Variable Independence in Constraint Databases

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

Dept. of Computer Science and Engineering, SUNY at Buffalo
226 Bell Hall, Box 602000, Buffalo, NY 14260-2000, U.S.A.

Dept. of Computer Science and Engineering, University of Connecticut
191 Auditorium Road, U-155 Storrs, CT 06269-3155, U.S.A.

Database Systems Research Department, Bell Laboratories
600 Mountain Ave., 2A-410, Murray Hill, NJ 07974, U.S.A.

Department of Computer Science, University of Waterloo
200 University Avenue West, Waterloo, Ontario, Canada N2L 3G1

November 29, 2001

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.

Keywords: Constraint databases, aggregation, closure, integrity constraints, spatiotemporal databases.

1 Introduction

Constraint databases [KKR95] 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 [BJM93, BK95, BLLM95, PVdBVG94, VGVG95], temporal [Cho94, Tom97], and spatiotemporal databases [GRS98a, GRS99]. A variety of constraint languages have been proposed, including dense order constraints, linear arithmetic constraints and the most general – polynomial constraints [GK96, BDLW98].

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*. 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 [Kup94],
- enhancing the expressive power of practical spatiotemporal query languages through *data interoperability* [CR99],
- simplifying spatiotemporal *query evaluation* [GRS98a, GRS99],
- enabling compilation to SQL for temporal query languages with multiple temporal dimensions [Tom97].

The first problem, which provided the original motivation for studying variable independence conditions, is discussed below; the remaining problems are described later in the paper.

As shown in [Kup94], relational algebra over linear constraint databases is not closed under aggregation using **area**. 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 (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 Figure 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 the full generality of such a language (and for which the *closure* problem would be unavoidable), this is not the

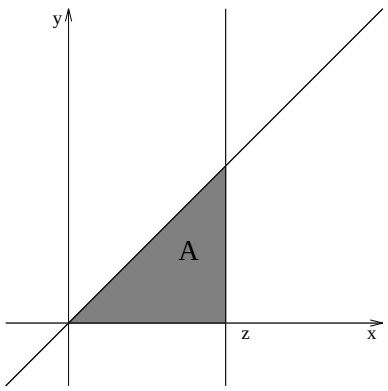

Figure 1: $area(A) = z^2/2$

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, the evolution of area ownership is finitely representable.

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

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

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 [Wor94, Wor95] 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.

The contributions of this paper are as follows:

- the notions of *variable independence condition* (vics) and *generalized variable independence condition* (gvics). The latter can succinctly capture all variable independence conditions holding in a relation.

- a new, geometric technique for proving *PTIME decidability* of vics and gvics for semi-linear relations.
- a sound and (under certain restrictions) complete set of rules for *inferring gvics* in the results of relational algebra queries.
- a proof of *closure* of relational algebra queries with restricted aggregation. The restriction requires the argument relation to satisfy a gvic.
- a description of the *applications* of vics and gvics in temporal and spatiotemporal databases.

This paper is organized as follows. In Section 2, we review the basic concepts of constraint databases and constraint query languages. In Section 3, we define the notion of *variable independence*, central to this paper. In Section 4 we show how variable independence can effectively be tested in linear and polynomial constraint databases, in PTIME data complexity using computational geometry techniques. In Section 5, we present a set of useful general rules for inferring variable independence. In Section 6, we define the relational algebra with restricted aggregation and show that it is closed. In Section 7, we present a number of other applications of variable independence in temporal and spatiotemporal databases. In Section 8, we discuss related work. We conclude in Section 9, outlining several directions for further work.

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 [KKR95]. We use here the definitions from [VdB00], 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 predicate 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-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.

Definition 2.4 *An* Ω-constraint *is an atomic first-order formula over Ω or its negation.*

Example 2.5 *The following classes of constraints are of special interest in this paper:*

- polynomial inequality constraints*: the constraints over $\Omega = (+, *, 0, 1, <)$.*
- 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.*
- equality constraints*: the constraints over $\Omega = (\{c\}_{c \in C})$.*

We will interpret the polynomial and linear constraints over the reals, the rationals or the integers, the order constraints over the rationals or the integers, and the inequality constraints over arbitrary infinite sets. It is well known [KKR95, Rev93] that all those structures admit effective quantifier elimination.

Definition 2.6

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 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 $x_1, \ldots, x_k$.*
3. *The* schema *of the constraint relation S is 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 2.7 *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

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

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 schema of a constraint relation and that 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 SC consisting of a finite number of constraint relation schemas.

Definition 2.8 *A relational calculus formula over* Ω *is a first-order formula over the signature* Ω *expanded with the relation names from* SC*.*

The semantics of relational calculus formulas viewed as queries are defined on unrestricted relations.

