not an X-rectangular decomposition, $\mathrm{part}(X), \mathrm{part}(U-X)$ must be a *proper refinement* of $\mathrm{part}_S(X), \mathrm{part}_S(U-X)$: All the cells $x \times y$ that only partially intersected R must have been refined in $\mathrm{part}(X), \mathrm{part}(U-X)$. Thus, we can find two points x and y such that

1. $x \in R$ and $y \notin R$,
2. $\pi_{U-X}(x) = \pi_{U-X}(y)$, and
3. $\pi_X(x), \pi_X(y)$ belong to the same cell in $\mathrm{part}_S(X)$ (the other case, swapping X and $U-X$, is symmetric).

However, by Lemma 4.5, x and y must then belong to different (and thus disjoint) cells in $\mathrm{part}(X)$. This is not possible as $\mathrm{part}_S(X)$ is an arrangement in $\mathrm{I\!R}^X$ that already contains images of all faces of R (by Lemma 4.7, as every such face is an element of the decomposition of R or $\mathrm{I\!R}^U - R$). Therefore, the element(s) separating x from y must already have been reflected in $\mathrm{part}_S(X)$, a contradiction.

Therefore, $\mathrm{part}_S(X), \mathrm{part}_S(U-X)$ must be a rectangular decomposition of R whenever R satisfies the vic $\mathcal{I}(X)$. □

The theorem and proof can be immediately generalized to decide if a gvic $\mathcal{I}(X_1, \ldots, X_k)$ holds in a semilinear relation R represented by a constraint relation S. The generalization merely considers projections of $\mathcal{A}_S$ into all of the subspaces X_i, rather than to X and $U-X$ only. The remainder of the argument is essentially the same.

Theorem 4.9. *Let S be a constraint relation over U and $X \subseteq U$, where all constraints are semilinear. Then, testing if S represents an unrestricted relation that satisfies $\mathcal{I}(X)$ takes $\mathcal{O}(n^{|U|})$, where n is the number of individual constraints in S. This bound is tight.*

Proof. n hyperplanes define an arrangement containing $\mathcal{O}(n^{|U|})$ cells in all dimensions; and the arrangement can be computed in time and space $\mathcal{O}(n^{|U|})$ [18]. This bound is tight: There are arrangements with $\Omega(n^{|U|})$ cells. Projecting the cells in the subspaces X and $U-X$ and computing the arrangements $\mathrm{part}_S(X)$ and $\mathrm{part}_S(U-X)$ will be dominated by the above cost. Testing for containment/disjointness of the rectangular cells with regard to S is a FO+LIN query. Altogether, the overall complexity of testing if a constraint relation represents an unrestricted relation satisfying $\mathcal{I}(X)$ is $\Theta(n^{|U|})$. □

We use the number of individual constraints rather than the number of constraint tuples since the individual constraints over U are bounded in size while constraint tuples are not. Therefore, measuring the size of S in the number of constraints is more appropriate.

In addition to testing whether a constraint relation represents an unrestricted relation that satisfies a vic $\mathcal{I}(X)$, our approach also *constructs* an alternative representation for the unrestricted relation in which every constraint tuple satisfies the vic $\mathcal{I}(X)$.

Similarly to semilinear relations, we can use the same construction to test for variable independence in constraint relations defined by *polynomial inequalities*. The complexity of the procedure remains unchanged [18].

5 Inference of Variable Independence in Query Results

This section introduces an inference system that allows us to deduce variable independence constraints in results of queries. In other words, we develop a system that allows us to derive judgments of the form

$$R_1 : \mathcal{I}_1, \ldots, R_k : \mathcal{I}_k \vdash Q : \mathcal{I}_Q,$$

where $\mathcal{I}_j$ are gvics [3] holding in relations R_j and Q is a query expression over these relations. In the rest of this section, we use the standard notation $\Gamma \vdash R : \mathcal{I}$ and $\Gamma \models R : \mathcal{I}$ to denote the facts that $R : \mathcal{I}$ is derivable from and logically implied by a finite set of assumptions $\Gamma = \{R_1 : \mathcal{I}_1, \ldots, R_k : \mathcal{I}_k\}$, respectively.

5.1 Disjunctions of Gvics

As the following example demonstrates, using single gvics to describe constraints holding in an expression cannot yield a sufficiently powerful set of inference rules:

Example 5.1. Consider the query $Q = \pi_{\{xz\}}(R \cup S)$, where the relations R and S with the same schema $\{x, y, z\}$ satisfy the gvics $\mathcal{I}(\{x\}, \{y, z\})$ and $\mathcal{I}(\{x, y\}, \{z\})$, respectively. It is easy to verify that $Q : \mathcal{I}(\{x\}, \{z\})$. However, the strongest gvic that holds in the subquery $R \cup S$ is $\mathcal{I}(\{x, y, z\})$ and, therefore, we cannot design a *sound and complete* inference rule for the projection (π_V) operation.

We therefore generalize our inference system to allow *disjunctions* of gvics to be attached to query expressions:

Definition 5.2. *Let $\mathcal{I}_1, \ldots, \mathcal{I}_k$ be a set of gvics. An unrestricted relation R satisfies $\mathcal{I}_1, \ldots, \mathcal{I}_k$ (we write $R : \mathcal{I}_1, \ldots, \mathcal{I}_k$) if there is a constraint relation S that finitely represents R such that each constraint tuple satisfies at least one of the gvics $\mathcal{I}_j$ for $0 < j \leq k$.*

Note that there are only finitely many gvics for a fixed schema, and that it is, therefore, sufficient to consider only finite sets of gvics. Before defining the inference rules, we need the following operations on (sets of) gvics:

Definition 5.3 (gvic operations). *Let $\mathcal{I}(X_1, \ldots, X_k)$ and $\mathcal{I}'(Y_1, \ldots, Y_l)$ be two gvics over the schemas X and Y, respectively. We define*

- *$\mathcal{I}(X_1, \ldots, X_k)\theta = \mathcal{I}(X_1\theta, \ldots, X_k\theta)$, where θ is a renaming of variables X to $X\theta$;*
- *$\pi_V \mathcal{I}(X_1, \ldots, X_k) = \mathcal{I}(V \cap X_1, \ldots, V \cap X_k)$, where $V \subseteq X$ is a set of variables;*
- *$\mathcal{I}(X_1, \ldots, X_k) \bowtie \mathcal{I}'(Y_1, \ldots, Y_l) = \mathcal{I}(Z_1, \ldots, Z_n)$, where $Z_i \subseteq X \cup Y$ and $Z_1, \ldots, Z_n$ is the finest partition of $X \cup Y$ such that for all $0 < i \leq k$ either $X_i \subseteq Z_j$ or $X_i \cap Z_j = \emptyset$ and for all $0 < i \leq l$ either $Y_i \subseteq Z_j$ or $Y_i \cap Z_j = \emptyset$.*

These operations essentially mimic the corresponding relational operations on constraint tuples; we therefore use the same names. The actual inference rules are defined in terms of these operations in Fig. 4. Given a query expression

3. We use gvics here as a single gvic that completely characterizes all vics in a relation.

AX.	$\Gamma \vdash R : \mathcal{I}$	for $R : \mathcal{I} \in \Gamma$
RR.	$\dfrac{\Gamma \vdash Q : \mathcal{I}_1, \dots, \mathcal{I}_k}{\Gamma \vdash Q\theta : \mathcal{I}_1\theta, \dots, \mathcal{I}_k\theta}$	for θ a renaming
RP.	$\dfrac{\Gamma \vdash Q : \mathcal{I}_1, \dots, \mathcal{I}_k}{\Gamma \vdash \pi_V Q : \pi_V \mathcal{I}_1, \dots, \pi_V \mathcal{I}_k}$	for V a set of variables
RJ.	$\dfrac{\Gamma \vdash Q : \mathcal{I}_1, \dots, \mathcal{I}_k \qquad \Gamma \vdash Q' : \mathcal{I}'_1, \dots, \mathcal{I}'_k}{\Gamma \vdash Q \bowtie Q' : \mathcal{I}_1 \bowtie \mathcal{I}'_1, \dots, \mathcal{I}_k \bowtie \mathcal{I}'_l}$	
RU.	$\dfrac{\Gamma \vdash Q : \mathcal{I}_1, \dots, \mathcal{I}_k \qquad \Gamma \vdash Q' : \mathcal{I}'_1, \dots, \mathcal{I}'_k}{\Gamma \vdash Q \cup Q' : \mathcal{I}_1, \dots, \mathcal{I}_k, \mathcal{I}'_1, \dots, \mathcal{I}'_l}$	for Q and Q' union-compatible
RN.	$\dfrac{\Gamma \vdash Q : \mathcal{I}_1, \dots, \mathcal{I}_k \qquad \Gamma \vdash Q' : \mathcal{I}'_1, \dots, \mathcal{I}'_k}{\Gamma \vdash Q - Q' : \mathcal{I}_1 \bowtie \mathcal{I}', \dots, \mathcal{I}_k \bowtie \mathcal{I}'}$	for $\mathcal{I}' = \mathcal{I}'_1 \bowtie \dots \bowtie \mathcal{I}'_l$ and Q and Q' union-compatible

Fig. 4. Gvic inference rules for PJRUN expressions.

Q, the rules are applied bottom-up following the syntactic structure of Q.[4]

Definition 5.4 (gvic Inference). *Let Γ be a set of assumptions of the form $R_j : \mathcal{I}_j$. We say that $Q : \mathcal{I}$ is* derivable *from Γ if $\Gamma \vdash Q : \mathcal{I}_1, \dots, \mathcal{I}_l$ is derived by the rules in Fig. 4 and $\mathcal{I} = \mathcal{I}_1 \bowtie \dots \bowtie \mathcal{I}_l$.*

To see the inference rules at work, consult Example 7.7. As we saw in Example 5.1, in general single gvics are not powerful enough to fully characterize independence in queries. However, it is easy to see that gvics are sufficient in the restricted case of conjunctive (PJR) queries: None of the AX, RR, RP, and RJ inference rules can infer disjunction of gvics from single gvics.

Now, we turn our attention to soundness and completeness of the inference rules in Fig. 4

Theorem 5.5 (Soundness). *The inference rules in Fig. 4 are sound; i.e.,*

$$\Gamma \vdash Q : \mathcal{I}_1, \dots, \mathcal{I}_k \Longrightarrow \Gamma \models Q : \mathcal{I}_1, \dots, \mathcal{I}_k.$$

Proof. By inspection of the rules and an inductive extension to query expressions. □

Similarly to other nontrivial properties of first-order queries, variable independence in query results is not decidable.

Theorem 5.6. *For arbitrary PJRUN queries, it is undecidable whether $\Gamma \models Q : \mathcal{I}$.*

Proof. By reduction from query emptiness. Let Q' be a closed (Boolean) PJRUN query and R a constraint relation over the schema $\{x, y\}$ containing a single tuple $\{x = y\}$. Then, $\emptyset \models (Q' \bowtie R) : \mathcal{I}(\{x\})$ if and only if Q' is empty. □

We, therefore, concentrate on the positive fragment of first-order queries—on the PJRU expressions.

4. There is only one deterministic way of applying the rules.

Theorem 5.7 (Completeness for PJRU Queries). *Let $\mathcal{M}$ be a constraint domain that contains equality and Q a PJRU query. Then,*

$$\Gamma \models Q : \mathcal{I}_1, \dots, \mathcal{I}_k \Longrightarrow \Gamma \vdash Q : \mathcal{I}_1, \dots, \mathcal{I}_k.$$

Proof. Let

$$R : \mathcal{I}_1(X^1_1, \dots, X^1_{k_1}), \dots, \mathcal{I}_l(X^l_1, \dots, X^l_{k_l}) \in \Gamma$$

be the (disjunction of) gvics imposed on R by Γ, where R is an unrestricted relation in the schema of the constraint database. We construct an instance D that satisfies Γ as follows: For the relation symbol $R(x_1, \dots, x_l)$, we define an unrestricted instance represented by the following constraint relation

$$S = \left\{ \left(\bigwedge_{x_i, x_j \in X^m_n, 0 < i < j \le l, 0 < n \le k_m} x_i = x_j \right) : 0 < m \le l \right\}.$$

The above instance *maximally* satisfies the assumptions posed on R in Γ; that is, for each gvic $\mathcal{I}_m$, the constraint relation S contains a constraint tuple that satifies the gvic $\mathcal{I}_m$, but does not satisfy any gvic finer than $\mathcal{I}_m$ (in the sense of Definition 3.12).

Applying a relational operation $f \in \{\theta, \pi_V, \bowtie, \cup\}$ on this instance produces a constraint relation representing the result $f(D)$ that maximally satisfies the gvics inferred by the inference rules in Fig. 4 for the $f(D)$. The result $f(D)$ can be added to the constraint database instance constructed above as a view. This view maximally satisfies the gvics inferred by our inference system and can be used as an input to further relational operations. The theorem then follows by extending this construction to (arbitrary) larger queries by composing the above single-step completeness argument. □

Taking into account the disjunctive nature of the inference system, we have started with Γ containing more general assumptions of the form $R_i : \mathcal{I}_{i,1}, \dots, \mathcal{I}_{i,k}$ and generated multiple maximal constraint tuples to maximally satisfy such disjunctions of gvics. However, without loss of generality, the assumptions for base relations R can all be of the form $R : \mathcal{I}$.