Definition 2.9 *Let* $\mathcal{M}$ *be an* Ω*-structure on* $\mathcal{U}$*,* ϕ *a relational calculus formula over* Ω*, and* D *an unrestricted database with schema* SC*. Let* $(\mathcal{M}, D)$ *be* $\mathcal{M}$ *expanded with the interpretations in* D *for every relation name in* SC*. The formula* ϕ *expresses the following query:*

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

Clearly, relational calculus formulas will be evaluated over constraint relations, not unrestricted ones. This can be done in two different 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. We show below how relational algebra expressions over unrestricted relations can be evaluated on the corresponding constraint relations.

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 2.7) 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 $\{t\theta | t \in S_1\}$ 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.*

This proposition implies that the relational algebra and calculus over all the classes of constraints and structures listed in Example 2.5 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. 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, and R be an unrestricted relation over U. We say that t satisfies a* variable independence condition (vic) $\mathcal{I}(X)$ *if*

$$t = \pi_X(t) \times \pi_{U-X}(t).$$

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 $\mathcal{I}(X)$.

We also 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 decidable constraint theories. 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 3.2 *Let R be an unrestricted relation, U the schema of R, and $X, Y \subseteq U$. Then*

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

define axioms for inferring vics in R.

Theorem 3.3 *The axioms in Definition 3.2 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. □

Corollary 3.4 *The vics form a boolean algebra with generators defined by the MVD basis for* **id**.

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

Example 3.5 *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). This relation satisfies the following vics (and their boolean combinations): $\mathcal{I}(\{n\})$, $\mathcal{I}(\{t\})$, and $\mathcal{I}(\{x, y\})$.

Example 3.6 *On the other hand, the relation R from Example 1.1 does not satisfy the vic $\mathcal{I}(\{x, y\})$.*

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 a 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 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 the latter 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:

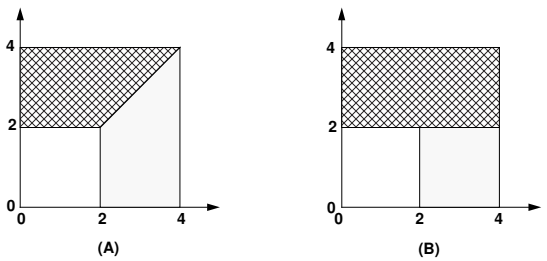

Figure 2: Instance (A) is equivalent to instance (B).

Example 3.7 *The instance of $R(x, y)$ in Figure 2 (A) contains two tuples:*

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

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

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

In Section 4, we show how to test whether a constraint relation 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) relates two sets of attributes (i.e., a set and its complement). It is natural to ask whether it can generalize to an arbitrary number of such sets. Consider the following definition:

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, and let t be a generalized tuple over U. We say that t satisfies a* generalized vic (gvic) $\mathcal{I}(X_1, \ldots, X_n)$ *if*

$$t = \pi_{X_1}(t) \times \cdots \times \pi_{X_n}(t).$$

The definition of variable independence for relations can be generalized in a similar way.

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*

$$R \text{ satisfies } \mathcal{I}(X_i) \iff R \text{ satisfies } \mathcal{I}(Y_1, \ldots, Y_l).$$

Proof: Let $Y_1, \ldots, Y_l$ be the MVD basis of $\mathbf{id} \twoheadrightarrow X_i$ for $0 < i \leq k$. The basis is unique and partitions the schema of R to 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 also the vics $\mathcal{I}(X_i)$ for $0 < i \leq k$ 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 $\{\mathcal{I}(X_i) : 0 < i \leq k\}$. The rest follows from Theorem 3.9. □

4 Testing Variable Independence

Given a relation R whose schema restricts X to be independent, we need to maintain this restriction. One way of enforcing independence restrictions on R's schema is by stipulating that each tuple satisfies these restrictions. This is the solution adopted by [GRS98b].

To allow the user maximum flexibility, we must be able to test R for independence of X. In this section, we provide such an *independence test* for linear constraint databases. Unlike the work of [Las90], where all tests are performed on constraint sets (i.e., tuples), this test is for disjunctions of constraint sets (i.e., relations).

While variable independence is a property of *unrestricted* relations, in the CDB 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 (possibly different) constraint relation S' also representing R such that all constraint tuples in S' satisfy $\mathcal{I}(X)$. Therefore, the input of the decision procedure is a *constraint relation* S rather than the relation R.

4.1 Variable Independence for Semi-linear Sets

Testing for variable independence is specific to each constraint class. In this section we present a technique based on computational geometry [BCR98, Hal97] that allows testing for variable independence of *semi-linear* relations (sets in n-dimensional Euclidean space). Those sets can be defined using linear equalities and inequalities in our constraint framework. At the end of the section we also generalize the result to relations that can be defined by more general polynomial (in)equalities.