The restriction to constraint domains that *contain* equality (or another symmetric-transitive binary relation that cannot be decomposed to product of unary relations) is essential—it allows us to construct counterexamples for gvics not inferred by the rules in Fig. 4. Without equality the inference system is incomplete even for conjunctive queries. Consider the following example:

Example 5.8. Let all constraints be defined using $<_Q$ only (i.e., the only nontrivial interpreted constraint is the strict order on rational numbers). In this setting, for any binary relation $R(x, y)$, we have

$$R \bowtie (R[x \mapsto y, y \mapsto x]) : \mathcal{I}(\{x\})$$

independently of the gvic(s) satisfied by R itself (where $[x \mapsto y, y \mapsto x]$ is a renaming that swaps x and y in R).

A similar example can be exhibited for the $\cup$ operation as well.

5.2 Constant Constraints and Constraint Selection

The inference system in the previous section concentrates on inference of gvics from other gvics and ignores interactions between gvics and *constant* constraints. In particular, we express selection as a natural join with a singleton constraint relation.

Definition 5.9. *Let C be a constraint tuple and $\mathcal{I}$ a gvic. We say that C* inherently satisfies *$\mathcal{I}$ if $\sigma_{C'}C : \mathcal{I}$ for all constraint tuples C'.*

This definition captures the relationship between constant constraint selections and gvics. In particular, this definition captures the fact that selections based on constraint tuples satisfying this condition guarantee variable independence in the result no matter how the input to the selection may look. The following example shows that gvics interact with the selection in a more complex way:

Example 5.10. Consider the query $Q = \sigma_{x=a}(R)$ applied on a binary relation $R(x, y)$. Then, $Q : \mathcal{I}(\{x\})$.

Clearly, any query that returns finite (unrestricted) results for a particular attribute satisfies a vic separating this attribute from the rest as long as the constant constraints can be defined in the underlying constraint structure. In general, however, *finiteness* may not be the only cause of inherent satisfaction of a gvic: This depends on constraints definable in the underlying structure.

If an unrestricted relation inherently satisfies a gvic, then this gvic is preserved by PJRU relational operations (unlike normal gvics).

Example 5.11. Consider again the query $Q = \sigma_{x=a}(R)$. Then, in any conjunctive query Q' that uses Q as a subquery the vic $\mathcal{I}\{x\}$ holds (assuming x is a top-level variable in Q'). This fact cannot be inferred by the gvic inference rule for $\bowtie$.

There is another reason why not to include vic inference based on finiteness into the inference system: In general, while finiteness implies gvics, i.e., the existence of a finite constraint representation in which every tuple satisfies a gvic, there may be no limit on the size of such representation:

Example 5.12. Let $\mathbb{Z} = (\mathbb{Z}, \leq, \{c\}_{c \in C})$ be a linearly ordered set of integers with constants. Then, every bounded set in $R(x_1, \ldots, x_k) \subset \mathbb{Z}^k$ satisfies the gvic $\mathcal{I}(\{x_1\}, \ldots, \{x_k\})$.

In particular,

$$\sigma_{c_1 \leq x \leq c_2 \wedge c_3 \leq y \leq c_4} R$$

satisfies $\mathcal{I}(\{x\}, \{y\})$ for any instance of R, in particular for $R = \{x = y\}$. The result of this query is bounded by $\min(c_2, c_4) - \max(c_1, c_3)$ and independent of $|R|$.

5.3 Inference of Gvics in Constraint Datalog

The rules in Fig. 4 can also be used to deduce gvics holding in relations defined by Constraint Datalog queries [19].

We use the inference rules in Fig. 4 to define an abstract immediate consequence operator that, given a set (disjunction) of gvics for relation symbols in a Datalog program P, computes the immediate gvic consequences of the clauses in P. Iterating this operator yields the desired result. More formally:

Definition 5.13. *Let P be a Constraint Datalog program. We define*

$$\begin{aligned}\mathrm{TP}_{\mathcal{I}}(I) = \{(\pi_V(\mathcal{I}_C \bowtie \mathcal{I}_1\theta_1 \bowtie \ldots \bowtie \mathcal{I}_k\theta_k))\theta : A \\ \leftarrow C, B_1, \ldots, B_k \in P,\ \mathcal{I}_j : B_j \in I\},\end{aligned}$$

where $\mathcal{I}_C$ is the gvic associated with the constraint C, V is the set of variables in the head of the clause, A, θ is the renaming from the variables in the clause to the schema of A, and θ_i a renaming from schema of B_i to the variables in the clause.

Definition 5.14. *Let P be a Constraint Datalog program. Then,*

1. *The closure ordinal of $\mathrm{TP}_{\mathcal{I}}$ is finite, and*
2. *If $R : \mathcal{I}_1, \ldots, R : \mathcal{I}_k \in (\mathrm{TP}_{\mathcal{I}})^{\omega}(\emptyset)$ are all gvics associated with R, then $R : \mathcal{I}_1 \bowtie \ldots \bowtie \mathcal{I}_k$ holds in the answers of P. Moreover, the inference procedure is complete; no finer (Definition 3.12) gvic holds in the answer.*

Proof. Item 1 follows from the fact that there are only finitely many different gvics over a fixed schema, and Item 2 follows from Theorems 5.5 and 5.7 and the fact that the closure ordinal of $\mathrm{TP}_{\mathcal{I}}$ is finite. □

The inference of gvics in the results of deductive queries can be naturally relativized to an extensional database (*EDB*); it is sufficient to provide gvics describing the relations in the EDB as the input to the $\mathrm{TP}_{\mathcal{I}}$ iteration.

6 Restricted Aggregation

As shown in [21], Klug's relational calculus and algebra with aggregation can be extended to constraint databases with minor modifications, at least as far as the underlying semantics on unrestricted relations is concerned. For our purposes here, it is sufficient to assume that the algebra is extended by the following operator for every aggregate function, whose syntax was originally defined in [20]:

Definition 6.1. *Let R be an unrestricted relation over U, $X \subseteq U$, $Y \cap U = \emptyset$, and f an aggregate function that maps subsets of D^{U-X} to D^Y (where D is the universe of values). Then, the* *aggregate operator $\langle X, f\rangle$, when applied to R, produces a new relation R' with attributes $X \cup Y$*

$$R\langle X, f\rangle = \{[t; f(\{\pi_{U-X}(t') : t' \in R \wedge \pi_X(t') = t\})] : t \in \pi_X(R)\},$$

where $[t; f]$ is the tuple constructed by concatenating the value assignments to X and Y contained in t and f, respectively. For clarity, we write the occurrences of f as f^{Y}_{U-X}.

Intuitively, the above corresponds to grouping the tuples in R on X and applying the aggregation operator to the remaining attributes in each group. For more details, including the construction of an equivalent calculus, see [21].

We can now express the query from Example 1.1 as $R\langle\{z\}, \mathbf{area}^a_{x,y}\rangle$. As shown in this example, constraint relational algebra with aggregation may fail to be closed even for dense order constraints [21]. In this section, we impose a variable independence condition on the schema of those relations to which aggregation is applied. In particular, we will stipulate that $\mathcal{I}(X)$ must hold in R (see Definition 3.1).

This variable independence condition acts as a *restriction* on applying the aggregate operator. So, the expression $R\langle\{z\}, \mathbf{area}^a_{x,y}\rangle$ from Example 1.1 would not be permitted by our restriction, as $\mathcal{I}(\{z\})$ does not hold in R. We show that for all reasonable constraint classes, relational algebra with *restricted aggregation* is *closed*, i.e., the result of any relational algebra expression can be represented as a constraint relation.

Note that restricted aggregation, like other relational algebra operators, can be nested. It should also be noted that the relation R in Definition 6.1 can be either extensional or intensional. We show later in this section how to maintain the restriction in both of those cases.

Definition 6.2. *The* relational algebra with restricted aggregation *consists of the standard relational algebra augmented with aggregation operators $\langle X, f\rangle$ such that their every occurrence of the form $Q\langle X, f\rangle$, where Q is an expression of the algebra, must satisfy the following conditions:*

1. *$\mathcal{I}(X)$ is guaranteed to hold in $Q(D)$ for every constraint database D, and*
2. *the set $f(R)$ can be represented by a constraint relation whenever $R \subseteq D^{U-X}$ can be represented by a constraint relation.*

We will show later in this section how to determine whether those conditions hold.

Example 6.3. Let L be the cadastral relation in Example 1.2. The relational algebra query

$$L\langle\{n, t\}, \mathbf{area}^a_{x,y}\rangle$$

lists the area of land owned by each person at all times. Since the unrestricted relation represented by L satisfies $\mathcal{I}(X)$, then the above query satisfies our restriction on the use of aggregation.

Theorem 6.4. *Relational algebra with restricted aggregation is closed.*

Proof. Let R be an unrestricted relation satisfying $\mathcal{I}(X)$ and $S = \{c^X_1 \wedge c^{U-X}_1, \ldots, c^X_l \wedge c^{U-X}_l\}$ its finite constraint representation in which every tuple satisfies $\mathcal{I}(X)$. We construct a set $\{d^X_1, \ldots, d^X_k\}$ of constraints in D^X such that, for all $0 < i \le l, 0 < j \le k$:

1. $d^X_j \to c^X_i$;
2. $c^X_i \to d_{i_1} \vee \ldots \vee d_{i_t}$;
3. $\neg(d^X_i \wedge d^X_j)$ if $i < j$.

This set always exists since it can be constructed by considering all possible Boolean combinations of the constraints c^X_i. Let

$$S' = \{d^X_i \wedge c^{U-X}_j : d^X_i \to c^X_j\}.$$

Then, S' is a constraint relation that also represents R. However, in S', projections of the individual constraint tuples into D^X define pairwise-disjoint sets. Due to the vic $\mathcal{I}(X)$, the sets of valuations for variables $U - X$ are constant in each of these sets. We form

$$S'\langle X, f\rangle = \{d^X_i \wedge \overline{y} = f(\{t : c^{U-X}_j(t), (d^X_i \wedge c^{U-X}_j) \in S'\}) : d^X_i \in \pi_X(S')\}.$$

The constraint relation $S'\langle X, f\rangle$ represents $R\langle X, f\rangle$. □

From the above proof, we can also derive the following rule for inferring variable independence constraints in results of queries with aggregation that parallels those for the relational algebra operators presented in Section 5.

Aggregation:

$$\textbf{RAg}. \frac{\Gamma \vdash Q : \mathcal{I}(X_1, \ldots, X_k)}{\Gamma \vdash Q\langle X, f\rangle : \mathcal{I}(Y, X_{i_1}, \ldots, X_{i_l})}$$

for $X = X_{i_1} \cdots X_{i_l}$ and Y attributes defined by f

for Y standing for the set of variable names introduced by the aggregate operator. The soundness of the rules follows from the construction used in the proof of Theorem 6.4. In particular, if R satisfies $\mathcal{I}(Z)$ for $Z \subseteq X$, then all the constraints d^X_i could have been written as conjunctions of d^Z_i and d^{X-Z}_i and, thus, the rule is sound.

To enforce the first condition of Definition 6.2, we must be able to guarantee, given a constraint database D and a relational expression $Q\langle X, f\rangle$, that $\mathcal{I}(X)$ holds in D. If Q is some database relation R, this is accomplished by maintaining the satisfaction of $\mathcal{I}(X)$ in every instance of R (see Section 4). Otherwise, if Q is an arbitrary relational expression, variable independence can be inferred at query compile-time (see Section 5), under the assumption that all vics in D are properly maintained. Note that the above is accomplished without affecting the runtime query performance; as a result, we believe that restricted aggregation is a very promising approach to assuring the closure of queries with aggregation.

The second condition of Definition 6.2 is also important and, unfortunately, can be easily violated by natural aggregation operators.

Example 6.5 [23]. Let $R = \{xy : x^2 + y^2 < 1\}$. This relation defines a unit circle using polynomial constraint(s) with rational coefficients. However,

$$R\langle\emptyset, \mathbf{area}^a_{x,y}\rangle$$

cannot be defined using polynomial constraint(s) with rational coefficients as the result is a singleton relation $\{(\pi)\}$ that is not definable even when allowing algebraic numbers as constants.

Essentially, the rational (and also algebraic) numbers cannot provide sufficiently many "constants" to define volumes. However, unlike [2], this constraint can be approximated arbitrarily tightly using polynomial constraint(s) with rational coefficients (follows from density of $\mathbb{Q}$ in $\mathbb{R}$ and the fact that the variable independence condition guarantees there are only finitely many volume values needed). On the other hand, using semilinear sets evades this issue.

Theorem 6.6. *For bounded constraint relations defined using semilinear constraints with rational (algebraic) coefficients, the result of the* **area** *aggregate operator is always definable by linear equality with rational (algebraic) coefficients.*

We conclude this section by noting that perhaps Definition 6.2 is overly restrictive. It may happen that a relational algebra expression is closed while some of its subexpressions aren't.