Notation: Let X be a finite set of variables. We denote by $\mathbb{R}^X$ the $|X|$-dimensional Euclidean space with coordinates labeled by the variable names in X.

An *arrangement* $\mathcal{A}(H_1, \ldots, H_l)$ for a finite set $H_1, \ldots, H_l$ of hyper-planes in $\mathbb{R}^U$ is the coarsest common refinement of the partitions of $\mathbb{R}^U$ induced by each of the H_i. 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 [Hal97].

Lemma 4.1 *$S \subseteq \mathbb{R}^U$ is a semi-linear set if and only if there are hyper-planes $H_1, \ldots, H_l$ in $\mathbb{R}^U$ such that $S = \bigcup_{q \in I} q$ for some $I \subseteq \mathcal{A}(H_1, \ldots, H_l)$.*

Proof: Each semi-linear set S can be defined as a DNF of semi-linear constraints. The corresponding linear constraints define the hyper-planes H_i.

Given a semi-linear constraint relation S over U we define $\mathcal{A}_S$ to be an arrangement in $\mathbb{R}^U$ induced by the hyper-planes defined by the constraints in S (each constraint of the form $\mathbf{a} \cdot \mathbf{x} > k \in S$ is associated with the hyper-plane $\mathbf{a} \cdot \mathbf{x} = k$).

To test if a semi-linear relation S satisfies a vic $\mathcal{I}(X)$ we use the notion of X-rectangular decomposition:

Definition 4.2 *A pair of finite arrangements $\mathsf{part}(X)$ in $\mathbb{R}^X$ and $\mathsf{part}(U - X)$ in $\mathbb{R}^{U-X}$ is called a* X-rectangular decomposition of S *if for all*

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

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

In other words, S is equivalent to a finite union of products of pairs of cells taken from $\mathsf{part}(X)$ and $\mathsf{part}(U - X)$, respectively.

Lemma 4.3 *Let S be a semi-linear set and x and y two points in $\mathbb{R}^U$ such that (i) $x \in S$ and $y \notin S$, and (ii) $\pi_{U-X}(x) = \pi_{U-X}(y)$. Then $\pi_X(x)$ and $\pi_X(y)$ belong to different cells in $\mathsf{part}(X)$ of every X-rectangular decomposition of S.*

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

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 4.4 *Let S be a constraint semi-linear relation in $\mathbb{R}^U$, X a subset of U, and $\mathcal{A}_S$ a partition of $\mathbb{R}^U$ (derived from S). We define $\mathsf{part}_S(X)$ to be the arrangement in $\mathbb{R}^X$ induced by projecting elements of $\mathcal{A}_S$ to $\mathbb{R}^X$ one by one.*

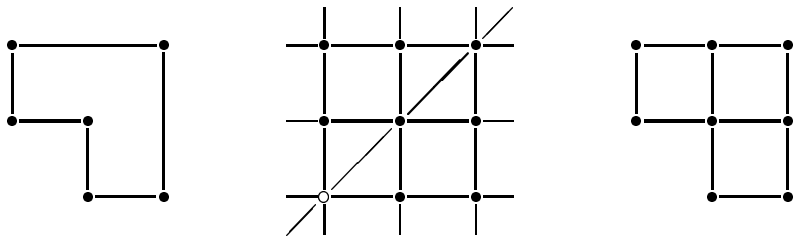

Figure 3: Dimensional Decomposition, Arrangement, and an Independent Instance in $\mathbb{R}^2$.

We claim that if S represents an unrestricted relation R that satisfies $\mathcal{I}(X)$ then the pair of arrangements $\mathsf{part}_S(X)$ and $\mathsf{part}_S(U-X)$ in the respective subspaces $\mathbb{R}^X$ and $\mathbb{R}^{U-X}$ contains a *canonical* X-rectangular decomposition of S. The crucial observation is that while an unrestricted semi-linear 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 4.5 *Let $R \subseteq \mathbb{R}^n$ be a semi-linear relation and S an arbitrary constraint relation that represents R. Then every element of the (dimensional) decomposition*[1] *of R and $\mathbb{R}^U - R$ (R's complement) of dimension $k \leq n$ is contained in a finite union of elements of $\mathcal{A}_S$ of dimension at most k.*

Thus, every arrangement $\mathcal{A}_S$ contains representation of every face of R; the lemma follows from results on real semi-algebraic sets [Shi97, BCR98] and is similar to techniques used in [DGV97]. Figure 3 shows (a) the dimensional decomposition, (b) the arrangement $\mathcal{A}_S$, and (c) the resulting semilinear set that satisfies the vic $\mathcal{I}(\{x\})$ and is equivalent to the constraint relation introduced in Example 3.7.