Example 6.7. Consider the query

$$\sigma_{a<1}(R\langle\{z\}, \mathbf{area}^a_{x,y}\rangle)$$

for the relation R from Example 1.1. This query is excluded by Definition 6.2 because the application of aggregation violates the vic $\mathcal{I}(\{x,y\})$. However, the result of this query is obtained by solving quadratic inequalities and can thus always be represented using order constraints.

7 Applications in Temporal, Spatiotemporal, and Constraint Databases

We discuss here further applications of variable independence. As we show, this notion arises naturally in spatiotemporal databases (in the context of discrete change), point-based temporal query languages, and gap-order constraint databases.

7.1 Spatiotemporal Databases

Spatiotemporal relation schemas contain three kinds of attributes: spatial, temporal, and thematic. In the context of constraint databases, such relations are modeled as constraint relations consisting of conjunctions of constraints over three sets of variables: spatial, temporal, and thematic. In general, a single constraint may involve multiple kinds of variables. For example, to represent *continuous* change, it is necessary to have spatial and temporal variables related together.

However, as pointed out in the introduction, in many spatiotemporal database applications, it is natural to consider only *discrete* changes. This observation was made first by Worboys [37], [38], who proposed a spatiotemporal data model in which temporal attributes are represented using products-of-intervals and spatial attributes using polygons. Worboys' model has a natural reconstruction using linear constraint databases with constraint relations in which spatial, temporal, and thematic variables are independent. Consider the signature $\Omega = \{+, <, 0, 1\}$ interpreted over the reals.

Definition 7.1. *A* Worboys relation *is a constraint relation S over Ω with the schema* $\mathbf{DST}$ *such that unrestricted relation R represented by S satisfies the gvic $\mathcal{I}(\mathbf{D}, \mathbf{S}, \mathbf{T})$, where*

- $\mathbf{D}$ *is the set of variables corresponding to thematic attributes,*
- $\mathbf{S}$ *is the set of variables of cardinality at most 2, corresponding to spatial attributes, and*
- $\mathbf{T}$ *is the set of variables corresponding to temporal attributes.*

A Worboys database *is a set of Worboys relations.*

Example 7.2. The following constraint relation:

$$\{n = 935 \wedge x > 0 \wedge x < 1 \wedge y > 0 \wedge t > 0\}$$

is a Worboys relation with $\mathbf{D} = \{n\}$, $\mathbf{S} = \{x, y\}$, and $\mathbf{T} = \{t\}$.

Worboys [37] also proposed a query language for his model. The language is essentially a version of relational algebra, in which care is taken to ensure that every operator produces a Worboys relation. This is a serious limitation in expressive power since it makes it impossible to formulate queries in which intermediate results have multiple sets of spatial attributes. Such queries are essential, for example, to compare spatial objects corresponding to different time instants.

Example 7.3. Consider the class C of queries of the following form:

$$\pi_{\mathbf{S},\mathbf{T},\mathbf{D}}(\sigma_{\alpha(\mathbf{D},\mathbf{D}')}(\sigma_{\beta(\mathbf{T},\mathbf{T}')}(\sigma_{\gamma(\mathbf{S},\mathbf{S}')}(R(\mathbf{S},\mathbf{T},\mathbf{D}) \times R'(\mathbf{S}',\mathbf{T}',\mathbf{D}'))))).$$

Such queries express joins involving temporal, spatial, and thematic attributes, followed by a projection. As an example, consider a query that for any time point, returns the part of a given spatial object that overlaps with some other spatial object. It can be written as

$$\pi_{\mathbf{S},\mathbf{T},\mathbf{D}}(\sigma_{\mathbf{D}\neq\mathbf{D}'}(\sigma_{\mathbf{T}=\mathbf{T}'}(\sigma_{\mathbf{S}=\mathbf{S}'}(R(\mathbf{S},\mathbf{T},\mathbf{D}) \times R(\mathbf{S}',\mathbf{T}',\mathbf{D}'))))).$$

Queries in C cannot be expressed in the query language of the Worboys model because they require intermediate results with multiple sets of spatial $(\mathbf{S}, \mathbf{S}')$ attributes.

We propose to use the standard relational algebra to query Worboys databases. To obtain closure, we will consider only queries that compute Worboys relations. However, the intermediate results may contain multiple sets of spatial attributes and thus fail to be Worboys relations. Such a scenario is termed *data interoperability* [11] (see Fig. 5).

Worboys' model **Linear arithmetic constraint databases**

$$\begin{tikzcd}
D \arrow[r, "\text{attrs independent}"'] \arrow[d, dashed, "\text{no query}"'] & C_D \arrow[d, "\text{Relational algebra query } Q"] \\
D' & Q(C_D) \arrow[l, "\text{if attrs independent}"]
\end{tikzcd}$$

Fig. 5. Data interoperability.

To capture intermediate results that are more general than Worboys relations, we introduce the notion of *general* Worboys relation.

Definition 7.4. *Assume a fixed database schema consisting of Worboys relations $R_1, \ldots, R_n$. A* general Worboys relation *is a constraint relation S over Ω with attributes* $\mathbf{DST}$ *such that the unrestricted relation R represented by S satisfies a finite set of gvics of the form $\mathcal{I}(\mathbf{D}, \mathbf{S_1}, \ldots, \mathbf{S_k}, \mathbf{T})$, where:*

- $\mathbf{D}$ *is a set of thematic variables, each of which corresponds to a thematic variable (possibly renamed) in one of $R_1, \ldots, R_n$,*
- $\mathbf{S_i}$*, $1 \leq i \leq k$, is a set of spatial variables, consisting of some spatial variables (possibly renamed) occurring in one of $R_1, \ldots, R_n$, and*
- $\mathbf{T}$ *is a set of temporal variables, each of which corresponds to a temporal variable (possibly renamed) in one of $R_1, \ldots, R_n$.*

We will use here the relational algebra defined in Section 2. The atomic expressions in this algebra will be the given Worboys relations or built-in relations (arbitrary linear constraints). We note that built-in relations satisfy in general only trivial gvics. We will require that every equijoin $R \bowtie S$, where R (or S) is built-in is a selection $\sigma_R(S)$ (respectively, $\sigma_S(R)$) in which R (respectively, S) involves only the thematic, the spatial, or the temporal variables of S (respectively, R). (This is not restrictive since built-in relations typically involve attributes of the same type.)

Theorem 7.5. *Assume a fixed database schema consisting of Worboys relations $R_1, \ldots, R_n$ and possibly some built-in relations. Let Q be a relational algebra query with the restrictions listed above. Then, for any Worboys database D over Ω corresponding to the above schema, $Q(D)$ is a general Worboys relation.*

Proof. By structural induction, using the inference rules in Fig. 4. □

Corollary 7.6. *Under the assumptions of Theorem 7.5, let Q be a query in which we additionally require that it return a constraint relation with schema* $\mathbf{DST}$*, where* $\mathbf{D}$ *consists of (possibly renamed) thematic variables,* $\mathbf{S}$ *($|\mathbf{S}| \leq 2$) of (possibly renamed) spatial variables, and* $\mathbf{T}$ *of (possibly renamed) temporal variables. Then, for every Worboys database D over Ω, $Q(D)$ is a Worboys relation.*

Example 7.7. We illustrate how the queries of class C defined in Example 7.3 can be shown to return Worboys relations (provided the database contains Worboys relations). We will use the inference rules in Fig. 4, replacing equijoin with a built-in relation by the corresponding selection.

1. $R : \mathcal{I}(\mathbf{D}, \mathbf{S}, \mathbf{T})$ and $R' : \mathcal{I}(\mathbf{D}', \mathbf{S}', \mathbf{T}')$ (original assumptions),
2. $R \times R' : \mathcal{I}(\mathbf{S}, \mathbf{D}, \mathbf{T}, \mathbf{S}', \mathbf{D}', \mathbf{T}')$ (join $\equiv$ cartesian product),
3. $\sigma_{\gamma(\mathbf{S},\mathbf{S}')}(R \times R') : \mathcal{I}(\mathbf{SS}', \mathbf{D}, \mathbf{T}, \mathbf{D}', \mathbf{T}')$ (join $\equiv$ selection),
4. $\sigma_{\beta(\mathbf{T},\mathbf{T}')}(\sigma_{\gamma(\mathbf{S},\mathbf{S}')}(R \times R')) : \mathcal{I}(\mathbf{SS}', \mathbf{TT}', \mathbf{D}, \mathbf{D}')$ (join $\equiv$ selection),
5. $\sigma_{\alpha(\mathbf{D},\mathbf{D}')}(\sigma_{\beta(\mathbf{T},\mathbf{T}')}(\sigma_{\gamma(\mathbf{S},\mathbf{S}')}(R \times R')))$: $\mathcal{I}(\mathbf{DD}', \mathbf{SS}', \mathbf{TT}')$ (join $\equiv$ selection),
6. $\pi_{\mathbf{S},\mathbf{T},\mathbf{D}}(\sigma_{\alpha(\mathbf{D},\mathbf{D}')}(\sigma_{\beta(\mathbf{T},\mathbf{T}')}(\sigma_{\gamma(\mathbf{S},\mathbf{S}')}(R \times R'))))$: $\mathcal{I}(\mathbf{D}, \mathbf{S}, \mathbf{T})$ (projection).

Worboys relations have an advantage over general spatiotemporal relations from the point of view of query evaluation. Grumbach et al. [14] note that, if time and space are independent, then the intersection of two spatiotemporal objects, done tuple-wise, consists of intersecting the temporal and spatial parts of each tuple. If those parts are represented using well-established constructs like products-of-intervals for the temporal and polygons (or polytopes) for the spatial, the intersection can be done very efficiently.

Our notion of Worboys relation is slightly more general than Worboys' since we allow spatial attributes to describe unbounded spatial objects, and thematic and temporal variables to participate in linear constraints. It can be shown that, under the more restricted notion, we will also obtain closure for *positive* relational algebra (without the set difference operator).

7.2 Temporal Extensions of SQL

SQL/TP [32] is a temporal extension of SQL. Here, tuples in temporal relations are extended with an extra attribute (variable) storing the time instant associated with the tuple. Individual tuples are not stored separately. Rather the set of tuples whose nontemporal components are identical and temporal components are consecutive time instants is stored as a single tuple with the set of time instants represented as an interval.

The originality of SQL/TP compared to other temporal extensions of SQL is that the interval-based representation is not exposed in the query language. Instead, the user writes queries in terms of time instants as he would do if he were dealing with a constraint-based representation (see the example below).

However, SQL/TP is not implemented like typical constraint query languages, e.g., DEDALE [15]. Instead, SQL/TP queries are compiled into plain SQL where results are to be represented as (products of) intervals. Therefore, SQL/TP uses vics to restrict allowed queries as follows: 1) the temporal attributes in the result of the query are *independent*; 2) the aggregate operations must obey the vics required for aggregation in Section 6.

Example 7.8. Consider a temporal database containing a single relation r with a temporal attribute $year$. The query in which only `r1.year` appears in the select list does not violate variable independence requirement:

```
select  r1.year
from    r r1, r r2
where   r1.year < r2.year
```

The following SQL/TP query, however, is not allowed because the attributes `r1.year` and `r2.year` are not independent, violating the first requirement.

```
select   r1.year, r2.year
from     r r1, r r2
where    r1.year < r2.year
```

Evaluating this query on a nonempty instance of r, e.g., an instance containing, e.g., a single tuple for which $year = [1900, 2000]$ yields a result that cannot be represented in the SQL/TP data model.

The result of this query is a *constraint* relation representing a *triangle* (or a set of triangles), these cannot be represented by a (similarly sized) set of products of intervals. Even in cases where the instance of `r` is bounded and, thus, the result is finite, we do not admit such a query as the number of independent tuples in the result may be extremely large (cf. Example 5.12).

Example 7.9 The following query is also not allowed because it leads to a lack of closure under aggregation:

```
select     count(r1.year), r2.year
from       r r1, r r2
where      r1.year < r2.year
group by   r2.year
```

The closure problem is the same as the problems outlined in Examples 1.1 and 1.2

7.3 Gap-Order Datalog

Gap-order constraints in Z are atomic formulas of the form $x + c \leq y$, $x \leq d$, and $d \leq x$ for $c, d \in \mathbb{Z}, c > 0$. Constraint tuples formed from gap-order constraints over variables $x_1, \dots, x_k$ can be represented by a directed graph on $k + 2$ [28] or equivalently by a matrix representation of such a graph [13]. In both cases, the representation is *quadratic* in k (worst case).

A *Gap-Order Datalog* query (program) is a finite set of Horn clauses with variables constrained by gap-order constraints. Thus, both the bottom-up evaluation [28] and the top-down, memoing-based evaluation [33] of Gap-order Datalog queries need to manipulate $\Theta(k^2)$ sized tuples in the iterative construction of the least model or in the memoing tables [39], respectively. However, in many cases, the variables in the literals of the Datalog query are *independent*. Using Theorem 5.14, we can infer variable independence for all atoms in a given Datalog query and then tailor the representation to minimize the required space to store constraint tuples. Similarly to [16], it is sufficient to have only representation with $\sum_{i=1}^{k} |X_i|^2$ size for each atom r satisfying the gvic $\mathcal{I}(X_1, \dots, X_k)$.