Theorem 4.6 *Let S be a constraint semi-linear relation in $\mathbb{R}^U$. The unrestricted relation represented by S satisfies a variable independence constraint $\mathcal{I}(X)$ if and only if $\mathsf{part}_S(X), \mathsf{part}_S(U-X)$ is an X-rectangular decomposition of S.*

Proof: The "if" direction of the proof is immediate: S is a finite union of rectangular cells of the form $x \times y$ where $x \in \mathsf{part}_S(X)$ and $y \in \mathsf{part}_S(U-X)$ 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 S is a semi-linear set that satisfies the independence constraint $\mathcal{I}(X)$ but for which $\mathsf{part}_S(X)$ and $\mathsf{part}_S(U-X)$ do not form an X-rectangular decomposition of S. Therefore there must be an $x \in \mathsf{part}_S(X)$ and $y \in \mathsf{part}_S(U-X)$ that only partially intersects S, i.e., $x \times y \not\subseteq S$ and $(x \times y) \cap S \neq \emptyset$.

[1] A *decomposition of a semi-algebraic set* is a finite union of semi-algebraic sets, each of them maximal w.r.t. 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 [BCR98], Theorem 2.3.6.

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

Also, as $\mathsf{part}_S(X), \mathsf{part}_S(U-X)$ is not an X-rectangular decomposition, $\mathsf{part}(X), \mathsf{part}(U-X)$ must be a *proper refinement* of $\mathsf{part}_S(X), \mathsf{part}_S(U-X)$: all the cells $x \times y$ that only partially intersected S must have been refined in $\mathsf{part}(X), \mathsf{part}(U-X)$. Thus, we can find two points x and y such that (i) $x \in S$ and $y \notin S$, (ii) $\pi_{U-X}(x) = \pi_{U-X}(y)$, and (iii) $\pi_X(x), \pi_X(y)$ belong to the same cell in $\mathsf{part}_S(X)$ (the other case, swapping X and $U-X$, is symmetric). However, by Lemma 4.3, x and y must then belong to different (and thus disjoint) cells in $\mathsf{part}(X)$. This is not possible as $\mathsf{part}_S(X)$ is an arrangement in $\mathbb{R}^X$ that already contains images of all faces of R (by Lemma 4.5, as every such face is an element of the decomposition of R or $\mathbb{R}^U - R$). Therefore the element(s) separating x from y must already have been reflected in $\mathsf{part}_S(X)$, a contradiction.

Therefore, $\mathsf{part}_S(X), \mathsf{part}_S(U-X)$ must be a rectangular decomposition of S whenever S 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 semi-linear constraint relation R. 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 4.7 *Let S be a constraint relation over U and $X \subseteq U$ where all constraints are semi-linear. Then testing if S 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 hyper-planes define an arrangement containing $\mathcal{O}(n^{|U|})$ cells in all dimensions; and the arrangement can be computed in time and space $\mathcal{O}(n^{|U|})$ [Hal97]. 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 $\mathsf{part}_S(X)$ and $\mathsf{part}_S(U-X)$ will be dominated by the above cost. Testing for containment/disjointness of the rectangular cells w.r.t. S is a FO+Lin query. Alltogether, 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.

Similarly to semilinear constraints, we can use the same construction to test for variable independence in constraint relations defined by *polynomial inequalities*. The complexity of the procedure remains the same [Hal97].

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[2] 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. However, 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 $\{x, y, z\}$ that 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 inference rule for the projection (π_V) operation that would obtain the desired result.*

We therefore generalize the judgments in 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. A 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 $Y_i \cap Z_j = \emptyset$.*

[2] We use gvics here as a single gvic completely characterizes all vics in a relation.

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

Figure 4: Gvic Inference Rules for PJRUN Expressions.

These operations essentially mimic the actual relational operations on constraint tuples; we therefore use the same names. The actual inference rules are defined in terms of these operations in Figure 4. Given a query expression Q, the rules are applied bottom-up following the syntactic structure of Q[3].

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 $\Gamma \vdash Q : \mathcal{I}_1, \ldots, \mathcal{I}_l$ is derived by the rules in Figure 4 and $\mathcal{I} = \mathcal{I}_1 \bowtie \ldots \bowtie \mathcal{I}_l$.*

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

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

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

Example 5.1 shows that, 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 completeness of the inference rules in Figure 4. First, similarly to other non-trivial properties of first-order queries, variable independence in query results is not decidable.

[3]There is only one deterministic way of applying the rules.

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.

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

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

Proof: Let

$$R : \mathcal{I}_1(X_1^1, \ldots, X_{k_1}^1), \ldots, \mathcal{I}_l(X_1^l, \ldots, X_{k_l}^l) \in \Gamma$$

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 a (constraint) instance D that satisfies Γ as follows: for the relation $R(x_1, \ldots, x_l)$ we define a constraint relation

$$S = \left\{ \left(\bigwedge_{\substack{x_i, x_j \in X_n^m \\ 0<i<j\leq l, 0<n\leq k_m}} x_i = x_j \right) : 0 < m \leq l \right\}$$

that *maximally* satisfies the assumptions posed on R in Γ: for each gvic $\mathcal{I}_m$ the constraint relation S contains a constraint tuple that maximally satisfies the gvic (i.e., the tuple does not satisfy any gvic finer than $\mathcal{I}_m$).

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 Figure 4 for the $f(D)$.

The result $f(D)$ can be added to the 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 $R_i : \mathcal{I}_{i,1}, \ldots, \mathcal{I}_{i,k}$ 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 Figure 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 non-trivial interpreted constraint is the strict order on rational numbers). In this setting, for any binary relation $R(x, y)$ we have*

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

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).

Similar examples can be exhibited for the $\cup$ operation as well.

5.1 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 $|R|$.

5.2 Inference of gvics in Constraint Datalog

The rules in Figure 4 can also be used to deduce gvics holding in relations defined by Constraint Datalog queries [KKR95].

We use the inference rules in Figure 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*

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

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.

Theorem 5.14 *Let P be a Constraint Datalog program. Then*

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

Proof: (1) follows from the fact that there are only finitely many different gvics over a fixed schema; (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 [Kup94], 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:

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). For clarity, we write its occurrences as f^Y_{U-X}. 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.

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 [Kup94].

We can now express the query from Example 1.1 as $R\left\langle \{z\}, \mathbf{area}^a_{x,y} \right\rangle$. As shown in this example, constraint relational algebra with aggregation may fail to be closed even for dense order constraints [Kup94]. We propose to provide a restriction on applying the aggregate operator to a relation, and show closure for the resulting class of relational expressions. In particular, we will use the notion of *variable independence*, and stipulate that $\mathcal{I}(X)$ must hold in R (see Definition 3.1). So, the expression $R\left\langle \{z\}, \mathbf{area}^a_{x,y} \right\rangle$ from Example 1.1 would not be permitted by our restriction (as $\mathcal{I}(\{z\})$ does not hold in R).

It should 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.

In this section, we impose a variable independence condition on the schema of those relations to which aggregation is applied. 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.

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 $e\,\langle X, f\rangle$ must satisfy the following conditions*

1. *X is independent in every instance of e and*

2. *the set* $\{\overline{y} : \overline{y} = f(R)\}$ *can be represented by a constraint relation whenever* $R \subseteq D^{U-X}$ *can be represented by such 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*

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

lists the area of land owned by each person at all times. Since the attribute set $\{n,t\}$ *is independent, this 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 $S = \{c_1^X \wedge c_1^{U-X}, \ldots, c_l^X \wedge c_l^{U-X}\}$ its finite constraint representation in which every tuple satisfies $\mathcal{I}(X)$. We construct a set $\{d_1^X, \ldots, d_k^X\}$ of constraints in D^X such that

- $d_j^X \rightarrow c_i^X$ for all $0 < i \leq l, 0 < j \leq k$,
- $c_i^X \rightarrow d_{i_1} \vee \ldots \vee d_{i_t}$ for all $0 < i \leq l$,
- $\neg(d_i^X \wedge d_j^X)$ for all $0 < i < j \leq k$.

This set always exists since it can be constructed by considering all possible boolean combinations of the constraints c_i^X. Let

$$S' = \{d_i^X \wedge c_j^{U-X} : d_i^X \rightarrow c_j^X\}.$$

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

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

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:

$$\text{RAg.}\quad \frac{\Gamma \vdash Q : \mathcal{I}_j(X_1, \ldots, X_k)}{\Gamma \vdash Q\,\langle X, f\rangle : \mathcal{I}_j(Y, X_{i_1}, \ldots, X_{i_l})} \qquad \begin{array}{l}\text{for } X = X_{i_1} \cdots X_{i_l} \\ \text{and } Y \text{ attributes defined by } f.\end{array}$$

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_i^X could have been written as conjunctions of d_i^Z and d_i^{X-Z} and thus the first rule is sound. The second rule is an immediate consequence of the part of the construction that defines the values for the new variables introduced by the aggregation operator, $\overline{y} = f(\{t : c_j^{U-X}(t)\})$.