8 Related Work

The notion of *variable independence* was first proposed in [8] by three authors of the present paper. A similar notion of *orthographic dimension* was proposed later in [16]. Both papers claimed decidability of variable independence for linear constraint databases; as pointed out by Libkin [23], the proofs of both claims were incorrect. The first correct proof of the decidability of variable independence for linear and polynomial constraint databases[5] appears in [24]. The proof works by formulating variable independence as a formula in an appropriate, decidable constraint theory. Our approach in the present paper is different, as it draws on the techniques and results from computational geometry [18]. In addition, our approach not only decides the validity of a vic but also explicitly constructs a constraint relation that satisfies the vic, if one exists.

Spatiotemporal data models that require the spatial and temporal attributes be independent are described in [14], [37], [38]. Grumbach et al. [16] provide a test whether a relational expression introduces additional dependencies between attributes. This test can be used to achieve the same conclusions that we did in the temporal and spatiotemporal database examples in Section 7.

Most of the previous work on aggregation, with the exception of [20], [26], [21], [9], [17], [2], has concentrated on a fixed set of aggregation operators motivated by traditional database applications. Such operators are applicable to finite relations and do not generalize to infinite ones. In [9], a special aggregation operator for constraint relations that preserved closure of relational algebra operations was proposed. The issue of closure of general aggregation operators was not, however, addressed. An approach to aggregation based on approximation was presented in [17]. However, that approach does not solve the closure problem, as it does not show how to compute the result of the aggregate function when nonclosure arises (cf. Example 1.1). In fact, the application of aggregation seems still to require variable independence as a precondition, as in the current paper.

Benedikt and Libkin [2] have presented a comprehensive study of aggregation in the context of constraint databases. The conclusions of that paper can be summarized as follows: First, the authors show that existing

5. In fact, the proof is considerably more general.

approximation-based approaches to computing aggregation are infeasible for polynomial and linear arithmetic constraint databases because they introduce a huge blow-up in the representation size. Second, the authors show that adding new functions to the signature does not help. Third, they propose a new query language $FO + POLY + SUM$ that consists of relational calculus with polynomial constraints augmented with standard discrete aggregation operators applied to the outputs of range-restricted constraint queries. In this language, queries computing volumes of sets defined by linear arithmetic constraints can be expressed. This elegant approach requires, however, moving beyond linear constraints to represent aggregation results (which is unavoidable). Chomicki and Libkin [10] provide a summary of the recent work on aggregation in constraint databases.

9 Conclusions and Further Work

We have presented the notion of *variable independence conditions* (vics) that enriches the constraint data model. We have shown how to use this notion in a number of contexts, including closure under aggregation, data interoperability, and query evaluation in temporal and spatio-temporal databases. We have also established the decidability of vics in linear constraint databases and studied their inference.

Further extensions of this work include decision procedures for *vic*s for larger classes of constraints, the interaction of *gvic*s with other types of *integrity constraints*, e.g., functional dependencies, equational constraints, etc., including axiomatization and/or decision procedures for such combinations, and applications to non-first-order query languages with aggregation [25], [29], [31], [34].

Acknowledgments

Jan Chomicki is supported by the US National Science Foundation award IRI-9632870. Dina Goldin is supported by the US National Science Foundation grants 9733678 and 0296195. David Toman is supported by NSERC and CITO grants.

References

[1] M. Benedikt, G. Dong, L. Libkin, and L. Wong, "Relational Expressive Power of Constraint Query Languages," *J. ACM,* vol. 45, pp. 1-34, 1998.

[2] M. Benedikt and L. Libkin, "Exact and Approximate Aggregation in Constraint Query Languages," *Proc. ACM Symp. Principles of Database Systems,* May/June 1999.

[3] J. Bochnak, M. Coste, and M.F. Roy, "Real Algebraic Geometry (English)," *Ergebnise der Mathematik und ihrer Grenzgebiete,* vol. 36, Springer, 1998.

[4] A. Brodsky, J. Jaffar, and M.J. Maher, "Towards Practical Constraint Databases," *Proc. Int'l Conf. Very Large Data Bases,* 1993.

[5] A. Brodsky and Y. Kornatzky, "The LyriC Language: Constraining Objects," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* 1995.

[6] A. Brodsky, C. Lassez, J-L. Lassez, and M.J. Maher, "Separability of Polyhedra for Optimal Filtering of Spatial and Constraint Data," *Proc. ACM Symp. Principles of Database Systems,* 1995.

[7] J. Chomicki, "Temporal Query Languages: A Survey," *Proc. Temporal Logic, First Int'l Conf.,* D.M. Gabbay and H.J. Ohlbach, eds., pp. 506-534, 1994.

[8] J. Chomicki, D. Goldin, and G. Kuper, "Variable Independence and Aggregation Closure," *Proc. ACM Symp. Principles of Database Systems,* June 1996.

[9] J. Chomicki and G. Kuper, "Measuring Infinite Relations," *Proc. ACM Symp. Principles of Database Systems,* 1995.

[10] J. Chomicki and L. Libkin, "Aggregate Languages for Constraint Databases," *Constraint Databases,* G. Kuper, L. Libkin, and J. Paredaens, eds., pp. 131-154, 2000.

[11] J. Chomicki and P.Z. Revesz, "Constraint-Based Interoperability of Spatiotemporal Databases," *Geoinformatica,* vol. 3, no. 3, pp. 211-243, Sept. 1999. Preliminary version in SSD' 97.

[12] F. Dumortier, M. Gyssens, and L. Vandeurzen, "On the Decidability of Semi-Linearity for Semi-Algebraic Sets and its Implications for Spatial Databases," *Proc. ACM Symp. Principles of Database Systems,* pp. 68-77, 1997.

[13] D.Q. Goldin and P.C. Kanellakis, "Constraint Query Algebras," *Constraints J.,* vol. 1, nos. 1/2, pp. 45-83, 1996.

[14] S. Grumbach, P. Rigaux, and L. Segoufin, "Spatio-Temporal Data Handling with Constraints," *Proc. ACM Symp. Geographic Information Systems,* Nov. 1998.

[15] S. Grumbach, P. Rigaux, and L. Segoufin, "The DEDALE System for Complex Spatial Queries," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 213-224, June 1998.

[16] S. Grumbach, P. Rigaux, and L. Segoufin, "On the Orthographic Dimension of Constraint Databases," *Proc. Int'l Conf. Database Theory,* Jan. 1999. Also INRIA Verso Report 141, 1998.

[17] S. Grumbach and J. Su, "Towards Practical Constraint Databases," *Proc. ACM Symp. Principles of Database Systems,* pp. 28-39, June 1996.

[18] D. Halperin, "Arrangements," *Handbook of Discrete and Computational Geometry,* J.E. Goodman and J. O'Rourke, eds., pp. 389-412, CRC Press, 1997.

[19] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz, "Constraint Query Languages," *J. Computer and System Sciences,* vol. 51, no. 1, pp. 26-52, 1995.

[20] A. Klug, "Equivalence of Relational Algebra and Relational Calculus Query Languages Having Aggregate Functions," *J. ACM,* vol. 29, no. 3, pp. 699-717, 1982.

[21] G. Kuper, "Aggregation in Constraint Databases," *Proc. First Int'l Workshop Principles and Practice of Constraint Programming (PPCP '93),* pp. 61-172, 1994.

[22] J.L. Lassez, "Querying Constraints," *Proc. Ninth ACM Symp. Principles of Database Systems,* pp. 288-298, 1990.

[23] L. Libkin, "Some Remarks on Variable Independence, Closure, and Orthographic Dimension in Constraint Databases," *SIGMOD Record,* vol. 28, no. 4, pp. 24-28, 1999.

[24] L. Libkin, "Variable Independence, Quantifier Elimination, and Constraint Representations," *Proc. 27th Int'l Colloquium Automata, Languages, and Programming,* 2000.

[25] I.S. Mumick, H. Pirahesh, and R. Ramakrishnan, "Duplicates and Aggregates in Deductive Databases," *Proc. Int'l Conf. Very Large Data Bases,* Aug. 1990.

[26] G. Ozsoyoglu, Z.M. Ozsoyoglu, and V. Matos, "Extending Relational Algebra and Relational Calculus with Set-Valued Attributes and Aggregate Functions," *ACM Trans. Database Systems,* vol. 12, pp. 566-592, 1987.

[27] J. Paredaens, J. Van den Bussche, and D. Van Gucht, "Towards a Theory of Spatial Database Queries," *Proc. ACM Symp. Principles of Database Systems,* pp. 279-288, 1994.

[28] P.Z. Revesz, "A Closed-Form Evaluation for Datalog Queries with Integer (Gap)-Order Constraints," *Theoretical Computer Science,* vol. 116, pp. 117-149, 1993.

[29] K.A. Ross and Y. Sagiv, "Monotonic Aggregation in Deductive Databases," *Proc. ACM Symp. Principles of Database Systems,* pp. 114-126, 1992.

[30] M. Shiota, "Geometry of Subanalytic and Semialgebraic Sets," *Progress in Mathematics,* vol. 150, Birkhäuser, 1997.

[31] S. Sudarshan, D. Srivastava, R. Ramakrishnan, and C. Beeri, "Extending the Well-Founded and Valid Model Semantics for Aggregation," *Proc. Int'l Logic Programming Symp.,* 1993.

[32] D. Toman, "Point-Based Temporal Extensions of SQL," *Proc. Int'l Conf. Deductive and Object-Oriented Databases,* 1997.

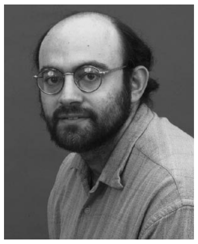
[33] D. Toman, "Memoing Evaluation for Constraint Extensions of Datalog," *Costraints,* vol. 2, nos. 3/4, pp. 337-359, 1997.

[34] A. Van Gelder, "The Well-Founded Semantics of Aggregation," *Proc. ACM Symp. Principles of Database Systems,* pp. 127-138, June 1992.

[35] J. Van den Bussche, "Constraint Databases, Queries, and Query Languages," *Constraint Databases,* G. Kuper, L. Libkin, and J. Paredaens, eds., Springer-Verlag, pp. 21-54, 2000.
[36] L. Vandeurzen, M. Gyssens, and D. Van Gucht, "On the Desirability and Limitations of Linear Spatial Database Models," *Proc. Int'l Symp. Large Spatial Databases,* pp. 14-28, 1995.
[37] M.F. Worboys, "A Unified Model for Spatial and Temporal Information," *Computer J.,* vol. 37, no. 1, pp. 26-34, 1994.
[38] M.F. Worboys, *GIS: A Computing Perspective.* Taylor & Francis, 1995.
[39] The XSB Group, "The XSB Logic Programming System, version 2.0," http://www.cs.sunysb.edu/~sbprolog, 1999.

Jan Chomicki received the MS degree in computer science with honors from Warsaw University, Poland, and the PhD degree, also in computer science, from Rutgers University. He is currently an associate professor in the Department of Computer Science and Engineering, University at Buffalo. He has held visiting positions at Hewlett-Packard Labs, European Community Research Center, Bellcore, and Lucent Bell Labs. He is the recipient of four US National Science Foundation research grants, the author of more than 45 publications, an editor of the book *Logics for Databases and Information Systems* (Kluwer, 1998) and an editor of *Information Processing Letters*. His research addresses issues in logical foundations of databases, data integrity, and novel database applications.

Dina Goldin (BS, Yale University; MS and PhD, Brown University) is a faculty member at the University of Connecticut, specializing in database models and query languages. She is a recipient of the prestigious US National Science Foundation Career grant, titled "Towards Practical Constraint Query Algebras." Professor Goldin was a member of the Information Management Panel of CC-2001 (ACM/IEEE-CS Joint Curriculum Task Force). She has also published in the areas of compiler optimization, similarity queries for time-series data, and models of interactive computation. She is a member of ACM and the New England Database Society. She also serves as a national CS program evaluator with the Computing Accreditation Commission.



Gabriel Kuper received the BSc degree in mathematics from the Technion, Israel, and the PhD degree in computer science from Stanford University. He was a researcher at the IBM T.J. Watson Research Center from 1985 until 1992, at the European Computer-Industry Research Centre (ECRC) in Munich from 1993 to 1996, and at Bell Labs in Murray Hill from 1996 to 2000. He is currently a professor at the Univerità di Trento, Italy. His current research interests include XML databases, and constraint databases.

David Toman received the BSc and MSc (Mgr.) degrees with honors from Masaryk University, Brno, Czech Republic, and the PhD degree from Kansas State University, all in computer science. He is currently an assistant professor in the School of Computer Science, University of Waterloo, Canada. He has been a NATO/NSERC postdoctoral fellow at the University of Toronto and visiting professor at BRICS, Centre for basic research in computer science funded by the Danish National Science Foundation, at University of Aarhus. He is the recipient of National Science and Engineering Research Council (NSERC), Communications and Information Technology Ontario (CITO), and Canadian Foundation for Innovation (CFI) research grants, and the author of more than 30 publications. His research interests include logical foundations of databases, query processing and optimization, embedded application of database technology, data integrity, and description logics.