To check the first condition of Definition 6.2, we must be able to decide whether $\mathcal{I}(X)$ holds in e given a constraint database $\{R_1, \ldots, R_n\}$ and a relational expression $e\,\langle X, f\rangle$. If e is some database relation R_i, this is accomplished by stipulating that $\mathcal{I}(X)$ holds R_i's schema (see Section 3). Otherwise, if e is an arbitrary relational expression, variable independence can be inferred at query compile-time (see Section 5), under the assumption that all schema restrictions for $\{R_1, \ldots, R_n\}$ are properly maintained. Note that the above is accomplished without affecting the run-time 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 *[Lib99] Let $R = \{xy : x^2 + y^2 < 1\}$. This relation defines a unit circle using polynomial constraint(s) with* rational *coefficients. However,*

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

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 [BL99], 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 semi-linear sets evades this issue:

Theorem 6.6 *For bounded constraint relations defined using semi-linear 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*

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

for the relation R from Example 1.1. This query is excluded by Definition 6.2 because the application of aggregation violates the requisite vic. 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 and spatiotemporal 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 [Wor94, Wor95], 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. Denote by C_{lin} the class of constraint databases over linear constraints interpreted over the reals.

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

- **D** *is the set of variables corresponding to thematic attributes,*
- **S** *is the set of variables of cardinality at most 2, corresponding to spatial attributes, and*
- **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:*

$$\{n = \text{'ab'} \wedge x > 0 \wedge x < 1 \wedge y > 0 \wedge t > 0\}$$

is a Worboys relation with $\mathbf{D} = \{n\}$, $\mathbf{S} = \{x, y\}$, *and* $\mathbf{T} = \{t\}$. *Note that our notion of Worboys relation is slightly more general that Worboys' since we allow spatial attributes to describe unbounded spatial objects, and thematic and temporal variables to participate in linear constraints.*

Worboys [Wor94] 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

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

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* [CR99]. To capture intermediate results that are more general than Worboys relation, 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 r over* C_{lin} *with the schema* $\mathbf{DST}$ *such that unrestricted relation* r *represented by r 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* $R_1, \ldots, R_n$,

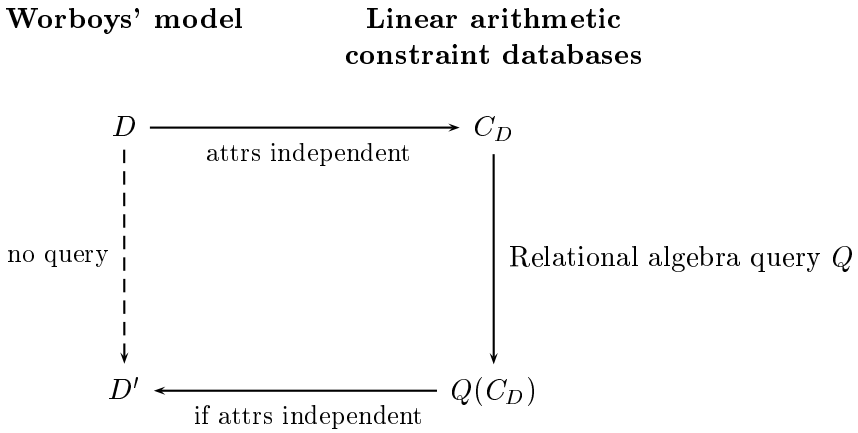

Figure 5: Data interoperability

- $\mathbf{S_i}$, $1 \leq i \leq k$, *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 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 $\sigma_R(S)$ (resp. $\sigma_S(R)$) in which R (resp. S) involves only the thematic, the spatial or the temporal variables of S (resp. 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 relation schemas* $R_1, \ldots, R_n$ *and built-in relation schemas* $W_1, \ldots, W_n$. *Let* Q *be a relational algebra query with the restrictions listed above. Then for any constraint database* D *over* C_{lin} *corresponding to the above schema (and consisting of Worboys and built-in relations),* $Q(D)$ *is a general Worboys relation.*

Proof: By structural induction, using the inference rules in Figure 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* r *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 constraint database* D *over* C_{lin}, $Q(D)$ *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 Figure 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≡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≡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≡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≡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. [GRS98a] 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.

To fully align our notion of Worboys relation with that of Worboys, it is not enough to assume that the relation satisfies a gvic $\mathcal{I}(\mathbf{D}, \mathbf{S}, \mathbf{T})$. Additionally, we need to ensure that data variables are defined by equality constraints of the form $x = c$, where c is a constant, and that temporal variables are defined using order constraints. Moreover, the spatial variables should define a bounded spatial object (a convex polygon). It can be shown that under this more restricted notion of Worboys relation we will obtain closure (and an analogue of Corollary 7.6) if we consider *positive* relational algebra (without the set difference operator).

7.2 Temporal Extensions of SQL

SQL/TP [Tom97] 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 non-temporal 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).

Example 7.8 *The query in which* `r1.year` *appears in the select list does not violate variable independence:*

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

However, SQL/TP is not implemented like typical constraint query languages, e.g., DEDALE [GRS98b]. 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:

- This is only possible if the temporal attributes in different range variables are *independent.*
- the aggregate operations must obey the vics required for aggregation in 6.

The first requirement disallows the following situation:

Example 7.9 *The following SQL/TP query is not allowed because the attributes* `r1.year` *and* `r2.year` *are not independent.*

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

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 it as the number of independent tuples in the result may be extremely large (cf. Examle 5.12).*

The second requirement is necessary to guarantee closure for aggregate operations. For example

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

cannot be allowed.

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 $x_1, \ldots, x_k$ can be represented by a directed graph on $k + 2$ [Rev93] or equivalently by a matrix representation of such a graph [GK96]. 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 [Rev93] and the top-down, memoing-based evaluation [Tom97b] 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 [XSB], 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 [GRS99], it is sufficient to have only representation with $\sum_{i=1}^{k} |X_i|^2$ size for each atom r satisfying the gvic $\mathcal{I}(X_1, \ldots, X_k)$.

8 Related work

The notion of *variable independence* was first proposed in [CGK96] by three authors of the present paper. A similar notion of *orthographic dimension* was proposed later in [GRS99]. Both papers claimed decidability of variable independence for linear constraint databases; as pointed out by Libkin [Lib99] the proofs of both claims were incorrect. The first correct proof of the decidability of variable independence for linear and polynomial constraint databases[4] appears in [Lib00]. 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 [Hal97]. 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 [GRS98a, Wor94, Wor95]. [GRS99] provides 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 [Klu82, OOM87, Kup94, CK95, GS96, BL99], has concentrated on a fixed set of aggregation operators motivated by traditional database applications. Such operators are applicable to finite relations

[4] In fact, the proof is considerably more general.

and do not generalize to infinite ones. In [CK95], 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 [GS96]. However, that approach does not solve the closure problem, as it does not show how to compute the result of the aggregate function when non-closure 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.

Recently, [BL99] has 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 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). [CL00] provides 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 spatiotemporal databases. We have also established the decidability of vics in linear constraint databases and studied their inference.

It will be interesting to see whether our inference rules for queries can be strengthened and whether for linear constraint databases relational algebra with restricted aggregation is in some sense a maximal closed subset of relational algebra with unrestricted aggregation.

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 [MPR90, RS92, SSRB93, VG92].

References

[BDLW98] M. Benedikt, G. Dong, L. Libkin, and L. Wong. Relational Expressive Power of Constraint Query Languages. *Journal of the ACM*, 45:1–34, 1998.

[BL99] M. Benedikt and L. Libkin. Exact and Approximate Aggregation in Constraint Query Languages. In *ACM Symposium on Principles of Database Systems*, May/June 1999.

[BCR98] J. Bochnak, M. Coste, and M.F. Roy. Real Algebraic Geometry (English). Ergebnise der Mathematik und ihrer Grenzgebiete, vol. 36, Springer, 1998.

[BJM93] A. Brodsky, J. Jaffar, and M.J. Maher. Towards Practical Constraint Databases. In *International Conference on Very Large Data Bases*, Dublin, Ireland, 1993.

[BK95] A. Brodsky and Y. Kornatzky. The LyriC Language: Constraining Objects. In *ACM SIGMOD International Conference on Management of Data*, San Jose, California, 1995.

[BLLM95] A. Brodsky, C. Lassez, J-L. Lassez, and M.J. Maher. Separability of Polyhedra for Optimal Filtering of Spatial and Constraint Data. In *ACM Symposium on Principles of Database Systems*, San Jose, California, 1995.

[Cho94] J. Chomicki. Temporal Query Languages: A Survey. In D.M. Gabbay and H.J. Ohlbach, editors, *Temporal Logic, First International Conference*, pages 506–534. Springer-Verlag, LNAI 827, 1994.

[CGK96] J. Chomicki, D. Goldin, G. Kuper. Variable Independence and Aggregation Closure. In *ACM Symposium on Principles of Database Systems*, Montreal, Canada, June 1996.

[CK95] J. Chomicki and G. Kuper. Measuring Infinite Relations. In *ACM Symposium on Principles of Database Systems*, 1995.

[CL00] J. Chomicki and L. Libkin. Aggregate Languages for Constraint Databases. In G. Kuper, L. Libkin, and J. Paredaens, editors, *Constraint Databases*, pages 131–154. Springer-Verlag, 2000.

[CR99] J. Chomicki and P. Z. Revesz. Constraint-Based Interoperability of Spatiotemporal Databases. *Geoinformatica*, 3(3):211–243, September 1999. Preliminary version in SSD'97.

[DGV97] F. Dumortier, M. Gyssens, and L. Vandeurzen. On the Decidability of Semi-Linearity for Semi-Algebraic Sets and its Implications for Spatial Databases. In *ACM Symposium on Principles of Database Systems*, pages 68–77, 1997.

[GK96] D.Q. Goldin and P.C. Kanellakis. Constraint Query Algebras. *Constraints Journal*, 1(1/2):45–83, 1996.

[GRS98a] S. Grumbach, P. Rigaux, and L. Segoufin. Spatio-Temporal Data Handling with Constraints. In *ACM Symposium on Geographic Information Systems*, November 1998.

[GRS98b] S. Grumbach, P. Rigaux, and L. Segoufin. The DEDALE System for Complex Spatial Queries. In *ACM SIGMOD International Conference on Management of Data*, pages 213–224, June 1998.

[GRS99] S. Grumbach, P. Rigaux, and L. Segoufin. On the Orthographic Dimension of Constraint Databases. In *International Conference on Database Theory*, January 1999. Also INRIA Verso Report 141, 1998.

[GS96] S. Grumbach and J. Su. Towards Practical Constraint Databases. In *ACM Symposium on Principles of Database Systems*, pages 28–39, Montreal, Canada, June 1996.

[Hal97] D. Halperin. Arrangements. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 389–412. CRC Press, 1997.

[KKR95] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint Query Languages. *JCSS*, 51(1): 26–52, 1995.

[Klu82] A. Klug. Equivalence of Relational Algebra and Relational Calculus Query Languages Having Aggregate Functions. *Journal of the ACM*, 29(3):699–717, 1982.

[Kup94] G. Kuper. Aggregation in Constraint Databases. In *PPCP'93, First International Workshop on Principles and Practice of Constraint Programming*, pages 161–172. MIT Press, 1994.

[Las90] J.L. Lassez. Querying Constraints. In *9th ACM Symposium on Principles of Database Systems*, pages 288–298, 1990.

[Lib99] L. Libkin. Some Remarks on Variable Independence, Closure, and Orthographic Dimension in Constraint Databases. SIGMOD Record 28(4): 24–28, 1999.

[Lib00] L. Libkin. Variable Independence, Quantifier Elimination, and Constraint Representations. In *27th International Colloquium on Automata, Languages and Programming*, 2000.

[MPR90] I.S. Mumick, H. Pirahesh, and R. Ramakrishnan. Duplicates and Aggregates in Deductive Databases. In *International Conference on Very Large Data Bases*, August 1990.

[OOM87] G. Ozsoyoglu, Z.M. Ozsoyoglu, and V. Matos. Extending Relational Algebra and Relational Calculus with Set-Valued Attributes and Aggregate Functions. *ACM Transactions on Database Systems*, 12:566–592, 1987.

[PVdBVG94] J. Paredaens, J. Van den Bussche, and D. Van Gucht. Towards a Theory of Spatial Database Queries. In *ACM Symposium on Principles of Database Systems*, pages 279–288, Minneapolis, Minnesota, 1994.

[Rev93] P. Z. Revesz. A Closed-Form Evaluation for Datalog Queries with Integer (Gap)-Order Constraints. *Theoretical Computer Science*, 116:117–149, 1993.

[RS92] K. A. Ross and Y. Sagiv. Monotonic Aggregation in Deductive Databases. In *ACM Symposium on Principles of Database Systems*, pages 114–126, 1992.

[Shi97] M. Shiota. Geometry of Subanalytic and Semialgebraic Sets. *Progress in Mathematics*, Vol. 150, Birkhäuser, 1997.

[SSRB93] S. Sudarshan, D. Srivastava, R. Ramakrishnan, and C. Beeri. Extending the Well-Founded and Valid Model Semantics for Aggregation. In *International Logic Programming Symposium*, 1993.

[Tom97] D. Toman. Point-based Temporal Extensions of SQL. In *International Conference on Deductive and Object-Oriented Databases*, 1997.

[Tom97b] D. Toman. Memoing Evaluation for Constraint Extensions of Datalog. *Costraints* 2(3/4), pages 337–359, 1997.

[VG92] A. Van Gelder. The Well-Founded Semantics of Aggregation. In *ACM Symposium on Principles of Database Systems*, pages 127–138, San Diego, California, June 1992.

[VdB00] J. Van den Bussche. Constraint Databases, Queries, and Query Languages. In Kuper, G. and Libkin, L. and Paredaens, J., editors, *Constraint Databases*, Springer-Verlag, pages 21–54, 2000.

[VGVG95] L. Vandeurzen, M. Gyssens, and D. Van Gucht. On the Desirability and Limitations of Linear Spatial Database Models. In *International Symposium on Large Spatial Databases*, pages 14–28, 1995.

[Wor94] M. F. Worboys. A Unified Model for Spatial and Temporal Information. *Computer Journal*, 37(1):26–34, 1994.

[Wor95] Michael F. Worboys. *GIS: A Computing Perspective*. Taylor&Francis, 1995.

[XSB] The XSB Group. The XSB Logic Programming System, version 2.0. URL: `http://www.cs.sunysb.edu/~sbprolog`, 1999.